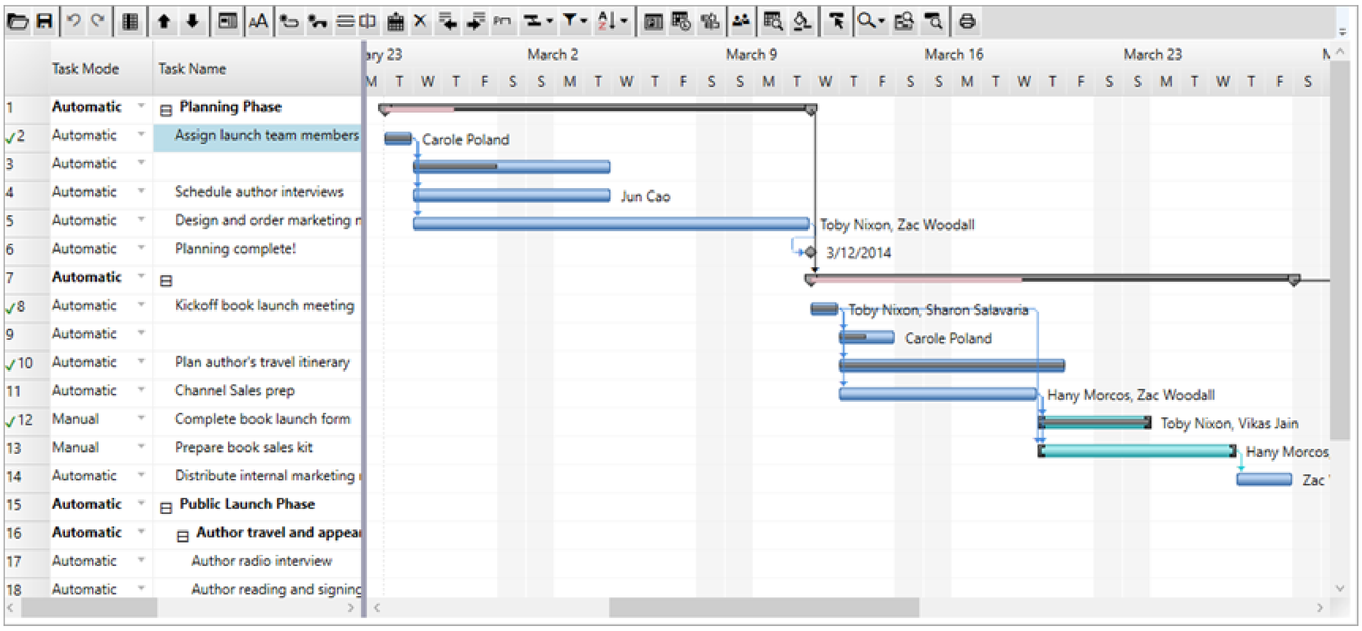
ComponentOne

GanttView for WPF

Table of Contents

GanttView for WPF

GanttView for WPF is a project management control designed to efficiently create, schedule, execute and complete projects. The GanttView control is derived from gantt charts, which are one of the oldest and most effective tools for project scheduling and management.



	Task Mode	Task Name
1	**Automatic**	⊟ **Planning Phase**
✓2	Automatic	Assign launch team members
3	Automatic	
4	Automatic	Schedule author interviews
5	Automatic	Design and order marketing r
6	Automatic	Planning complete!
7	**Automatic**	⊟
✓8	Automatic	Kickoff book launch meeting
9	Automatic	
✓10	Automatic	Plan author's travel itinerary
11	Automatic	Channel Sales prep
✓12	Manual	Complete book launch form
13	Manual	Prepare book sales kit
14	Automatic	Distribute internal marketing
15	**Automatic**	⊟ **Public Launch Phase**
16	**Automatic**	⊟ **Author travel and appea**
17	Automatic	Author radio interview
18	Automatic	Author reading and signing

GanttView enables users to track various aspects related to project management and scheduling as follows:

- Various tasks/activities involved in the project.
- Beginning and end of each task/activity.
- Time duration assigned to each task/activity.
- Overlapping tasks/activities.
- Closely follow deadlines and overall project completion.
- Start and end date of the entire project.

Help with WPF Edition

For information on installing ComponentOne Studio WPF Edition, licensing, technical support, namespaces, and creating a project with the controls, please visit Getting Started with WPF Edition.

Key Features

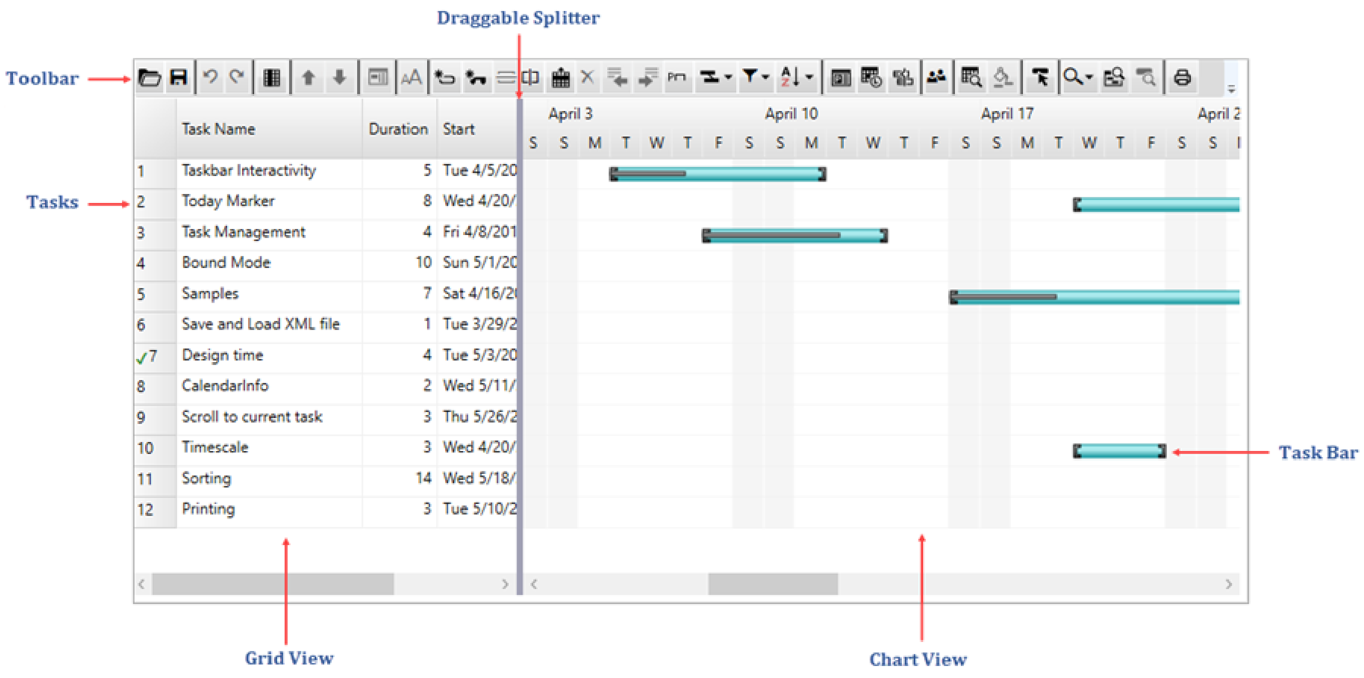
Being a data management tool, GanttView comes with a number of in-built features to make it an efficient tool for various real-world applications such as project scheduling and resource management. Here is a quick look at the key feature that GanttView provides.

Key Features	**Description**
Scheduling Modes	GanttView supports both automatic and manual scheduling modes to provide greater flexibility and control over project planning. For more information, see Scheduling mode.
Built-In Toolbar	GanttView comes with a built-in toolbar to help users quickly access various operations that can be performed at runtime. For information on toolbar, see GanttView toolbar.
Custom Calendars	GanttView provides you with the capability to create and save custom calendars for specifying work weeks, work timings and exceptions.
Timescales	GanttView provides timescales with three visible tiers, namely Top, Bottom and Middle tiers. The appearance of these tiers can customized through runtime dialogs. For more information on timescale, see Timescale.
Percent Complete	GanttView displays task related attributes through task bars. In addition, you can display the task bar as progress bar to visualize the current status of your tasks.
Milestones	GanttView supports creating Milestones to define landmarks in various phases of project management and scheduling. Milestones are created as tasks with zero duration.
Resource Handling	GanttView allows you to manage resource pools throughout your project plan. You can handle three types of resources including people, material and cost. For information on resource handling, see Resources.
Dependencies	GanttView supports task dependencies to create a feasible scheduling relationship between dependent tasks. For more information on handling dependencies in GanttView, see Tasks dependency.
Load and Save Project Schedule as XML	GanttView lets you store as well as load your project schedule in XML format. The built-in toolbar provides buttons to directly save or load project schedules from XML files.
Import Project Schedules from Microsoft Project	GanttView lets you import project schedule from Microsoft Project as well. To know about this, see Loading content from XML.
Custom Appearance	GanttView provides various options to customize the default appearance of its elements such as task bars, fields. To know more, see Style and appearance.
Themes	GanttView supports built-in ComponentOne themes to provide consistent and custom look and feel to your application. To know more, see Applying themes.

GanttView Elements

The GanttView control provides an interactive user interface to display the complete project lifecycle. GanttView comprises two major elements, a grid and a chart, separated by a draggable splitter. The grid and the chart remain in sync with each other. If the user makes any changes in the task within the grid, the same gets reflected in the chart.

Here is how a GanttView control presents various project-related attributes, such as task/activity name, mode, duration, etc.



	Task Name	Duration	Start
1	Taskbar Interactivity	5	Tue 4/5/20
2	Today Marker	8	Wed 4/20/
3	Task Management	4	Fri 4/8/201
4	Bound Mode	10	Sun 5/1/20
5	Samples	7	Sat 4/16/2
6	Save and Load XML file	1	Tue 3/29/2
✓7	Design time	4	Tue 5/3/20
8	CalendarInfo	2	Wed 5/11/
9	Scroll to current task	3	Thu 5/26/2
10	Timescale	3	Wed 4/20/
11	Sorting	14	Wed 5/18/
12	Printing	3	Tue 5/10/2

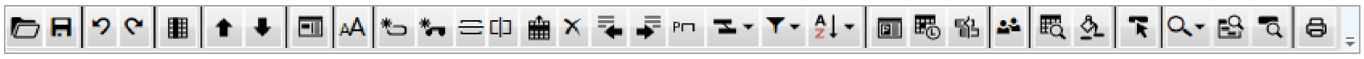
Major UI Elements

- **Grid**: The grid displays various attributes pertaining to a particular task/activity. You can edit these attributes through the grid.
- **Chart**: The chart provides visual representation of the task details presented on the grid. These details are presented through bars on the chart.
- **Splitter**: The draggable splitter divides the chart and grid.
- **Task**: Each task/activity is represented by a bar whose position and length indicate the start date, end date, and task duration.
- **Timescale**: GanttView displays a timescale along a horizontal timeline, which comprises three tiers.
- **Toolbar**: The toolbar provides quick access to various operations that one can perform in GanttView. For more information on Toolbar, refer to GanttView Toolbar.

GanttView Toolbar

GanttView comes with a built-in toolbar that provides quick access to various operations you can perform in GanttView. For example, you can use the toolbar to load and save files, apply filters on tasks, sort tasks, move tasks up and down, set the time scale, etc. The toolbar remains hidden by default. However, this behavior can be changed by setting the ShowToolbar property to true/false as required.

Here is how the toolbar in GanttView control appears.



The toolbar provides specific buttons to perform various operations. On hovering over each of these buttons, you find a tooltip that provides the information about the operation that can be performed by that button. The following table summarizes each of these toolbar buttons along with their tooltips and operations.

Button	Tooltip	Operation
	Load	Opens the **Load From XML File** dialog that lets you browse the XML file that you wish to load in GanttView.
	Save	Opens the **Save As XML** dialog that lets you save the GanttView as an XML file.
↶	Undo	Undo the last action performed.
↷	Redo	Redo the last action performed.
	Grid Columns	Opens the **Grid Columns** dialog to enable or disable default columns available in GanttView.
⬆	Move Task Up	Moves a particular task one place up in the grid.
⬇	Move Task Down	Moves a particular task one place down in the grid.
	Task Information	Opens **Task Information** dialog that summarizes various attributes such as task name, duration, start date, finish date, etc. pertaining to a particular task/activity.
AA	Field Styles	Opens the **Field Styles** dialog that lets you customize the text fields appearing in the grid.
	Add Task	Adds a new task in GanttView.
	Add Summary Task	Adds a summary task in GanttView.
	Inactive	Inactivates/reactivates the selected task in the grid.
	Split Task	Splits the selected task in the grid.
	Add Blank Row	Adds a blank row in GanttView.
✕	Delete Task	Deletes a particular task from the grid.
	Outdent Task	Increases the outline level of a task.
	Indent Task	Decreases the outline level of a task.
	Show Project Summary	Shows/hides the project summary task.
	Group By	Opens the Filter menu that lets you filter tasks by **Completed Tasks**, **Incomplete Tasks**, **Late Tasks**, **Milestones**, **Summary Tasks**, **Tasks with Duration Only**. The menu also provides **No Filter** option to clear the existing filter; **Date Range** option that opens the Date Range filter dialog to apply custom filter by date range; **Using Resource** option that opens the Using Resource

		dialog to apply custom filter by specific resource; **Advance Filter** option that opens the Advanced Filter dialog to apply custom filters; **More Filters** option that opens the More Filter dialog to choose specific advanced filters. The Show Related Summary Rows option lets you view the related summary.
	Filter	Applies filter on the tasks displayed in GanttView.
	Sort	Opens the Sort menu that lets you sort tasks by **Name**, **Start Date**, **Finish Date**, and **Duration**. The menu also provides **Remove Sort** option to clear existing sort, and **Sort By** option that opens the Sort By dialog to apply custom sort.
	Project Information	Opens Project Information dialog that lets you view various project details such as Start Date, Finish Date, Default Working Times, Default Days Off, etc.
	Change Working Time	Opens **Change Working Time** dialog that lets you view and change the default working time, working weeks, etc.
	Progress Line	Opens the **Progress Line** dialog to create and customize progress lines in GanttView.
	Project Resources	Opens the **Resources** dialog that lets you assign resources to tasks.
	Timescale	Opens **Timescale** dialog that lets you customize the Timescale settings in your GanttView.
	Bar Styles	Opens the **Bar Styles** dialog that lets you customize the task bars.
	Scroll to Task	Scrolls to tasks represented by bars in the chart area of GanttView.
	Zoom	Provides **Zoom In**, **Zoom Out** and **Zoom** options. Selecting the **Zoom** option opens the **Zoom** dialog that lets you zoom the GanttView to custom ranges.
	Zoom Entire Project	Zooms the entire project schedule in the GanttView.
	Zoom Selected task	Zooms the selected task in the GanttView.
	Print	Opens the **Print** dialog that lets you print the information displayed in GanttView.

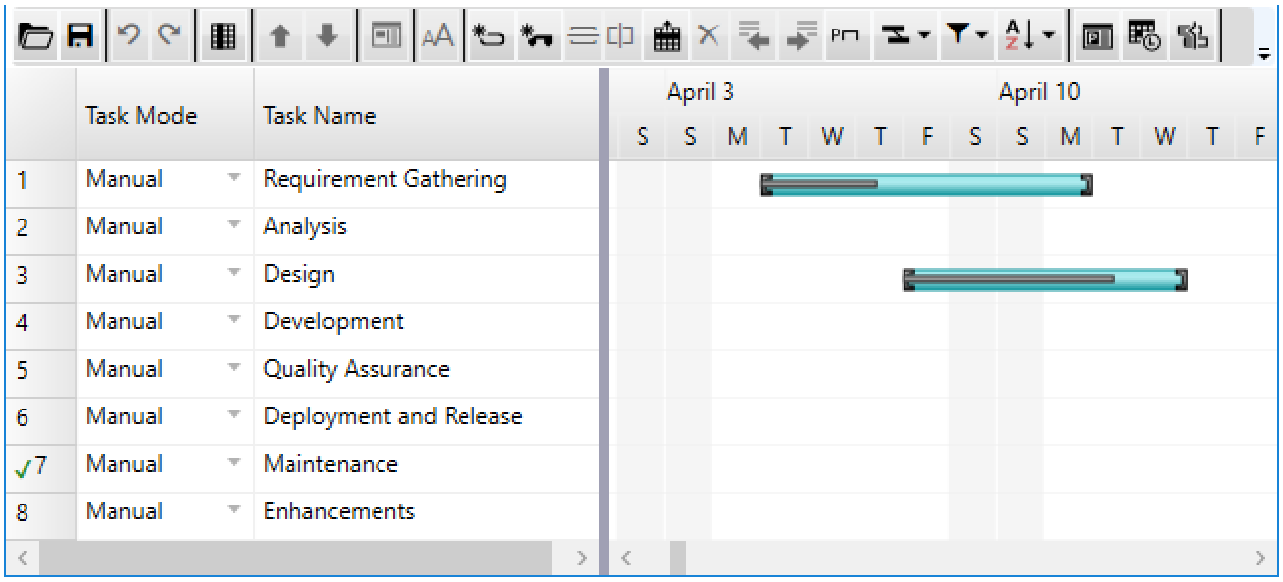
Quick Start: GanttView for WPF

This quick start section is intended to familiarize you with the GanttView control. You begin by creating a WPF application in Visual Studio, adding GanttView control to the MainWindow, and add tasks to GanttView in unbound mode through code.

Complete the steps given below to see how GanttView control appears on running the application

- **Step 1: Setting up the application**
- **Step 2: Adding tasks to GanttView in unbound mode through code**
- **Step 3: Running the application**

Here is how a basic GanttView control appears with tasks displayed.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Step 1: Setting up the application

1. Create a WPF application in Visual Studio.
2. Drag and drop the C1GanttView control to the MainWindow
3. Edit the XAML code to include relevant namespaces and set some basic properties of the GanttView control.

XAML

```
    <Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        xmlns:GanttView="clr-namespace:C1.GanttView;assembly=C1.WPF.GanttView.4"
        x:Class="QuickStart.MainWindow"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

        <c1:C1GanttView Name="gv">
            <c1:C1GanttView.Columns>
                <GanttView:TaskPropertyColumn ID="682738157" Property="Mode"/>
                <GanttView:TaskPropertyColumn ID="200464275" Property="Name"/>
                <GanttView:TaskPropertyColumn ID="1932261525" Property="Duration"
Visible="False"/>
                <GanttView:TaskPropertyColumn ID="2041428170"
```

```
Property="DurationUnits" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="877817002" Property="Start"
Visible="False"/>
                <GanttView:TaskPropertyColumn ID="1833319410" Property="Finish"
Visible="False"/>
                <GanttView:TaskPropertyColumn ID="204763723"
Property="PercentComplete" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="2112691121"
Property="ConstraintType" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="1026682979"
Property="ConstraintDate" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="46763901"
Property="Predecessors" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="1809421465" Property="Deadline"
Visible="False"/>
                <GanttView:TaskPropertyColumn ID="1041667598" Property="Calendar"
Visible="False"/>
                <GanttView:TaskPropertyColumn ID="223524084"
Property="ResourceNames" Visible="False"/>
                <GanttView:TaskPropertyColumn ID="2142090402" Property="Notes"
Visible="False"/>
            </c1:C1GanttView.Columns>
        </c1:C1GanttView>


    </Grid>
</Window>
```

Step 2: Adding tasks to GanttView in unbound mode through code

You can add tasks to the GanttView control in unbound mode. For this, you can define tasks in a delegate and add them to the Tasks collection of GanttView using the Add method.

1. Switch to the code view, that is MainWindow.xaml.cs file, and add the following import statements.

   Visual Basic

   ```
   Imports C1.GanttView
   Imports C1.WPF.GanttView
   ```

   C#

   ```
   using C1.GanttView;
   using C1.WPF.GanttView;
   ```

2. Switch to the MainWindow.xaml.cs and add the following code to subscribe the Loaded event of the C1GanttView class.

   Visual Basic

   ```
   AddHandler gv.Loaded, AddressOf gv_Loaded
   ```

   C#

   ```
   gv.Loaded += gv_Loaded;
   ```

3. Add the following code in the Loaded event for configuring tasks and adding them to the GanttView control.

Visual Basic

```
Private Sub gv_Loaded(sender As Object, e As RoutedEventArgs)
    'Configure tasks
    Dim t As New Task()
    t.Mode = TaskMode.Manual
    t.Name = "Requirement Gathering"
    t.Start = New DateTime(2016, 4, 5)
    t.Duration = 5
    t.PercentComplete = 0.5

    Dim t1 As New Task()
    t1.Mode = TaskMode.Manual
    t1.Name = "Analysis"
    t1.Start = New DateTime(2016, 4, 20)
    t1.Duration = 8
    t1.PercentComplete = 0

    Dim t2 As New Task()
    t2.Mode = TaskMode.Manual
    t2.Name = "Design"
    t2.Start = New DateTime(2016, 4, 8)
    t2.Duration = 4
    t2.PercentComplete = 0.6

    Dim t3 As New Task()
    t3.Mode = TaskMode.Manual
    t3.Name = "Development"
    t3.Start = New DateTime(2016, 5, 1)
    t3.Duration = 10
    t3.PercentComplete = 0.2

    Dim t4 As New Task()
    t4.Mode = TaskMode.Manual
    t4.Name = "Quality Assurance"
    t4.Start = New DateTime(2016, 4, 16)
    t4.Duration = 7
    t4.PercentComplete = 0.2

    Dim t5 As New Task()
    t5.Mode = TaskMode.Manual
    t5.Name = "Deployment and Release"
    t5.Start = New DateTime(2016, 3, 29)
    t5.Duration = 1
    t5.PercentComplete = 0.5

    Dim t6 As New Task()
    t6.Mode = TaskMode.Manual
    t6.Name = "Maintenance"
    t6.Start = New DateTime(2016, 5, 3)
    t6.Duration = 4
    t6.PercentComplete = 1

    Dim t7 As New Task()
    t7.Mode = TaskMode.Manual
    t7.Name = "Enhancements"
    t7.Start = New DateTime(2016, 5, 11)
    t7.Duration = 2
    t7.PercentComplete = 0.8

    'Add the tasks to the GanttView
    gv.Tasks.Add(t)
    gv.Tasks.Add(t1)
    gv.Tasks.Add(t2)
    gv.Tasks.Add(t3)
    gv.Tasks.Add(t4)
    gv.Tasks.Add(t5)
    gv.Tasks.Add(t6)
    gv.Tasks.Add(t7)
End Sub
```

C#

```
void gv_Loaded(object sender, RoutedEventArgs e)
{
    //Configure tasks
    Task t = new Task();
    t.Mode = TaskMode.Manual;
    t.Name = "Requirement Gathering";
    t.Start = new DateTime(2016, 4, 5);
    t.Duration = 5;
    t.PercentComplete = 0.5;

    Task t1 = new Task();
    t1.Mode = TaskMode.Manual;
    t1.Name = "Analysis";
    t1.Start = new DateTime(2016, 4, 20);
    t1.Duration = 8;
    t1.PercentComplete = 0;

    Task t2 = new Task();
    t2.Mode = TaskMode.Manual;
    t2.Name = "Design";
    t2.Start = new DateTime(2016, 4, 8);
    t2.Duration = 4;
    t2.PercentComplete = 0.6;

    Task t3 = new Task();
    t3.Mode = TaskMode.Manual;
    t3.Name = "Development";
    t3.Start = new DateTime(2016, 5, 1);
    t3.Duration = 10;
    t3.PercentComplete = 0.2;

    Task t4 = new Task();
    t4.Mode = TaskMode.Manual;
    t4.Name = "Quality Assurance";
    t4.Start = new DateTime(2016, 4, 16);
    t4.Duration = 7;
    t4.PercentComplete = 0.2;

    Task t5 = new Task();
    t5.Mode = TaskMode.Manual;
    t5.Name = "Deployment and Release";
    t5.Start = new DateTime(2016, 3, 29);
    t5.Duration = 1;
    t5.PercentComplete = 0.5;

    Task t6 = new Task();
    t6.Mode = TaskMode.Manual;
    t6.Name = "Maintenance";
    t6.Start = new DateTime(2016, 5, 3);
    t6.Duration = 4;
    t6.PercentComplete = 1;
```

```
            Task t7 = new Task();
            t7.Mode = TaskMode.Manual;
            t7.Name = "Enhancements";
            t7.Start = new DateTime(2016, 5, 11);
            t7.Duration = 2;
            t7.PercentComplete = 0.8;

            //Add the tasks to the GanttView
            gv.Tasks.Add(t);
            gv.Tasks.Add(t1);
            gv.Tasks.Add(t2);
            gv.Tasks.Add(t3);
            gv.Tasks.Add(t4);
            gv.Tasks.Add(t5);
            gv.Tasks.Add(t6);
            gv.Tasks.Add(t7);
        }
```

Step 3: Running the application

Press **F5** to run the application and observe how task attributes get displayed in the control.

Data Binding

Data binding refers to the process of binding a data provider to a data consumer in a synchronized manner. You can bind GanttView control to any .NET data source requiring little code and display your project related attributes. GanttView supports data binding with various ADO.NET objects such as DataTable, DataView and DataSet.

The instructions given below explain step-by-step procedure for binding the GanttView control to a data source. This example uses a sample database file, **Nwind.mdb**, as the data source for binding.

Binding to a Data Source

Complete the steps given below to bind the GanttView control to a data source.

1. Create a WPF application (by the name DataBinding) and place C1GanttView control onto the MainWindow.
2. Set the Name property of the control to 'gv' in the XAML markup.

   XAML
   ```
   <c1:C1GanttView x:Name="gv" Margin="0,37,0,0">
   ```
3. Click **View** | **Other Windows** | **Data Sources** to add a data source to your project.
4. In **Data Sources** window, click **Add New Data Source** option to open the **Data Configuration Wizard** window.
5. In the **Data Configuration Wizard** window, select **Database** as the Data Source Type and click **Next**.
6. Select **Dataset** as the Database Model and click **Next**.
7. Click **New Connection** to connect to a database file. The **Add Connection** dialog appears.
8. In the **Add Connection** dialog, browse to the database file with which you want to set the connection.

   The **Nwind.mdb** database file is by default kept in the installed folder at the following location.

   C:\Users\...\Documents\ComponentOne Samples\Common\Nwind.mdb
9. Click **Test Connection** to test that the connection is successful and click **OK**.
10. Click **Next** and then **OK** to copy the database file in your project.
11. In the **Tables** drop-down list, select **Calendars**, **Properties**, **Resources** and **Tasks** tables as the object models and click **Finish**.

Storing and Retrieving Data

Complete the following steps to store data in your dataset and retrieve the same to display various project related attributes in GanttView.

1. Add the following import statements.

   Visual Basic
   ```
   Imports C1.GanttView
   Imports DataBinding.C1NWindDataSetTableAdapters
   ```
   C#
   ```
   using C1.GanttView;
   using DataBinding.C1NWindDataSetTableAdapters;
   ```
2. Create the objects of the NwindDataSet and data adapters for Calendars, Properties, Resources and Tasks tables.

   Visual Basic
   ```
   Private c1NwindDataSet1 As New C1NWindDataSet()
   Private tasksTableAdapter As New TasksTableAdapter()
   Private calendarsTableAdapter As New CalendarsTableAdapter()
   Private resourcesTableAdapter As New ResourcesTableAdapter()
   Private propertiesTableAdapter As New PropertiesTableAdapter()
   ```
   C#

```
private C1NWindDataSet c1NwindDataSet1 = new C1NWindDataSet();
private TasksTableAdapter tasksTableAdapter = new TasksTableAdapter();
private CalendarsTableAdapter calendarsTableAdapter = new CalendarsTableAdapter();
private ResourcesTableAdapter resourcesTableAdapter = new ResourcesTableAdapter();
private PropertiesTableAdapter propertiesTableAdapter = new
PropertiesTableAdapter();
```

3. Fill the tables using the data adapters and set different storage mappings for C1GanttViewStorage using the given code.

   Visual Basic

```
Me.tasksTableAdapter.Fill(c1NwindDataSet1.Tasks)
Me.resourcesTableAdapter.Fill(c1NwindDataSet1.Resources)
Me.propertiesTableAdapter.Fill(c1NwindDataSet1.Properties)
Me.calendarsTableAdapter.Fill(c1NwindDataSet1.Calendars)
```

   C#

```
this.tasksTableAdapter.Fill(c1NwindDataSet1.Tasks);
this.resourcesTableAdapter.Fill(c1NwindDataSet1.Resources);
this.propertiesTableAdapter.Fill(c1NwindDataSet1.Properties);
this.calendarsTableAdapter.Fill(c1NwindDataSet1.Calendars);
```

4. Subscribe the Loaded event of the C1GanttView class.

   Visual Basic

```
AddHandler gv.Loaded, AddressOf gv_Loaded
```

   C#

```
gv.Loaded+=gv_Loaded;
```

5. Add the following code to set different storage mappings for C1GanttViewStorage class in the Loaded event.

   Visual Basic

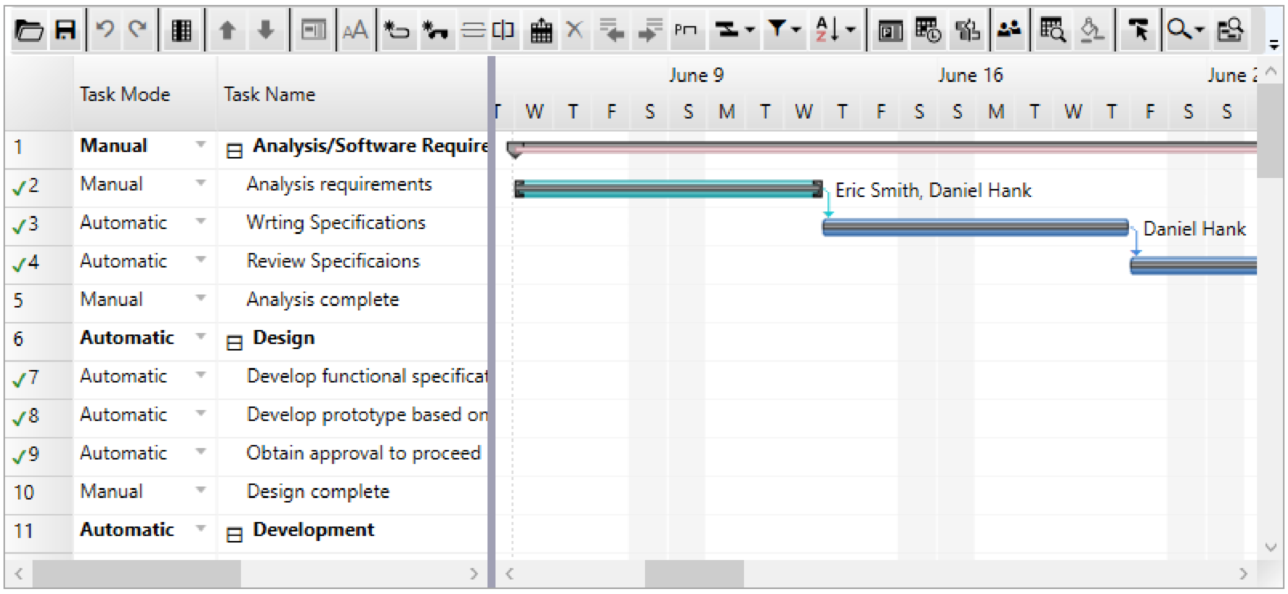
```
Private Sub gv_Loaded(sender As Object, e As RoutedEventArgs)

    Dim storage As C1GanttViewStorage = gv.DataStorage

    storage.CalendarStorage.Mappings.CalendarID.MappingName = "CalendarID"
    storage.CalendarStorage.Mappings.Data.MappingName = "Data"
    storage.CalendarStorage.Mappings.IdMapping.MappingName = "Id"
    storage.CalendarStorage.Mappings.Name.MappingName = "Name"
    storage.CalendarStorage.DataMember = "Calendars"
    storage.CalendarStorage.DataSource = Me.c1NwindDataSet1

    storage.PropertyStorage.Key.MappingName = "Key"
    storage.PropertyStorage.Value.MappingName = "Value"
    storage.PropertyStorage.DataMember = "Properties"
    storage.PropertyStorage.DataSource = Me.c1NwindDataSet1

    storage.ResourceStorage.Mappings.Cost.MappingName = "Cost"
    storage.ResourceStorage.Mappings.IdMapping.MappingName = "Id"
    storage.ResourceStorage.Mappings.Name.MappingName = "Name"
    storage.ResourceStorage.Mappings.Notes.MappingName = "Notes"
    storage.ResourceStorage.Mappings.ResourceID.MappingName = "ResourceID"
    storage.ResourceStorage.Mappings.ResourceType.MappingName = "ResourceType"
    storage.ResourceStorage.Mappings.UnitOfMeasure.MappingName = "UnitOfMeasure"
    storage.ResourceStorage.DataMember = "Resources"
    storage.ResourceStorage.DataSource = Me.c1NwindDataSet1

    storage.TasksStorage.Mappings.CalendarID.MappingName = "CalendarID"
    storage.TasksStorage.Mappings.ConstraintDate.MappingName = "ConstraintDate"
    storage.TasksStorage.Mappings.ConstraintType.MappingName = "ConstraintType"
    storage.TasksStorage.Mappings.CustomFields.MappingName = "CustomFields"
    storage.TasksStorage.Mappings.Deadline.MappingName = "Deadline"
    storage.TasksStorage.Mappings.Duration.MappingName = "Duration"
    storage.TasksStorage.Mappings.DurationUnits.MappingName = "DurationUnits"
    storage.TasksStorage.Mappings.Finish.MappingName = "Finish"
    storage.TasksStorage.Mappings.HideBar.MappingName = "HideBar"
    storage.TasksStorage.Mappings.IdMapping.MappingName = "Id"
    storage.TasksStorage.Mappings.Initialized.MappingName = "Initialized"
    storage.TasksStorage.Mappings.Mode.MappingName = "Mode"
    storage.TasksStorage.Mappings.Name.MappingName = "Name"
    storage.TasksStorage.Mappings.NextID.MappingName = "NextID"
    storage.TasksStorage.Mappings.Notes.MappingName = "Notes"
    storage.TasksStorage.Mappings.Parts.MappingName = "Parts"
    storage.TasksStorage.Mappings.PercentComplete.MappingName = "PercentComplete"
    storage.TasksStorage.Mappings.Predecessors.MappingName = "Predecessors"
    storage.TasksStorage.Mappings.Resources.MappingName = "Resources"
    storage.TasksStorage.Mappings.Start.MappingName = "Start"
    storage.TasksStorage.Mappings.TaskID.MappingName = "TaskID"
    storage.TasksStorage.DataMember = "Tasks"
    storage.TasksStorage.DataSource = Me.c1NwindDataSet1

End Sub
```

C#

```
private void gv_Loaded(object sender, RoutedEventArgs e)
{
     C1GanttViewStorage storage = gv.DataStorage;
    storage.CalendarStorage.Mappings.CalendarID.MappingName = "CalendarID";
    storage.CalendarStorage.Mappings.CalendarID.MappingName = "CalendarID";
    storage.CalendarStorage.Mappings.Data.MappingName = "Data";
    storage.CalendarStorage.Mappings.IdMapping.MappingName = "Id";
    storage.CalendarStorage.Mappings.Name.MappingName = "Name";
    storage.CalendarStorage.DataMember = "Calendars";
    storage.CalendarStorage.DataSource = this.c1NwindDataSet1;

    storage.PropertyStorage.Key.MappingName = "Key";
    storage.PropertyStorage.Value.MappingName = "Value";
    storage.PropertyStorage.DataMember = "Properties";
    storage.PropertyStorage.DataSource = this.c1NwindDataSet1;

    storage.ResourceStorage.Mappings.Cost.MappingName = "Cost";
    storage.ResourceStorage.Mappings.IdMapping.MappingName = "Id";
    storage.ResourceStorage.Mappings.Name.MappingName = "Name";
    storage.ResourceStorage.Mappings.Notes.MappingName = "Notes";
    storage.ResourceStorage.Mappings.ResourceID.MappingName = "ResourceID";
    storage.ResourceStorage.Mappings.ResourceType.MappingName = "ResourceType";
    storage.ResourceStorage.Mappings.UnitOfMeasure.MappingName = "UnitOfMeasure";
    storage.ResourceStorage.DataMember = "Resources";
    storage.ResourceStorage.DataSource = this.c1NwindDataSet1;

    storage.TasksStorage.Mappings.CalendarID.MappingName = "CalendarID";
    storage.TasksStorage.Mappings.ConstraintDate.MappingName = "ConstraintDate";
    storage.TasksStorage.Mappings.ConstraintType.MappingName = "ConstraintType";
    storage.TasksStorage.Mappings.CustomFields.MappingName = "CustomFields";
    storage.TasksStorage.Mappings.Deadline.MappingName = "Deadline";
    storage.TasksStorage.Mappings.Duration.MappingName = "Duration";
    storage.TasksStorage.Mappings.DurationUnits.MappingName = "DurationUnits";
    storage.TasksStorage.Mappings.Finish.MappingName = "Finish";
    storage.TasksStorage.Mappings.HideBar.MappingName = "HideBar";
    storage.TasksStorage.Mappings.IdMapping.MappingName = "Id";
    storage.TasksStorage.Mappings.Initialized.MappingName = "Initialized";
    storage.TasksStorage.Mappings.Mode.MappingName = "Mode";
    storage.TasksStorage.Mappings.Name.MappingName = "Name";
    storage.TasksStorage.Mappings.NextID.MappingName = "NextID";
```

```
        storage.TasksStorage.Mappings.Notes.MappingName = "Notes";
        storage.TasksStorage.Mappings.Parts.MappingName = "Parts";
        storage.TasksStorage.Mappings.PercentComplete.MappingName = "PercentComplete";
        storage.TasksStorage.Mappings.Predecessors.MappingName = "Predecessors";
        storage.TasksStorage.Mappings.Resources.MappingName = "Resources";
        storage.TasksStorage.Mappings.Start.MappingName = "Start";
        storage.TasksStorage.Mappings.TaskID.MappingName = "TaskID";
        storage.TasksStorage.DataMember = "Tasks";
        storage.TasksStorage.DataSource = this.c1NwindDataSet1;
    }
```

6. Run the application and notice that the GanttView gets bound to the tables in the NwindDataSet.



	Task Mode	Task Name
1	**Manual**	**Analysis/Software Require**
✓2	Manual	Analysis requirements
✓3	Automatic	Wrting Specifications
✓4	Automatic	Review Specificaions
5	Manual	Analysis complete
6	**Automatic**	**Design**
✓7	Automatic	Develop functional specificat
✓8	Automatic	Develop prototype based on
✓9	Automatic	Obtain approval to proceed
10	Manual	Design complete
11	**Automatic**	**Development**

Design-Time Support

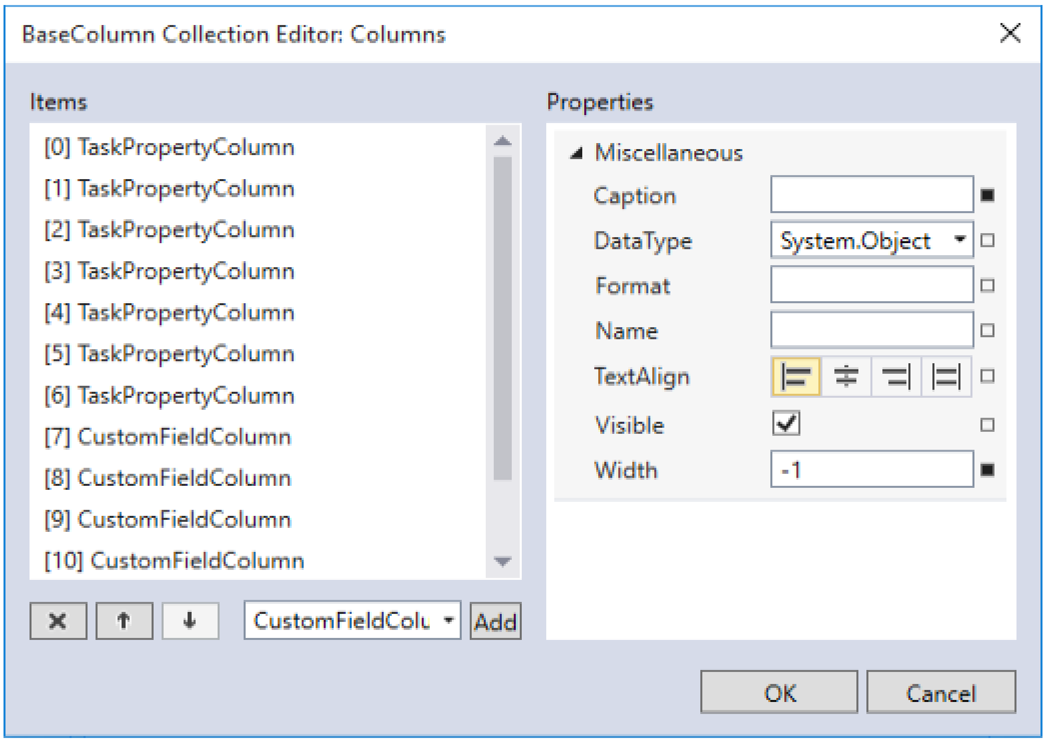
The GanttView control provides various collection editors to apply properties to various GanttView elements at design time. To know about these editors, click the links below:

- BaseColumn collection editor
- CustomCalendars collection editor
- CalendarException collection editor
- Task collection editor
- WorkWeek collection editor
- BarStyle collection editor
- FieldStyle collection editor
- ResourceRef collection editor

BaseColumn Collection Editor

The BaseColumn Collection Editor allows users to add TaskPropertyColumns and CustomFieldColumns, and modifying properties at design-time. To access BaseColumn Collection Editor, select the GanttView control placed on the designer, that is MainWindow, navigate to the Properties window and click the ellipsis button next to the Columns property.

The given image shows the BaseColumn Collection Editor:


BaseColumn Collection Editor: Columns

Items

[0] TaskPropertyColumn
[1] TaskPropertyColumn
[2] TaskPropertyColumn
[3] TaskPropertyColumn
[4] TaskPropertyColumn
[5] TaskPropertyColumn
[6] TaskPropertyColumn
[7] CustomFieldColumn
[8] CustomFieldColumn
[9] CustomFieldColumn
[10] CustomFieldColumn

Properties

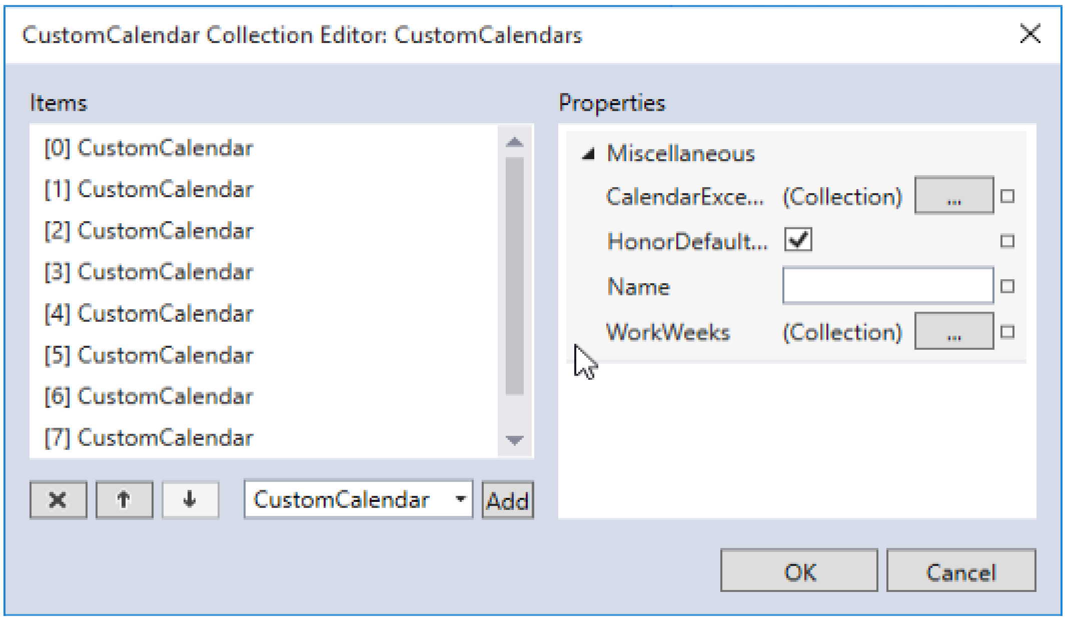
Miscellaneous

Caption	
DataType	System.Object
Format	
Name	
TextAlign	
Visible	✓
Width	-1

CustomCalendar Collection Editor

The CustomCalendar Collection Editor is used to add, modify or remove custom calendars at design-time. To access CustomCalendar Collection Editor, select the GanttView control placed on the designer, that is MainWindow, navigate to the Properties window and click the ellipsis button next to CustomCalendarCollection property.

The given image shows the CustomCalendar Collection Editor:


CustomCalendar Collection Editor: CustomCalendars

Items

[0] CustomCalendar
[1] CustomCalendar
[2] CustomCalendar
[3] CustomCalendar
[4] CustomCalendar
[5] CustomCalendar
[6] CustomCalendar
[7] CustomCalendar

Properties

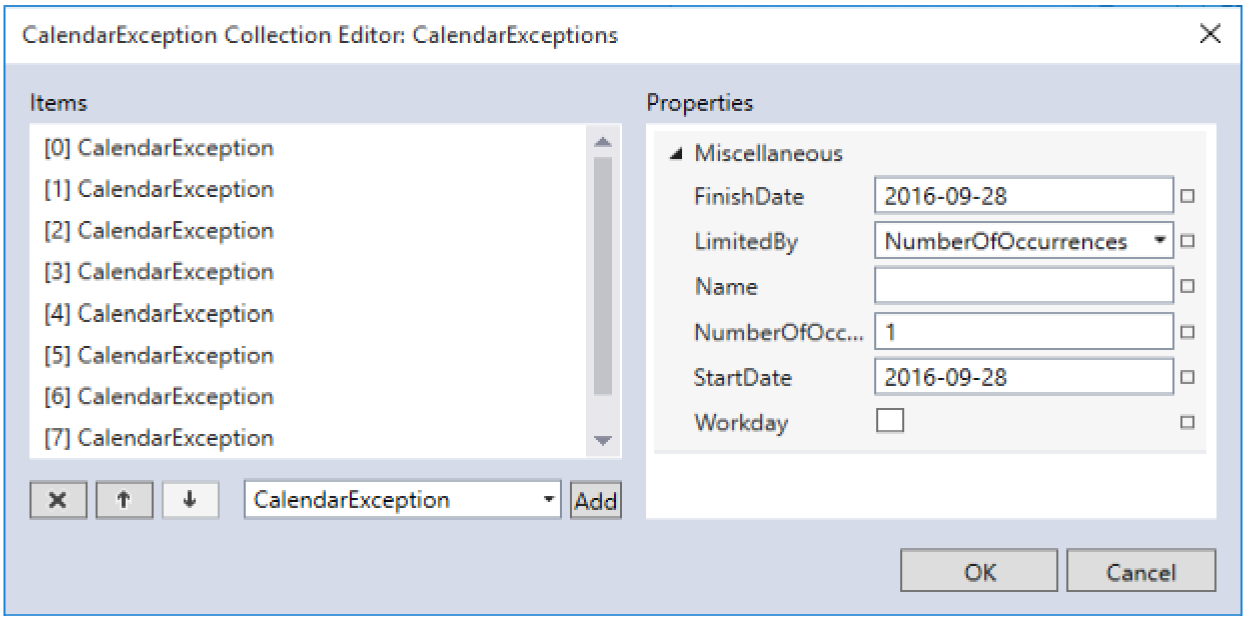
Miscellaneous

CalendarExce...	(Collection)
HonorDefault...	✓
Name	
WorkWeeks	(Collection)

CalendarException Collection Editor

The CalendarException Collection Editor is used for adding, removing, or modifying exceptions in custom calendars at design-time. To access CalendarException Collection Editor, select the GanttView control placed on the designer, that is MainWindow, navigate to the Properties windows and click the ellipsis button next to CalendarExceptionCollection property.

The given image shows the CalendarException Collection Editor:


CalendarException Collection Editor: CalendarExceptions

Items

[0] CalendarException
[1] CalendarException
[2] CalendarException
[3] CalendarException
[4] CalendarException
[5] CalendarException
[6] CalendarException
[7] CalendarException

Properties

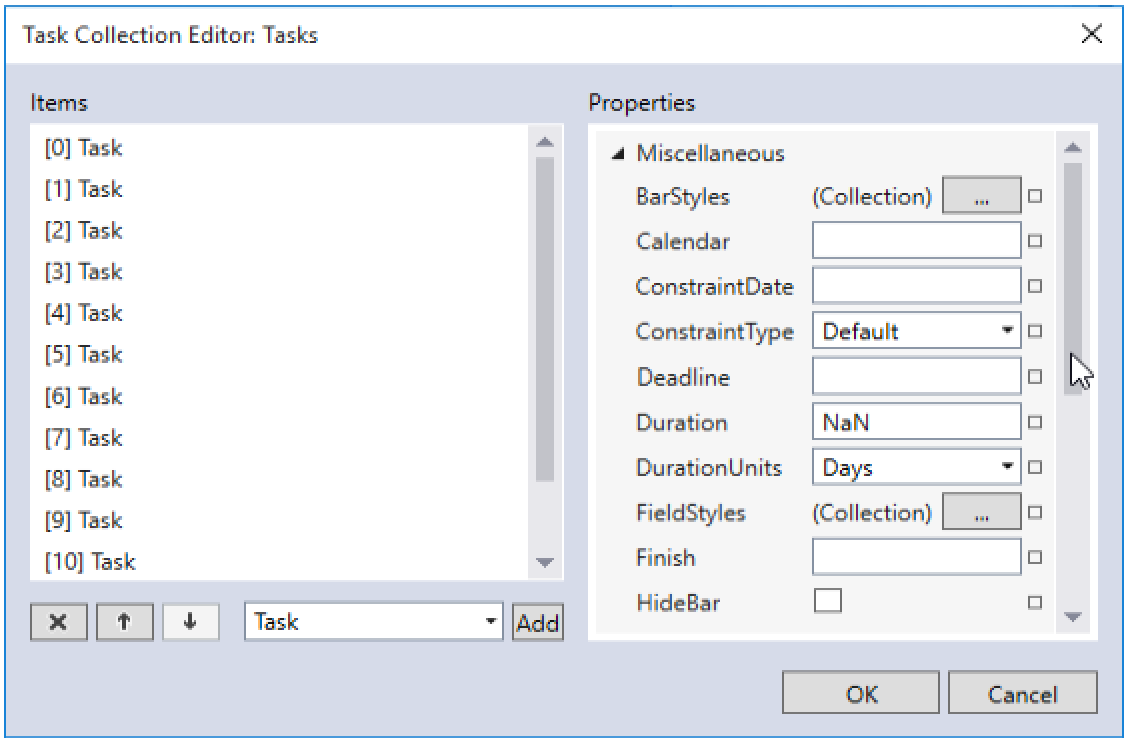
Miscellaneous

FinishDate	2016-09-28
LimitedBy	NumberOfOccurrences
Name	
NumberOfOcc...	1
StartDate	2016-09-28
Workday	☐

Task Collection Editor

The Task Collection Editor is used for adding, modifying and removing tasks from GanttView at design-time. To access Task Collection Editor, select the GanttView control placed on the designer, that is MainWindow, navigate to the Properties window and click the ellipsis button next to the Tasks property.

The given image shows the Task Collection Editor:


Task Collection Editor: Tasks

Items

[0] Task
[1] Task
[2] Task
[3] Task
[4] Task
[5] Task
[6] Task
[7] Task
[8] Task
[9] Task
[10] Task

Task | Add

Properties

◢ Miscellaneous

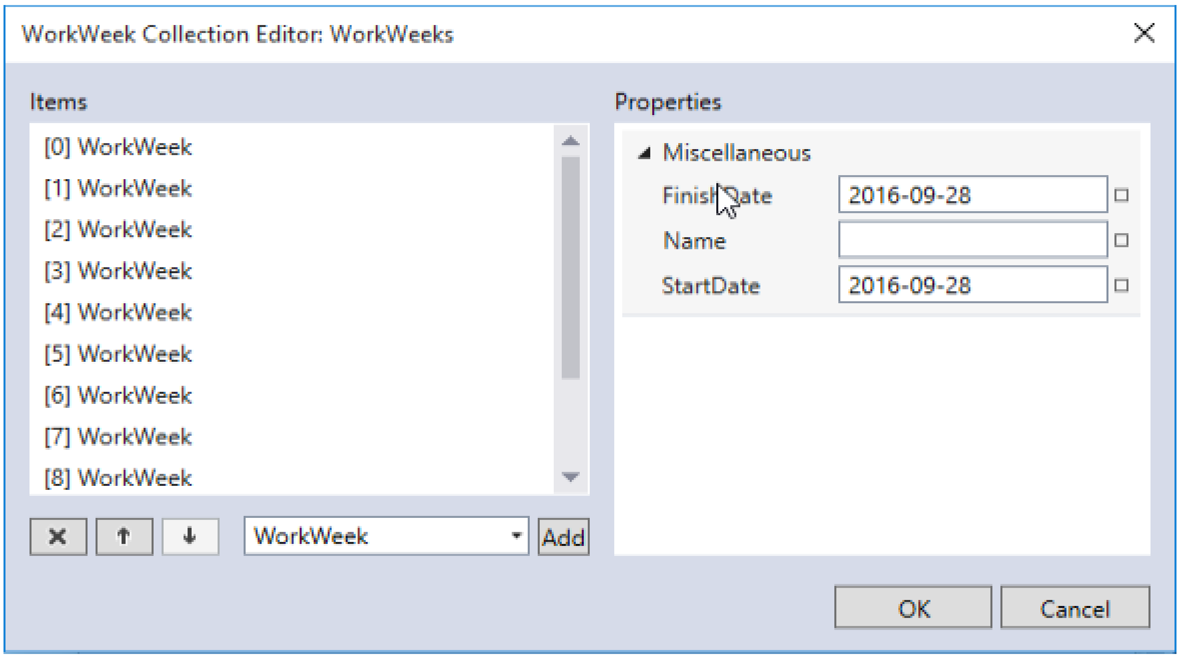
BarStyles	(Collection)
Calendar	
ConstraintDate	
ConstraintType	Default
Deadline	
Duration	NaN
DurationUnits	Days
FieldStyles	(Collection)
Finish	
HideBar	☐

OK | Cancel

WorkWeek Collection Editor

The WorkWeek Collection Editor is used for adding, modifying and removing work weeks in custom calendars at design-time. The WorkWeek Collection Editor can be accessed from the CustomCalendar Collection Editor.

The given image shows the WorkWeek Collection Editor:


WorkWeek Collection Editor: WorkWeeks

Items

[0] WorkWeek
[1] WorkWeek
[2] WorkWeek
[3] WorkWeek
[4] WorkWeek
[5] WorkWeek
[6] WorkWeek
[7] WorkWeek
[8] WorkWeek

WorkWeek | Add

Properties

◢ Miscellaneous

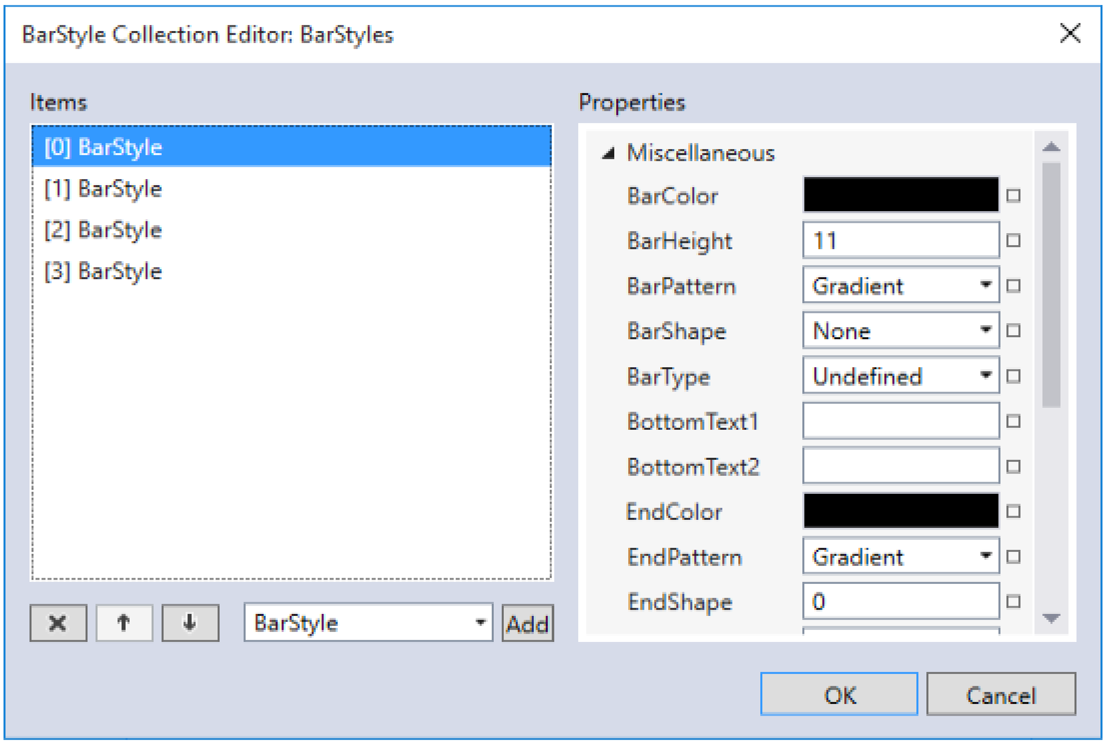
FinishDate	2016-09-28
Name	
StartDate	2016-09-28

OK | Cancel

BarStyle Collection Editor

The BarStyle Collection Editor allows users to add bar styles, and modify its properties at design time. To access BarStyle Collection Editor, select the GanttView control placed on the MainWindow, navigate to the Properties window and click the ellipsis button next to the BarStyles property.

The given image shows the BarStyle Collection Editor:



[0] BarStyle
[1] BarStyle
[2] BarStyle
[3] BarStyle

BarStyle

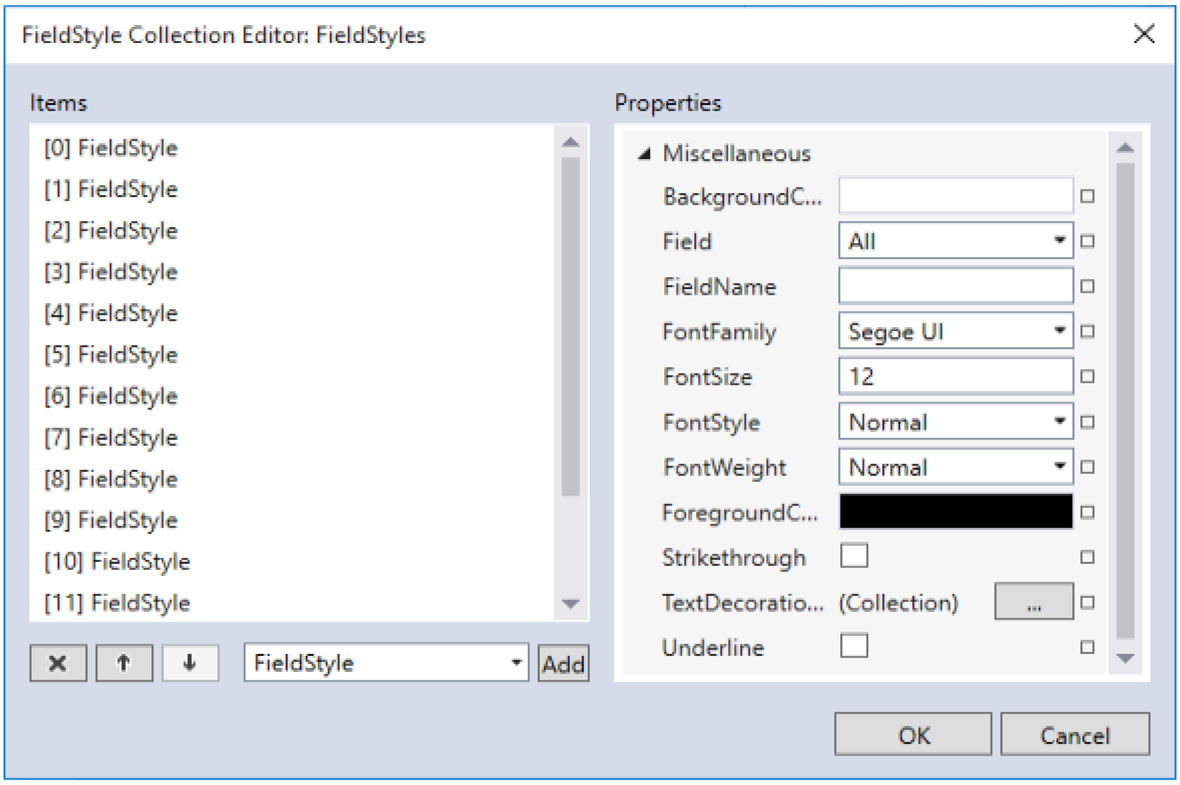
Miscellaneous

BarColor	■
BarHeight	11
BarPattern	Gradient
BarShape	None
BarType	Undefined
BottomText1	
BottomText2	
EndColor	■
EndPattern	Gradient
EndShape	0

FieldStyle Collection Editor

The FieldStyle Collection Editor allows users to add FieldStyles, and modify its properties at design time. The FieldStyle Collection Editor can be accessed from the Task Collection Editor.

The following image shows the FieldStyle Collection Editor:


FieldStyle Collection Editor: FieldStyles

Items

[0] FieldStyle
[1] FieldStyle
[2] FieldStyle
[3] FieldStyle
[4] FieldStyle
[5] FieldStyle
[6] FieldStyle
[7] FieldStyle
[8] FieldStyle
[9] FieldStyle
[10] FieldStyle
[11] FieldStyle

Properties

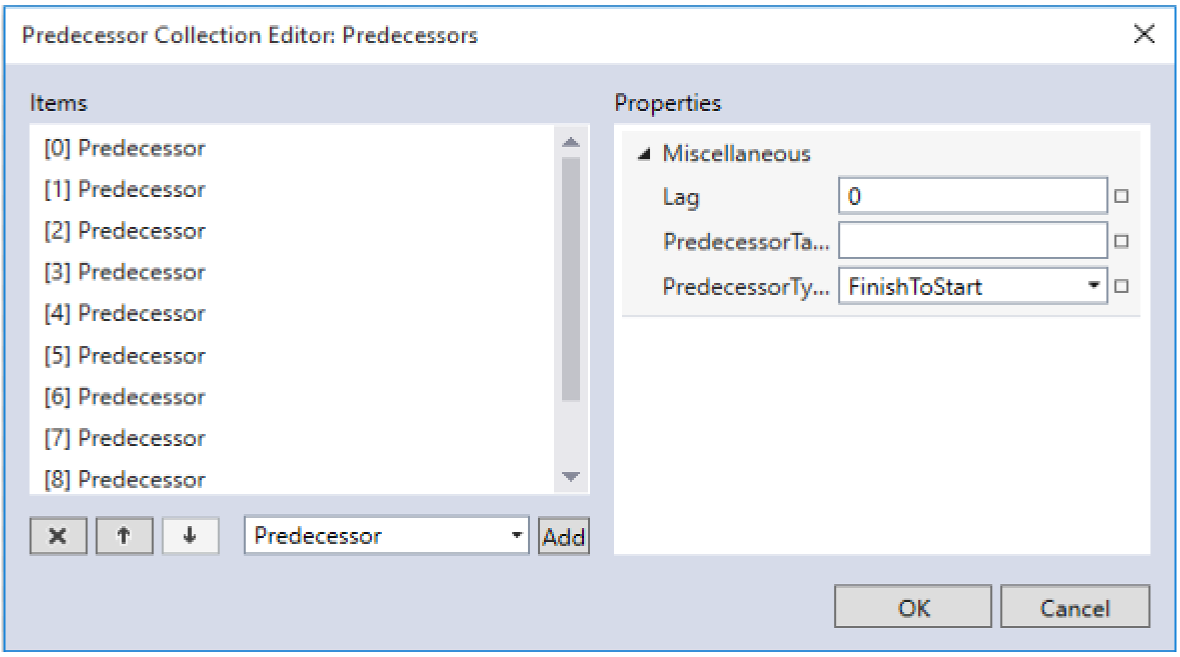
Miscellaneous

BackgroundC...	
Field	All
FieldName	
FontFamily	Segoe UI
FontSize	12
FontStyle	Normal
FontWeight	Normal
ForegroundC...	■
Strikethrough	☐
TextDecoratio...	(Collection)
Underline	☐

Predecessor Collection Editor

The Predecessor Collection Editor allows users to add a predecessor task(s), and modify its properties at design time. The Predecessor Collection Editor can be accessed from the Task Collection Editor.

The following image shows the Predecessor Collection Editor:


Predecessor Collection Editor: Predecessors

Items

[0] Predecessor
[1] Predecessor
[2] Predecessor
[3] Predecessor
[4] Predecessor
[5] Predecessor
[6] Predecessor
[7] Predecessor
[8] Predecessor

Properties

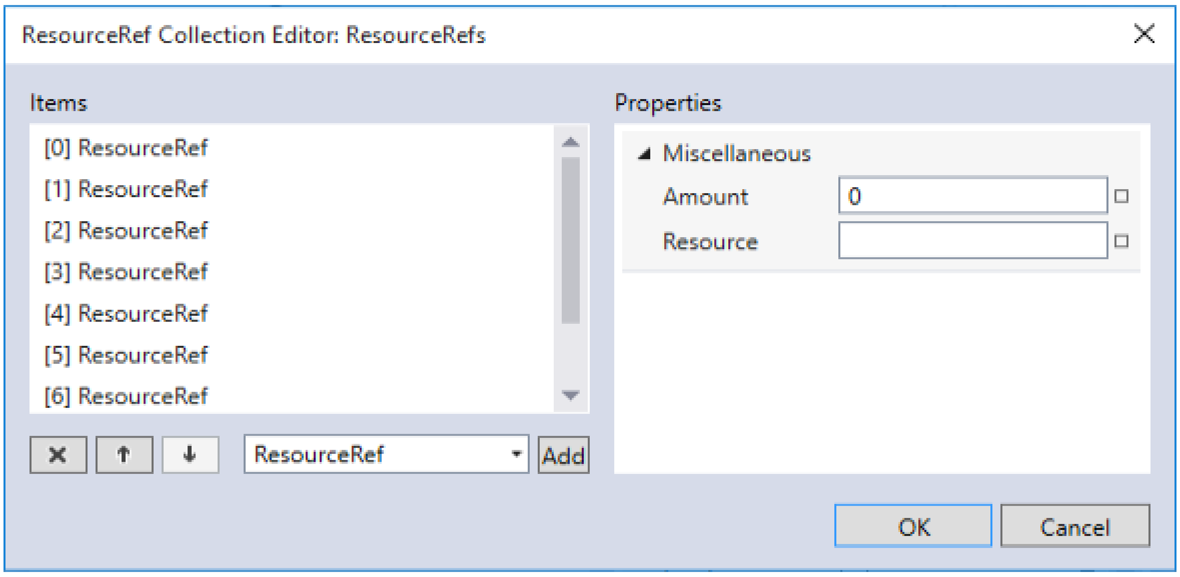
Miscellaneous

Lag	0
PredecessorTa...	
PredecessorTy...	FinishToStart

ResourceRef Collection Editor

The ResourceRef Collection Editor is used to add various resources to a particular task. The ResourceRef Collection Editor can be accessed from the Task Collection Editor.

The following image shows the ResourceRef Collection Editor:



Runtime Interaction

GanttView comes with various runtime features to help users seamlessly interact with the control. You can create a new task, delete a task, configure grid columns, view and modify project schedule, etc. without writing even a single line of code.

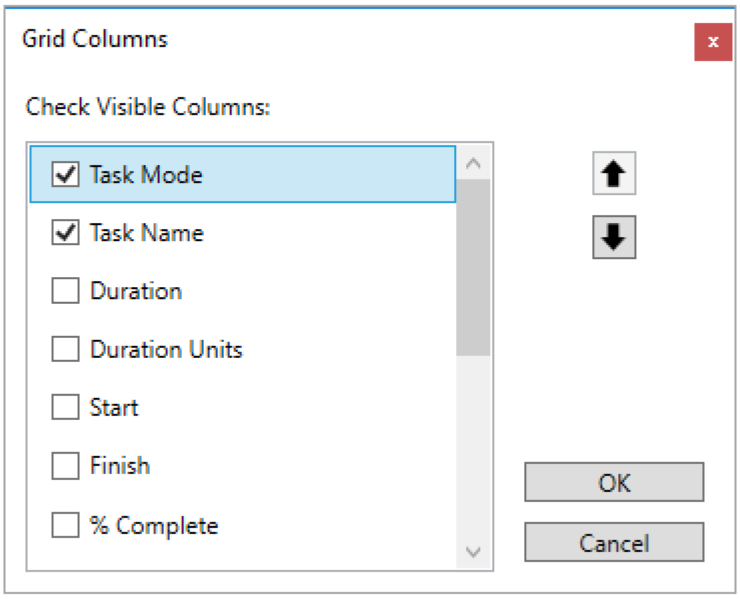
GanttView provides the following dialogs at runtime to access various features.

- Grid columns dialog
- Task information dialog
- Field styles dialog
- Group dialog
- DateRange dialog
- Using resource dialog
- Advanced filter dialog
- More filter dialog
- Sort dialog
- Project information dialog
- Change working time dialog
- Progress line dialog
- Resources dialog
- Timescale dialog
- Bar styles dialog
- Zoom dialog
- Print dialog

Grid Columns Dialog

The **Grid Columns** dialog is used to specify various attributes appearing in the columns of GanttView. This dialog appears on clicking the **Grid Columns** button on the toolbar at runtime. You can select the check box corresponding to the columns names to display them on the GanttView.

The following image shows the Grid Columns dialog with default visible columns including Task Mode and Task Name:


Grid Columns

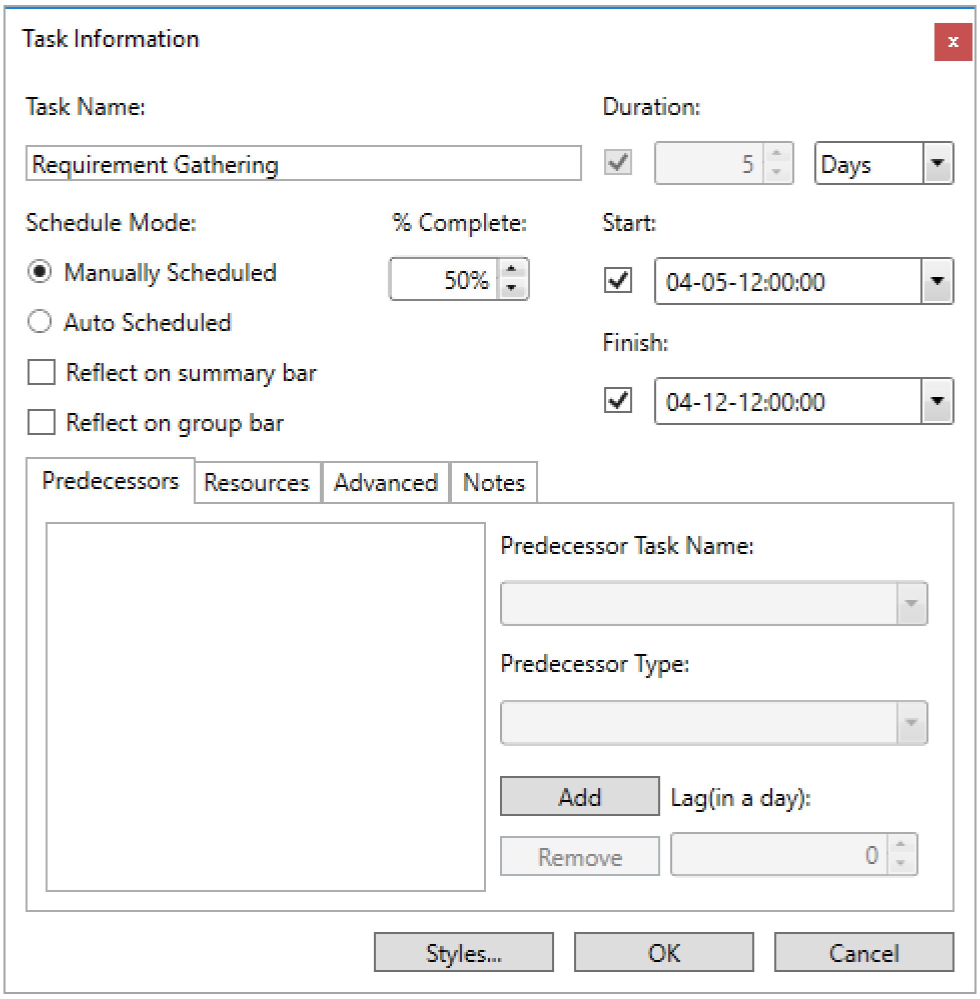
Check Visible Columns:

- ☑ Task Mode
- ☑ Task Name
- ☐ Duration
- ☐ Duration Units
- ☐ Start
- ☐ Finish
- ☐ % Complete

Task Information Dialog

The **Task Information** dialog is used to specify details for a new task such as task name, task mode, start and finish date, etc. This dialog appears on clicking the Task Information button on the toolbar.

The following image shows the **Task Information** dialog:



Task Name: Requirement Gathering

Duration: ✓ 5 Days

Schedule Mode: [✓ Manually Scheduled] Auto Scheduled

% Complete: 50%

Start: ✓ 04-05-12:00:00

Finish: ✓ 04-12-12:00:00

☐ Reflect on summary bar

☐ Reflect on group bar

Lag(in a day): 0

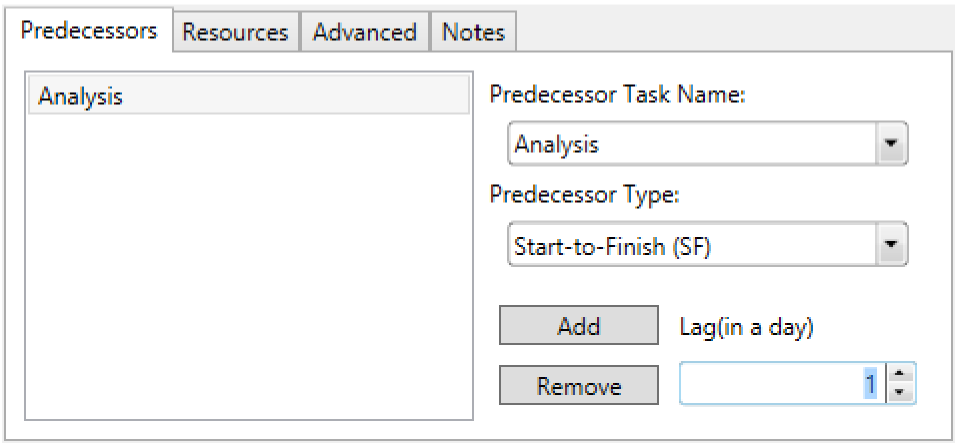
The Task Information dialog provides various options as follows:

Option	**Purpose**
Task Name	Specify the task name in the text box.
% Complete	Specify the task completion in percent in the numeric box.
Schedule Mode	Select the radio buttons to choose the task mode.
Duration	Select the check box to enable entering task duration.
Start	Select the start date from the drop-down list.
Finish	Select the finish date from the drop-down list.

In addition, the Task Information dialog provides four tabs namely **Predecessors**, **Resources**, **Advanced** and **Notes**, each comprising various options for achieving

Predecessors

The following image shows the **Predecessors** tab:



Analysis

Predecessor Task Name: Analysis

Predecessor Type: Start-to-Finish (SF)

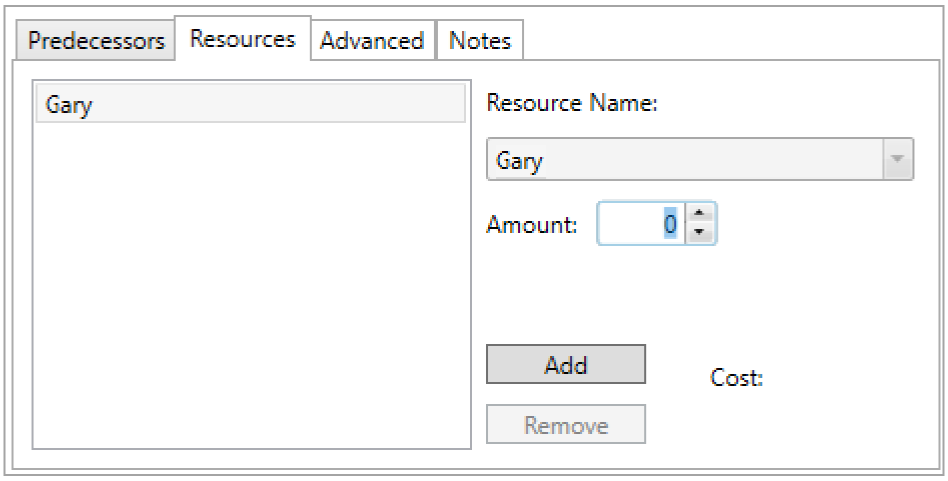
Lag(in a day) 1

The Predecessors tab provides the following options:

Option	**Purpose**
Predecessor Task Name	Specify the predecessor task from the drop-down list.
Predecessor Type	Specify the type of dependence from the drop-down list.
Add	Click the button to add the dependency.
Remove	Click the button to remove the selected dependency.
Lag (in a day)	Specify time delay in days between the dependent tasks in the numeric box.

Resources

The following image shows the **Resources** tab:



Gary

Resource Name: Gary

Amount: 0

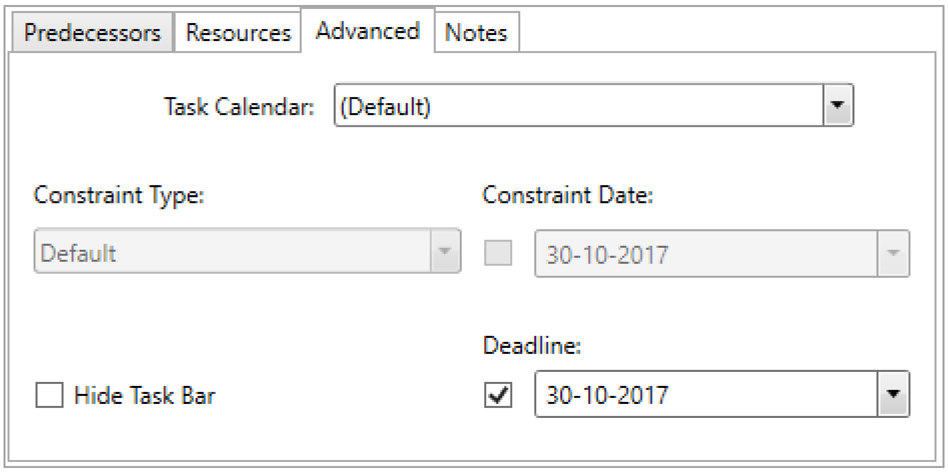
Cost:

The Resources tab provides the following options:

Option	**Purpose**
Resource Name	Specify the resource name from the drop-down list.
Amount	Specify the amount for the selected resource.
Add	Click the button to add the resource.
Remove	Click the button to remove the selected resource.

Advanced

The following image shows the **Advanced** tab:



Task Calendar: (Default)

Constraint Type: Default

Constraint Date: 30-10-2017

Deadline: ✓ 30-10-2017

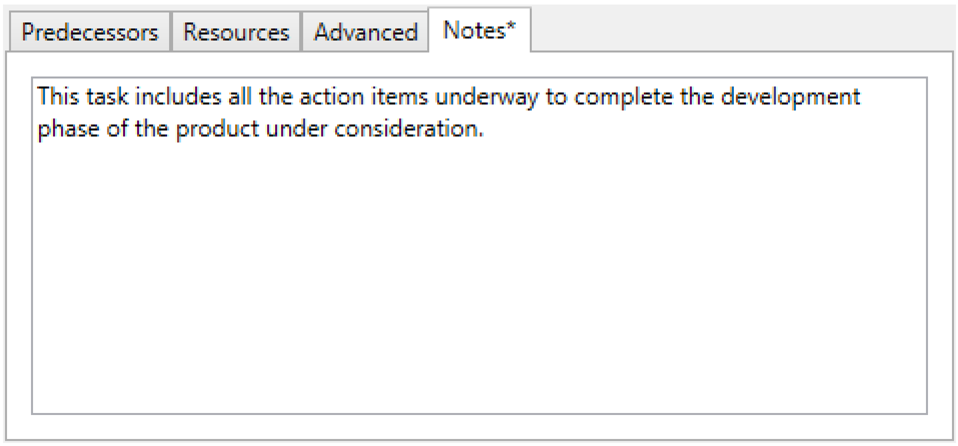
☐ Hide Task Bar

The Advanced tab provides the following options:

Option	Purpose
Task Calendar	Select the calendar from the drop-down list.
Constraint Type	Specify constraint type to be applied from the drop-down list.
Constraint Date	Specify constraint date in the calendar from the drop down.
Deadline	Select the check box to enable the option for specifying deadline date from the calendar in the drop down.
Hide Task Bar	Select the check box to hide the task bar.

Notes

The **Notes** tab provides a text box to enter task related details for reference as follows:

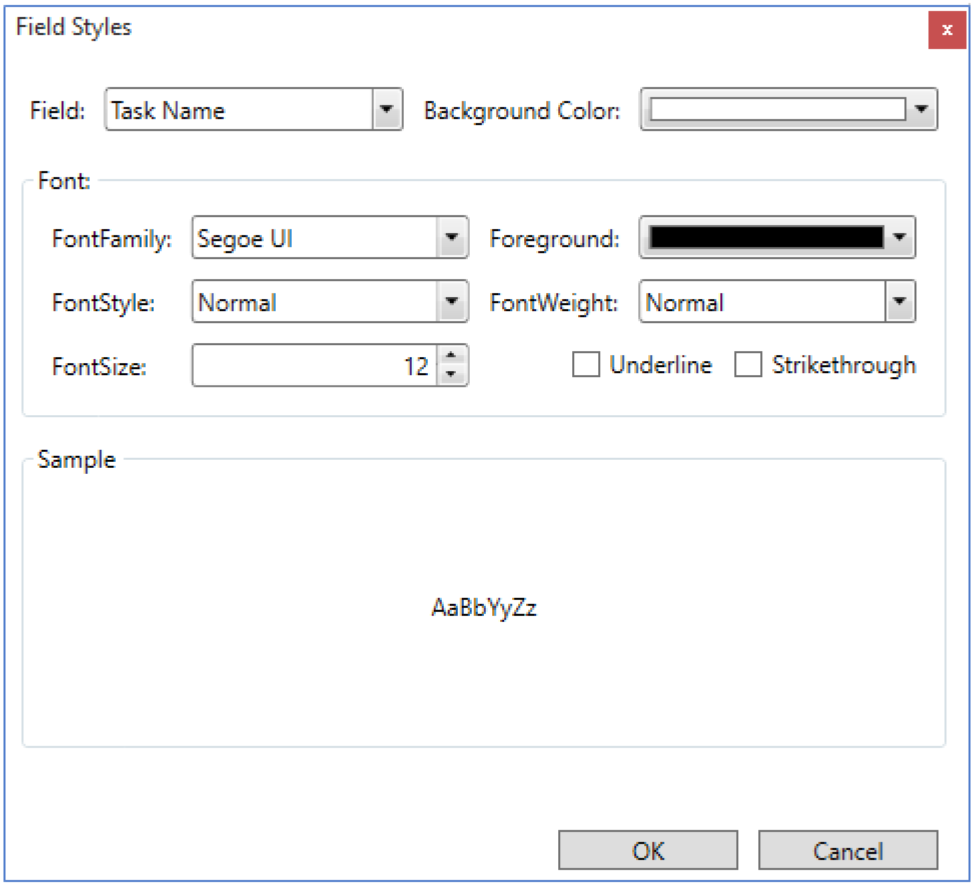


This task includes all the action items underway to complete the development phase of the product under consideration.

Field Styles Dialog

The **Field Styles** dialog is used for customizing text fields that appear in the grid view of the control. The dialog provides options to apply custom fonts, background color and other styling on other text fields. This dialog appears on clicking the **Field Styles** button on the toolbar.

The following image shows the Field Styles dialog:


Field Styles

Field: Task Name

Background Color:

Font:

FontFamily: Segoe UI

Foreground:

FontStyle: Normal

FontWeight: Normal

FontSize: 12

Underline

Strikethrough

Sample

AaBbYyZz

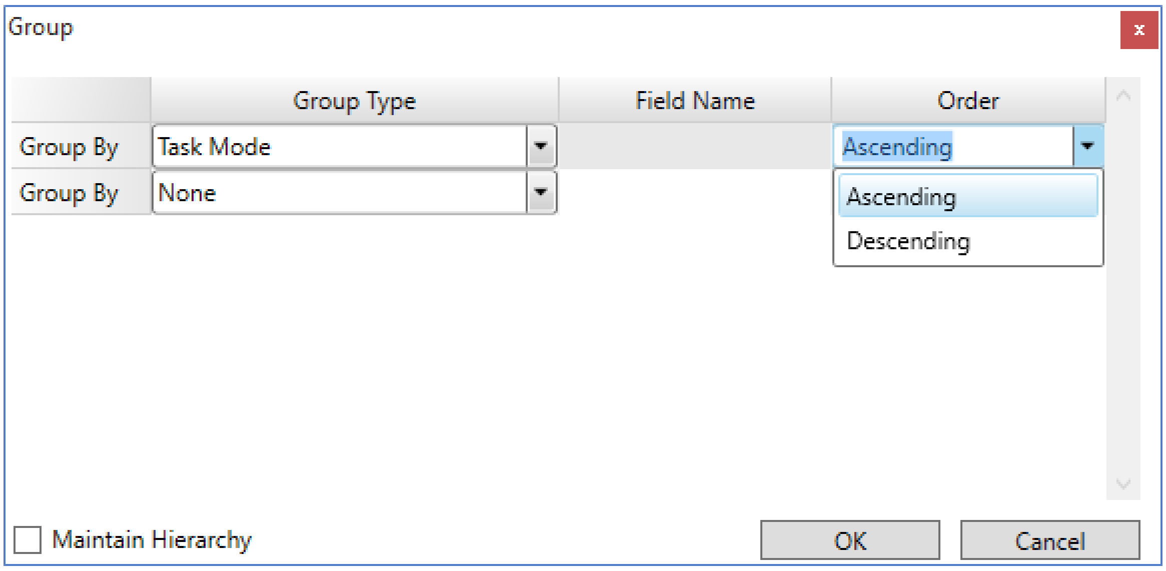
The Field Styles dialog provides various options as follows:

Option	**Purpose**
Field	Select a particular field on which the customization is to be applied from the drop-down list.
Background Color	Select a background color to be applied on the selected field from the drop-down list.
FontFamily	Select the font to be applied on the selected field from the drop-down list.
Foreground	Select the foreground color to be applied on the selected field from the drop-down list.
FontStyle	Select the font style to be applied on the selected field from the drop-down list.
FontWeight	Select the font weight from the drop-down list.
FontSize	Specify the font size in the numeric box.
Underline	Select the checkbox to underline the selected field.
Strikethrough	Select the checkbox to strikethrough the selected field.

Group Dialog

The **Group** dialog is designed to achieve grouping in tasks. The dialog appears on selecting the **New Group By** option under the **Group By** button on the toolbar.

The following image shows the Group dialog:



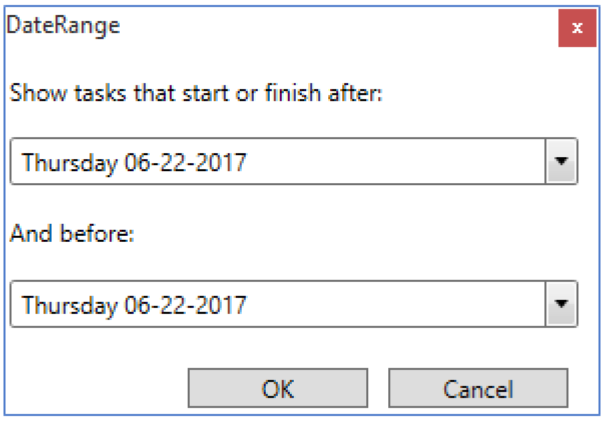
The Group dialog provides various options as follows:

Option	Purpose
Group By	Select a field for grouping from the drop-down list.
Order	Select the order of grouping that is Ascending or Descending from the drop-down list.

DateRange Dialog

The **DateRange** dialog is used to apply custom filters within a specified date range. The dialog appears on selecting the **Date Range** option available within the **Filter** menu on the toolbar.

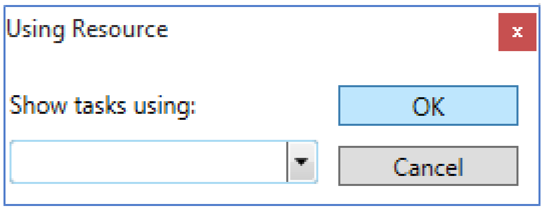
The following image shows the DateRange dialog:



Using Resource Dialog

The **Using Resource** dialog is used to apply custom filters on the basis of resources assigned in the project. The dialog appears on selecting the **Using Resource** option available within the **Filter** menu on the toolbar.

The following image shows the Using Resource dialog:



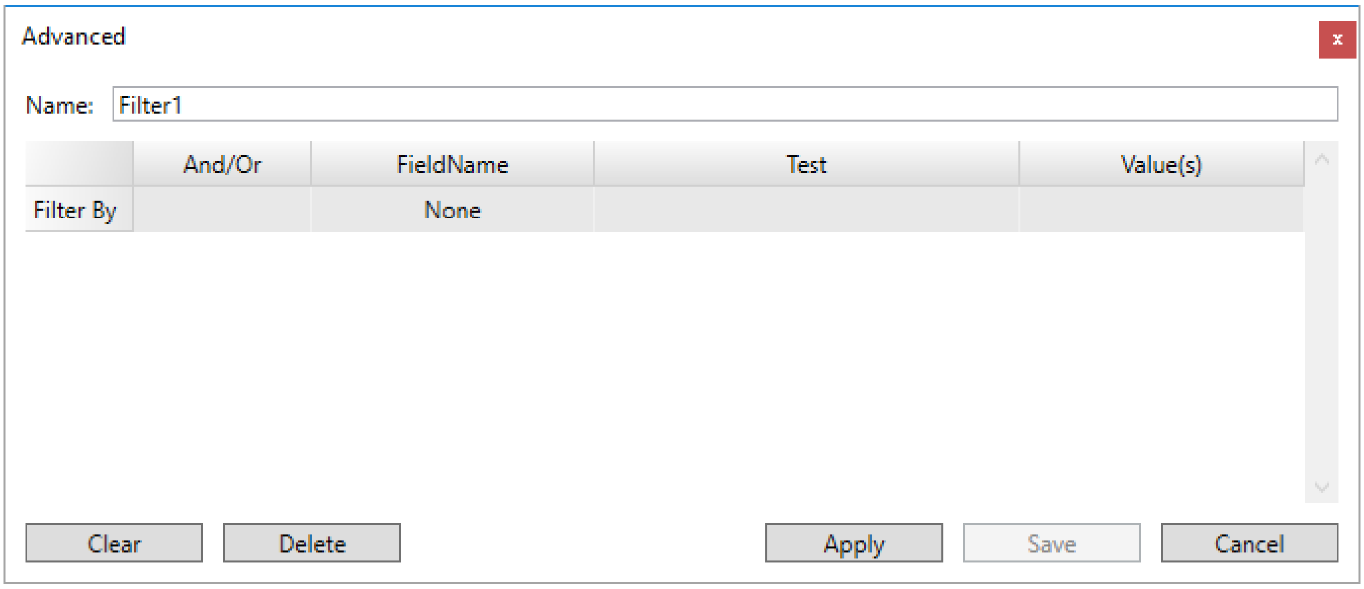
The Using Resource dialog provides options as follows:

Option	**Purpose**
Show tasks using	Shows resources in the drop-down list for filtering.
OK	Applies the filter.
Cancel	Closes the dialog.

Advanced Filter Dialog

The **Advanced Filter** dialog is designed to create and apply custom filters in GanttView. The dialog appears on selecting the **Advanced Filter** option available under the **Filter** button on the toolbar.

The following image shows the Advanced Filter dialog:



Name: Filter1

	And/Or	FieldName	Test	Value(s)
Filter By		None		

The Advanced Filter dialog provides various options as follows:

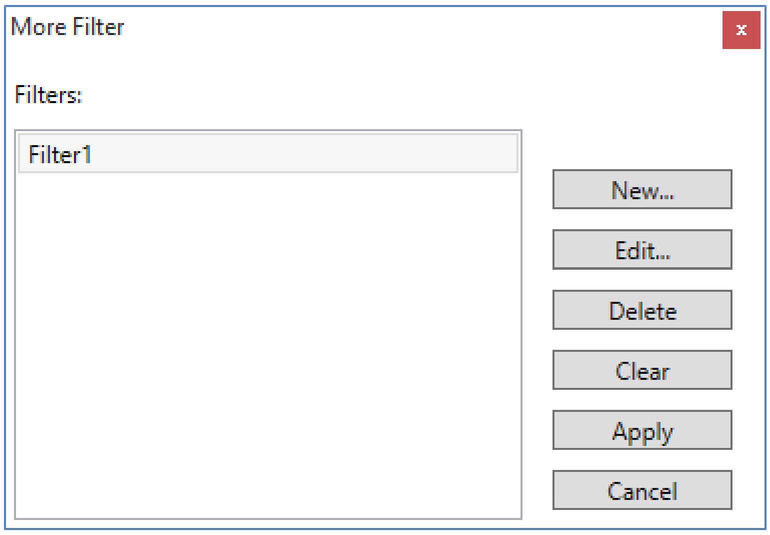
Option	**Purpose**
And/Or	Select 'And' or 'Or' conditional operator from the drop-down list.
FieldName	Select the field for filtering from the drop-down list.
Test	Select the test condition from the drop-down list.
Value(s)	Displays the values corresponding to the field selected in the FieldName option. For

	example, if the FieldName is set to Mode, the Value(s) drop-down menu shows Auto and Manual options for selection.
Clear	Click the button to clear the filter.
Apply	Click the button to apply the filter.
Save	Click the button to save the filter.
Cancel	Click the button to close the dialog.

More Filter Dialog

The **More Filter** dialog is designed to create and apply custom filters in GanttView. This dialog appears on selecting the **More Filters** option available in the **Filter** menu on the toolbar.

The following image shows the More Filter dialog:



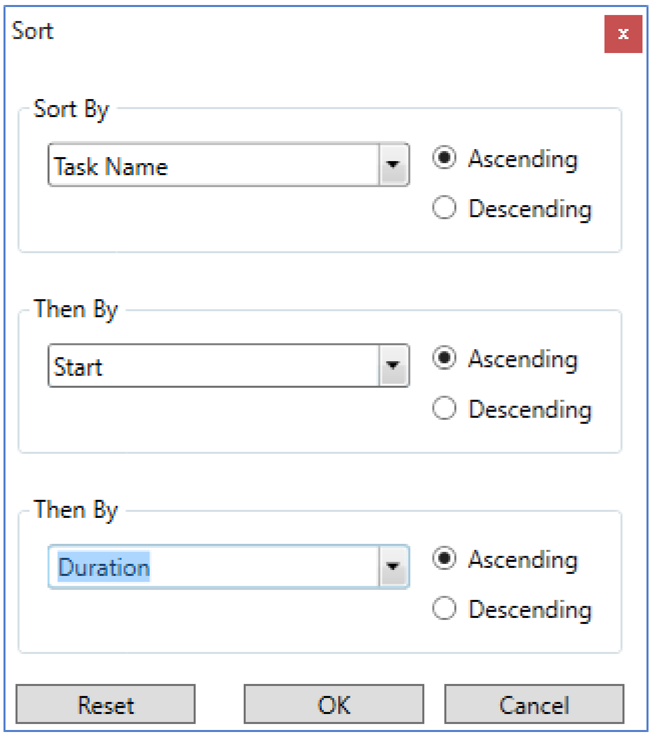
The More Filter dialog provides various options as follows:

Option	Purpose
New	Opens the Advanced Filter dialog that lets you create a custom filter(s).
Edit	Opens the Advanced Filter dialog to edit the selected filter.
Delete	Deletes the selected filter.
Clear	Clears all the filters.
Apply	Applies the available filter(s).
Cancel	Closes the dialog.

Sort Dialog

The **Sort** dialog is designed to achieve multicolumn sorting. This dialog appears on selecting the **Sort By** option available in the Sort menu on the toolbar.

The following image shows the Sort dialog:



The Sort dialog provides various options as follows:

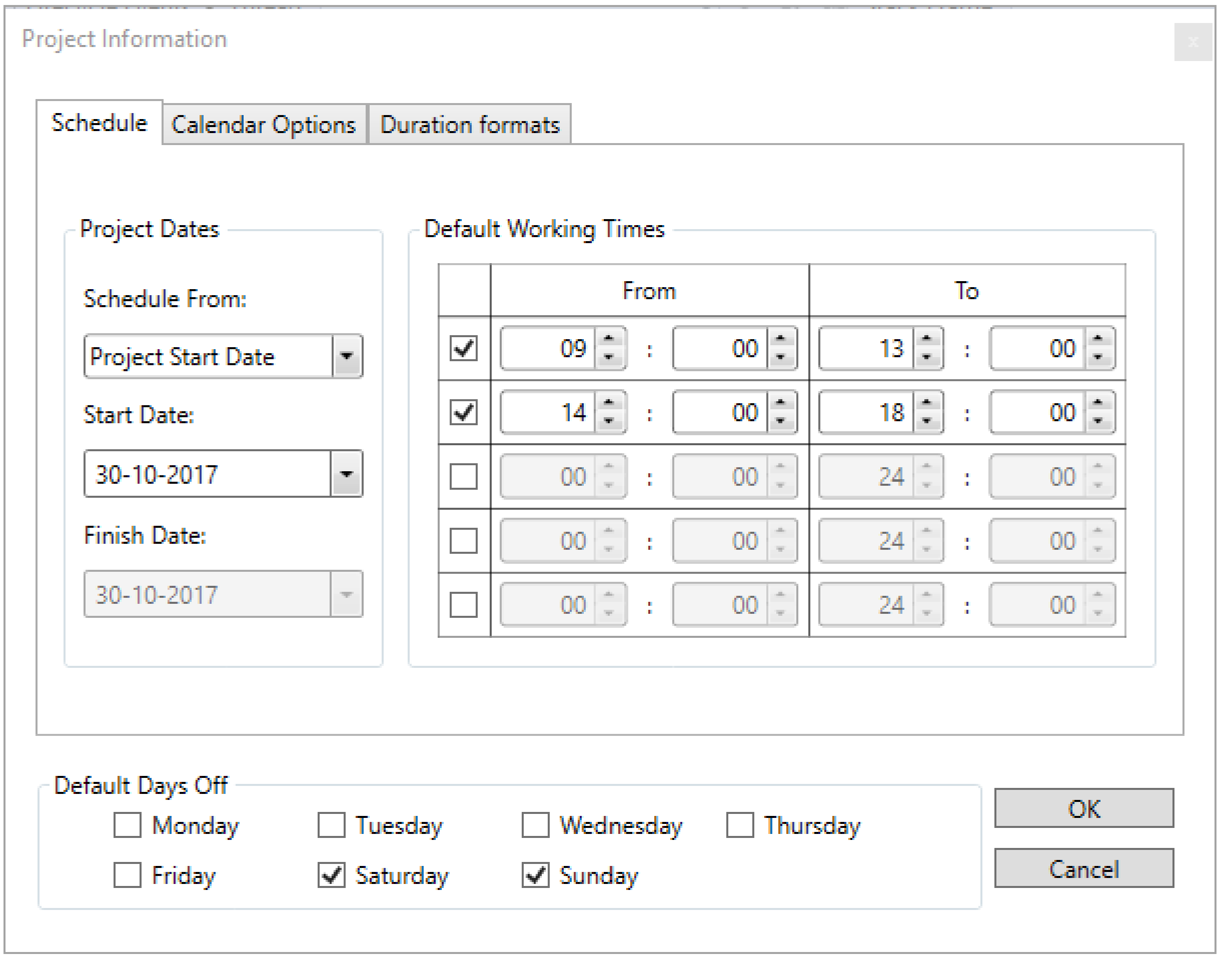
Option	**Purpose**
Sort By	Select a column to sort from the drop-down list.
Then By	Select the next column to sort from the drop-down list.
Ascending/Descending	Select the sort order from the drop-down list.
Reset	Reset the selected sort order from the drop-down list.
OK	Apply the selected sort order.
Cancel	Close the dialog.

Project Information Dialog

The **Project Information** is used for scheduling a new project. This dialog appears on clicking the **Project Information** button on the toolbar. The Project Information dialog consists three tabs, **Schedule**, **Calendar Options**, and **Duration Formats.** The Project Information dialog also includes **Default Days Off** groupbox that lets you select the days off in the project schedule.

Schedule

The following image shows the schedule tab of the Project Information dialog:



Project Dates

Schedule From: Project Start Date

Start Date: 30-10-2017

Finish Date: 30-10-2017

Default Working Times

	From			To		
✓	09	:	00	13	:	00
✓	14	:	00	18	:	00
☐	00	:	00	24	:	00
☐	00	:	00	24	:	00
☐	00	:	00	24	:	00

Default Days Off

☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday

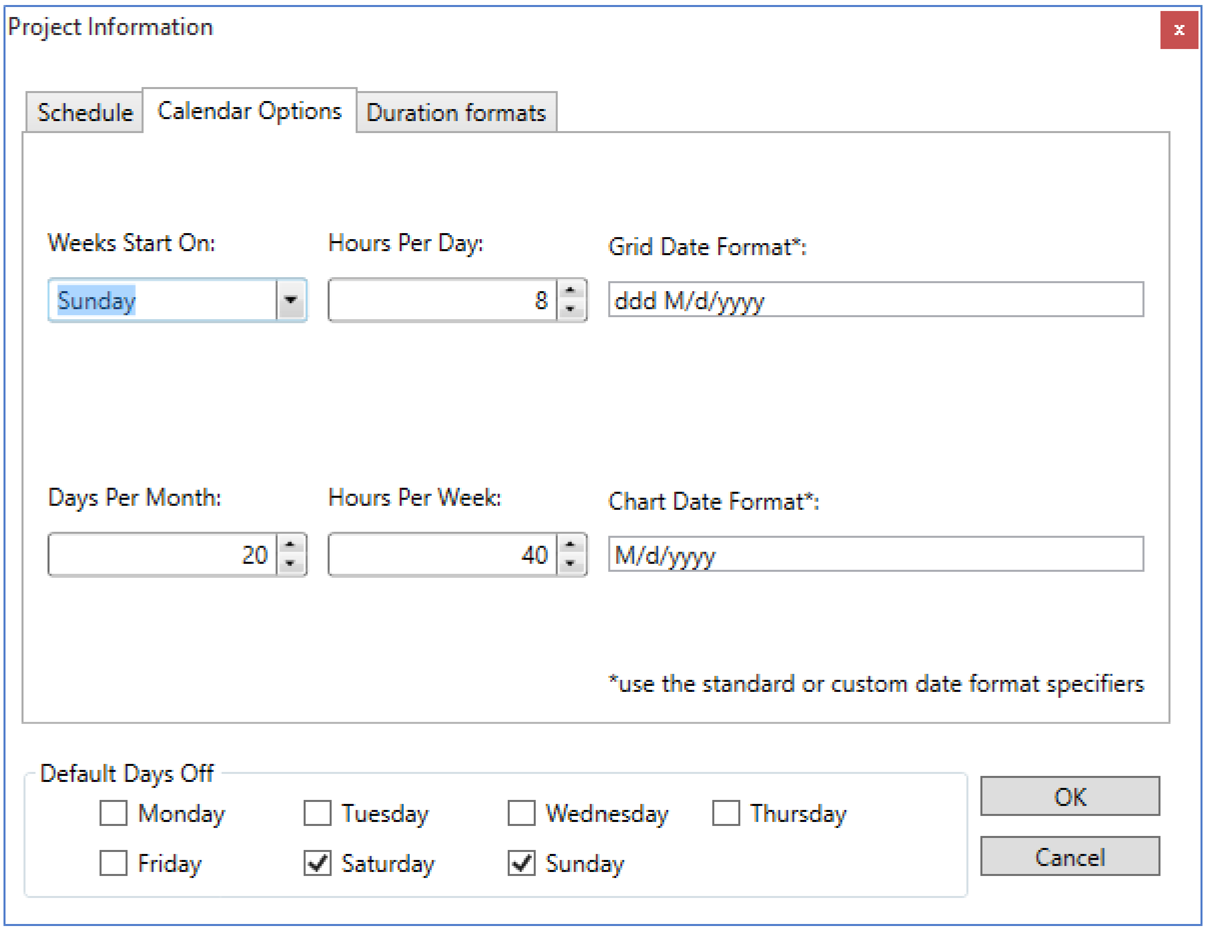
☐ Friday ✓ Saturday ✓ Sunday

The Schedule tab includes various options as follows:

Option	**Purpose**
Schedule From	Schedule the project either from Project Start Date or Project Finish Date. By default, any project schedule begins from the project start date.
Start Date	Select the Project Start Date from the drop-down calendar if you wish to schedule your project from project start date.
Finish Date	Select the Project Finish Date from the drop-down calendar if you wish to schedule your project from project finish date.

Calendar Options

The following image shows the Calendar Options tab of the Project Information dialog:


Project Information

Schedule | Calendar Options | Duration formats

Weeks Start On: Sunday

Hours Per Day: 8

Grid Date Format*: ddd M/d/yyyy

Days Per Month: 20

Hours Per Week: 40

Chart Date Format*: M/d/yyyy

*use the standard or custom date format specifiers

Default Days Off

☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday

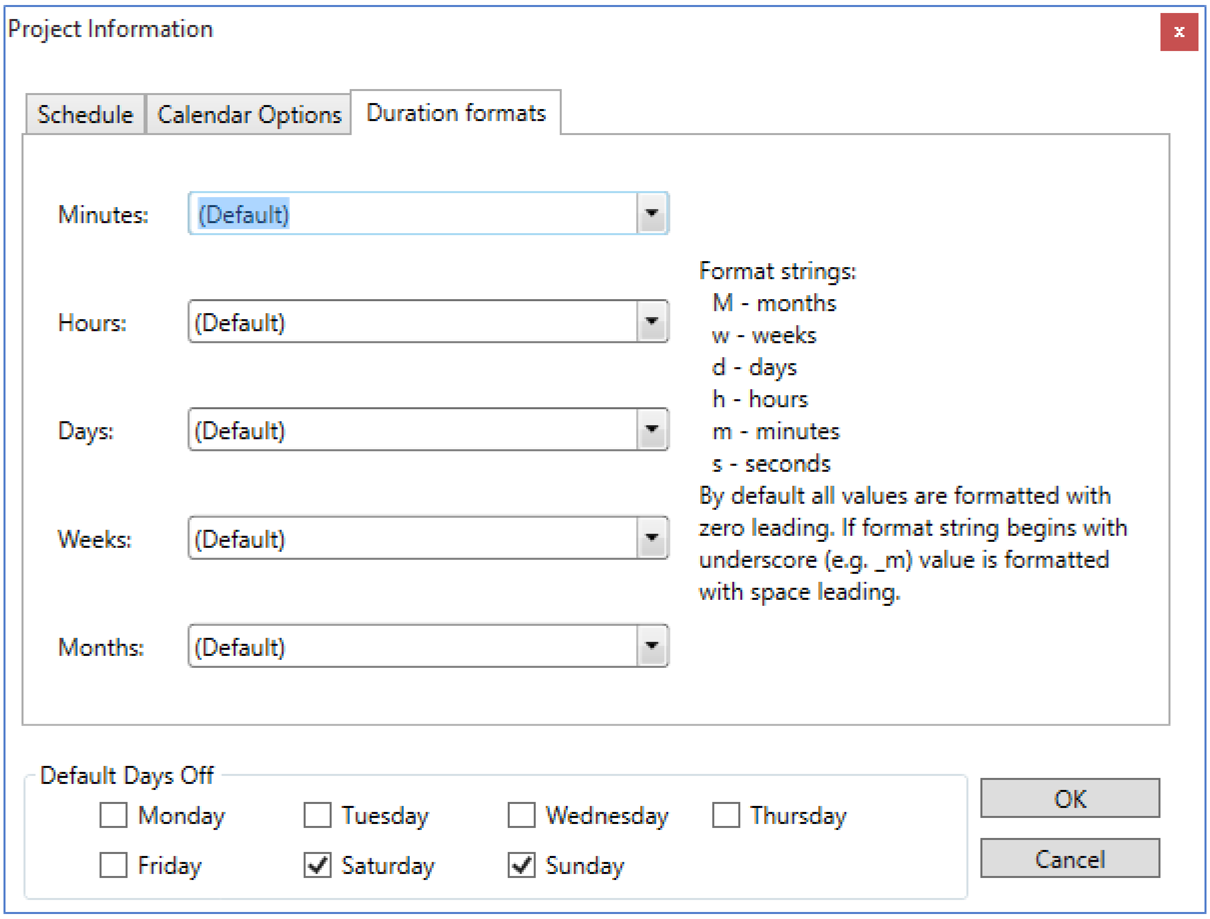
☐ Friday ☑ Saturday ☑ Sunday

The Calendar Options tab includes the following items:

Option	**Purpose**
Weeks Start On	Specify the day of the week on which the scheduled task starts from the drop-down list. By default, this field is set to Sunday.
Hours Per Day	Specify the hours per day for the scheduled task in the numeric box. GanttView calculates duration units as One day equals 8 hours.
Grid Date Format	Specify the date format appearing in the grid in the text box.
Days Per Month	Specify the total working days in a normal calendar month for automatically-scheduled tasks in the numeric box. By default, one month has 20 working days.
Hours Per Week	Specify the working hours per week for the automatically-scheduled task in the numeric box. By default, one working week equals 40 hours.
Chart Date Format	Specify the format for date appearing on chart in the text box. The format is based on standard or custom date format specifier.

Duration Formats

The following image shows the Duration Formats tab of the Project Information dialog:


Project Information

Schedule | Calendar Options | Duration formats

Minutes: (Default)

Hours: (Default)

Days: (Default)

Weeks: (Default)

Months: (Default)

Format strings:
M - months
w - weeks
d - days
h - hours
m - minutes
s - seconds
By default all values are formatted with zero leading. If format string begins with underscore (e.g. _m) value is formatted with space leading.

Default Days Off
☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday
☐ Friday ✓ Saturday ✓ Sunday

The Duration Formats tab includes the following options:

Option	Purpose
Minutes	Specify the format string for minutes.
Hours	Specify the format string for hours.
Days	Specify the format string for days.
Weeks	Specify the format string for weeks.
Months	Specify the format string for months.

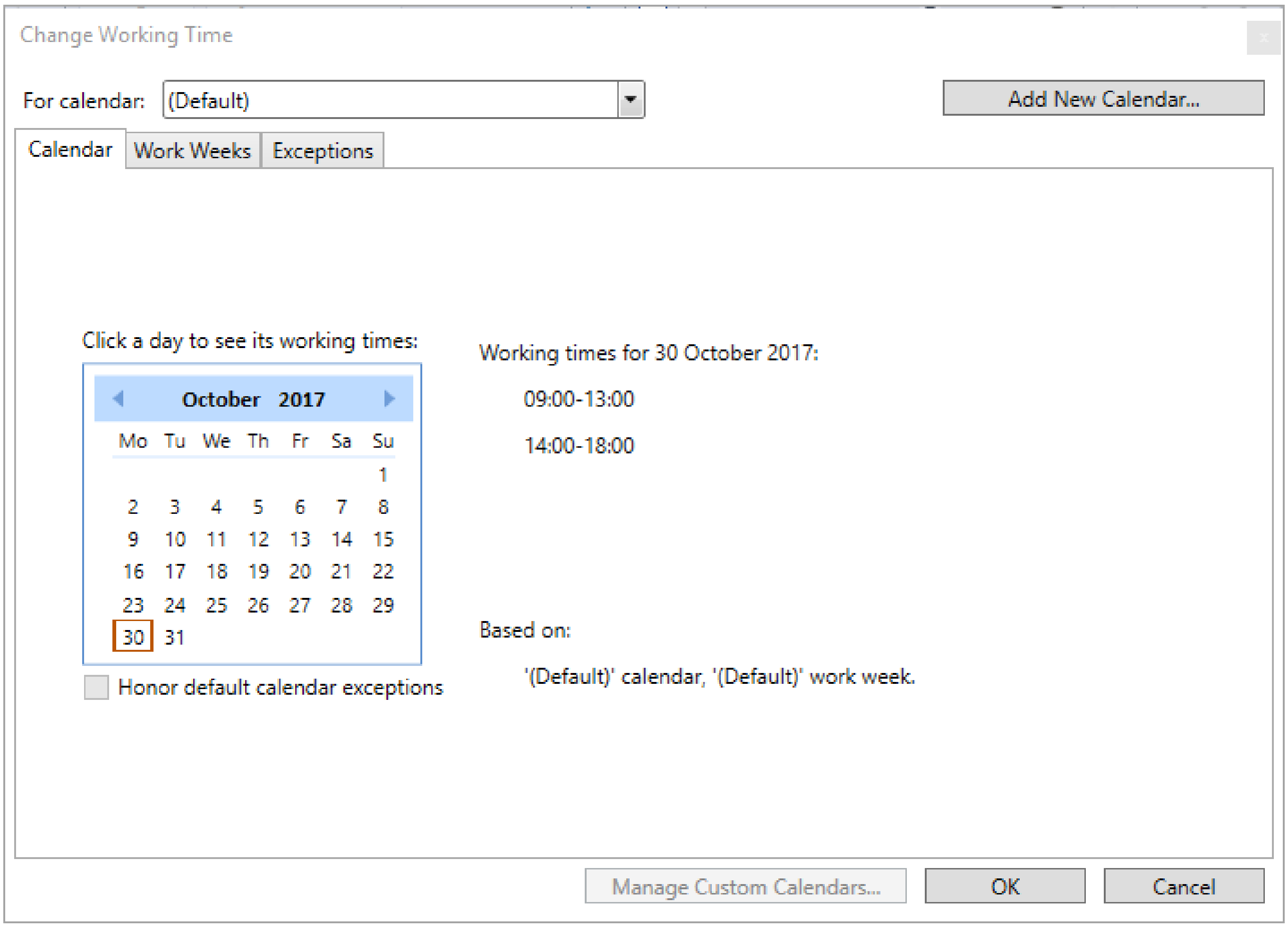
Change Working Time Dialog

The **Change Working Time** dialog is used to change the working time in project calendar. This dialog appears on clicking the **Change Working Time** button on the toolbar.

The Change Working Time dialog provides options as follows:

Option	Purpose
For calendar	Specify the calendar you wish to select.
Add New Calendar	Create a new calendar or a copy of a calendar based on default or custom calendar.
Manage Custom Calendars	Add or remove custom calendar.

The following image shows the Change Working Time dialog:



Working times for 30 October 2017:

09:00-13:00

14:00-18:00

Based on:

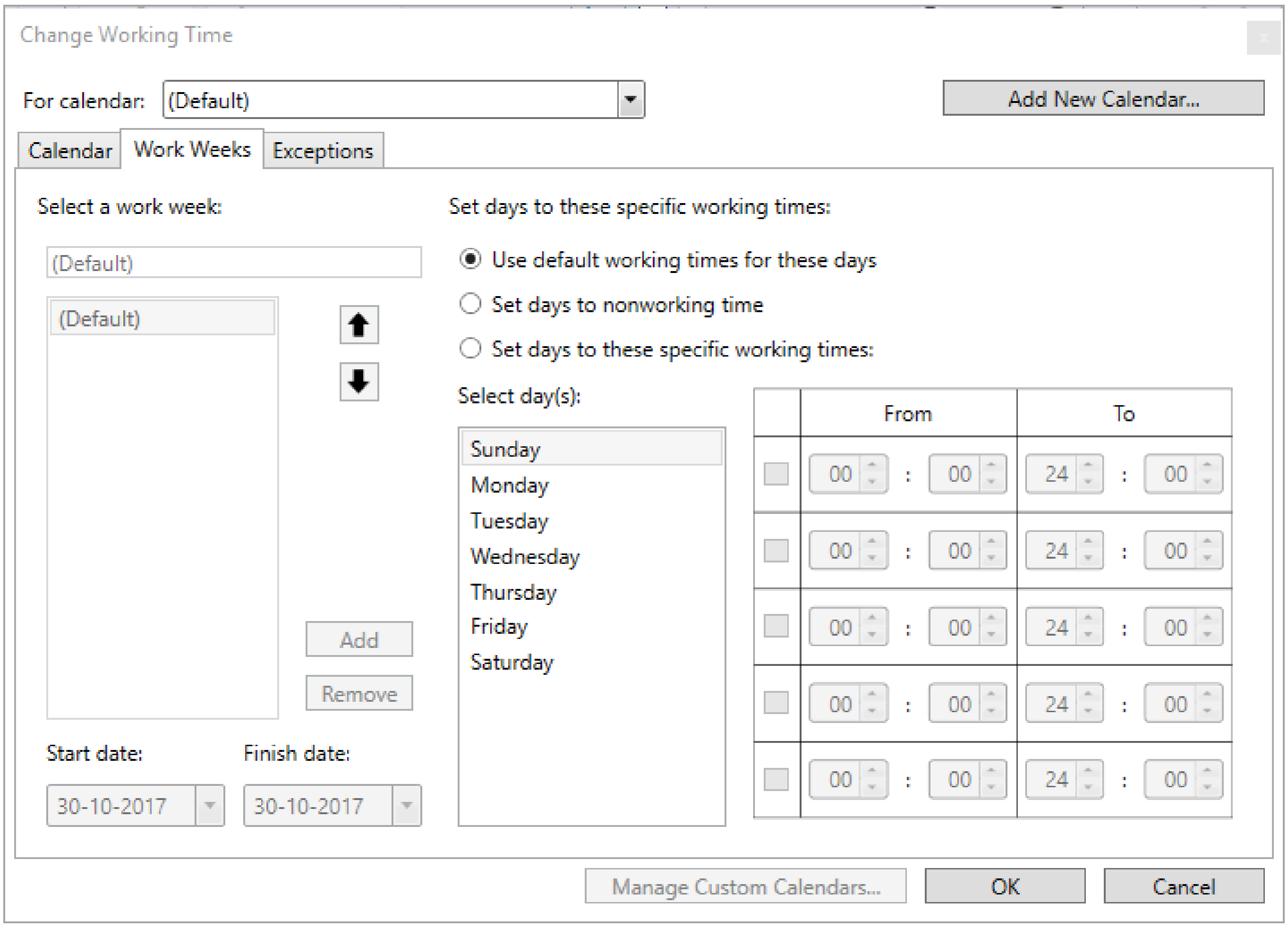
'(Default)' calendar, '(Default)' work week.

The Change Working Time dialog displays three tabs, namely **Calendars**, **Work Weeks**, **Exceptions**. The Calendars tab displays a calendar where you can click a day to view the working time based on the work week, and check whether or not to honor default calendar exceptions.

Work Weeks

The Work Weeks tab provides option to change the default work week for project calendar, resource calendar, or for any custom calendar that you create. You can also choose or create additional schedule for a range of days that differ from the default work day.

The following image shows the Work Weeks tab:


Change Working Time

For calendar: (Default)

Select a work week: (Default)

Set days to these specific working times:

(● Use default working times for these days)
○ Set days to nonworking time
○ Set days to these specific working times:

Select day(s): Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

	From	To
☐	00 : 00	24 : 00
☐	00 : 00	24 : 00
☐	00 : 00	24 : 00
☐	00 : 00	24 : 00
☐	00 : 00	24 : 00

Start date: 30-10-2017
Finish date: 30-10-2017

The Work Week tab provides various options as follows:

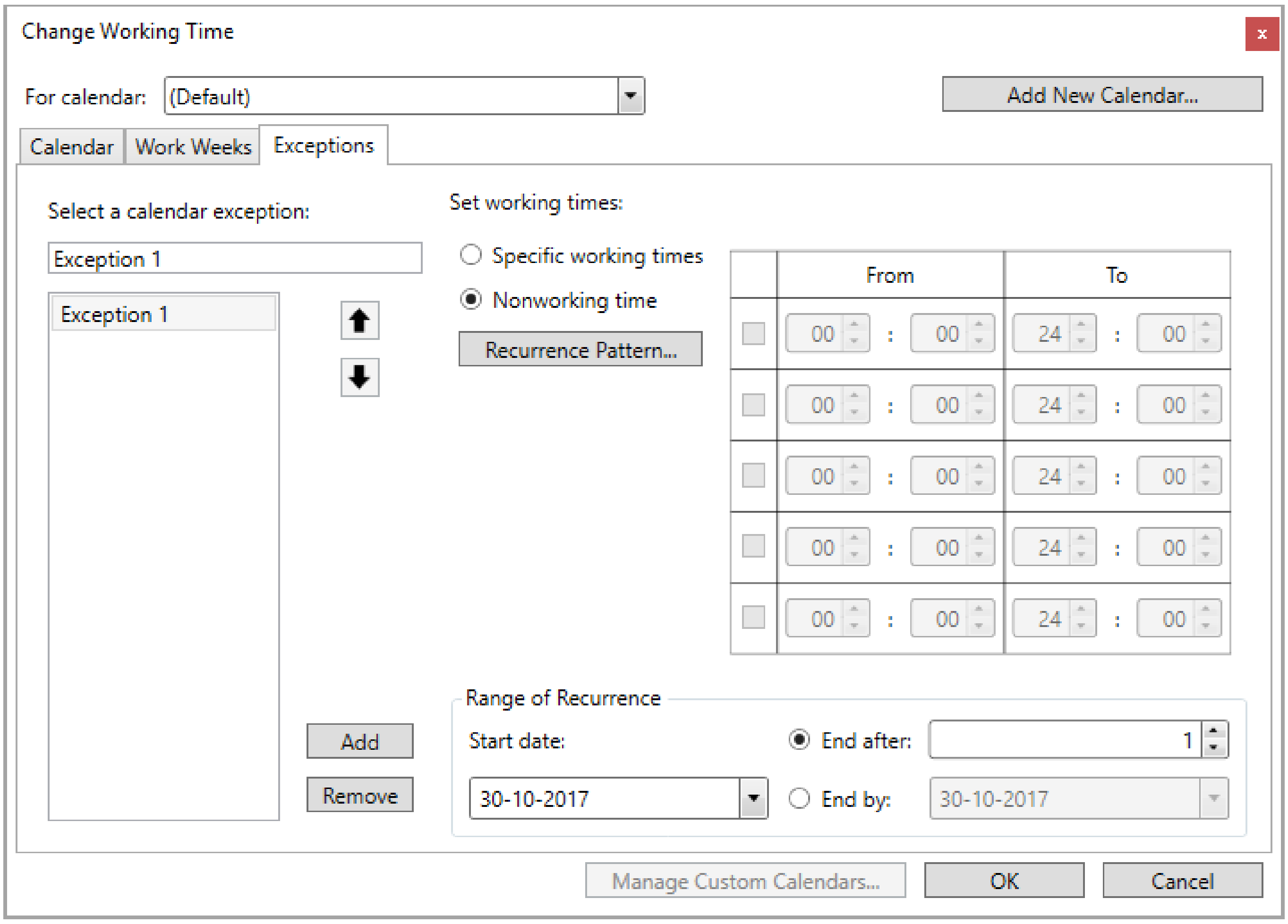
Option	**Purpose**
Select day(s)	Select one or more days that constitute a work week.
Name	Specify a name for a working week.
Start Date	Specify the Start date for a working week.
Finish Date	Specify the Finish date for a working week.

Once you select the day(s) you want to change from a working day to non-working day or vice versa, you can choose one of the following:

- Use default working times for these days - to specify the default working times, that is 8:00 am - 12:00 am and 1:00 pm - 5:00 pm, on selected days in the work week.
- Set days to nonworking time - to select nonworking day(s).
- Set days to these specific working times - to set the working times for the selected days throughout the schedule. Type the time when you wish the work to start in the **From** box and the time you want the work to end in the **To** box.

Exceptions

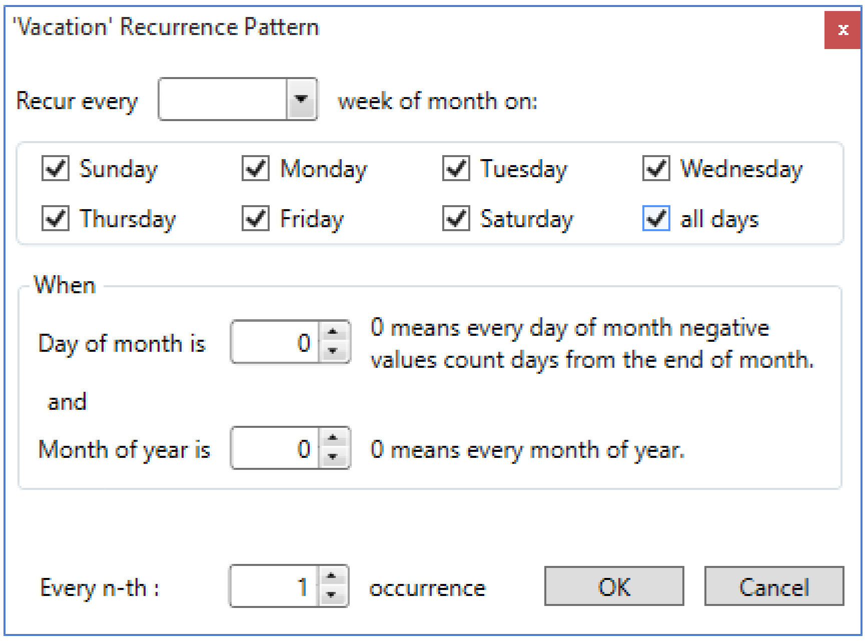
The Exceptions tab provides you the option to add exceptions in the calendar such as marking public holidays or vacation. The following image shows how this tab appears at runtime:



The Exception tab provides options as follows:

Option	Purpose
Start date	Shows a drop-down menu with a calendar to select the start date of exception.
End by	Shows a drop-down menu with a calendar to select the end date of exception.
Recurrence Pattern	Shows a button to set a repeat on an exception throughout the schedule. Clicking this button opens Recurrence Pattern dialog that lets you specify the days on which recurrence appear. Any task scheduled around an exception automatically gets rescheduled.

The following image shows how the Recurrence Pattern dialog appears at runtime:


'Vacation' Recurrence Pattern

Recur every [ ] week of month on:

☑ Sunday ☑ Monday ☑ Tuesday ☑ Wednesday
☑ Thursday ☑ Friday ☑ Saturday ☑ all days

When

Day of month is 0 — 0 means every day of month negative values count days from the end of month.

and

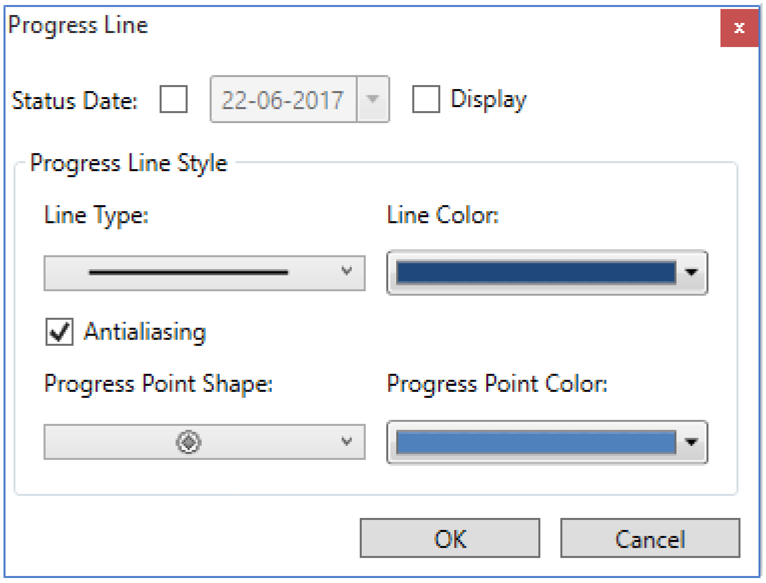
Month of year is 0 — 0 means every month of year.

Every n-th : 1 occurrence

Progress Line Dialog

The **Progress Line** dialog is used for displaying progressive lines on the chart view to highlight the tasks that are ahead of or behind the schedule. This dialog appears on clicking the **Progress Line** button on the toolbar.

The following image shows the Progress Line dialog:


Progress Line

Status Date: ☐ 22-06-2017 ☐ Display

Progress Line Style

Line Type:

Line Color:

☑ Antialiasing

Progress Point Shape:

Progress Point Color:

The Progress Line dialog provides various options as follows:

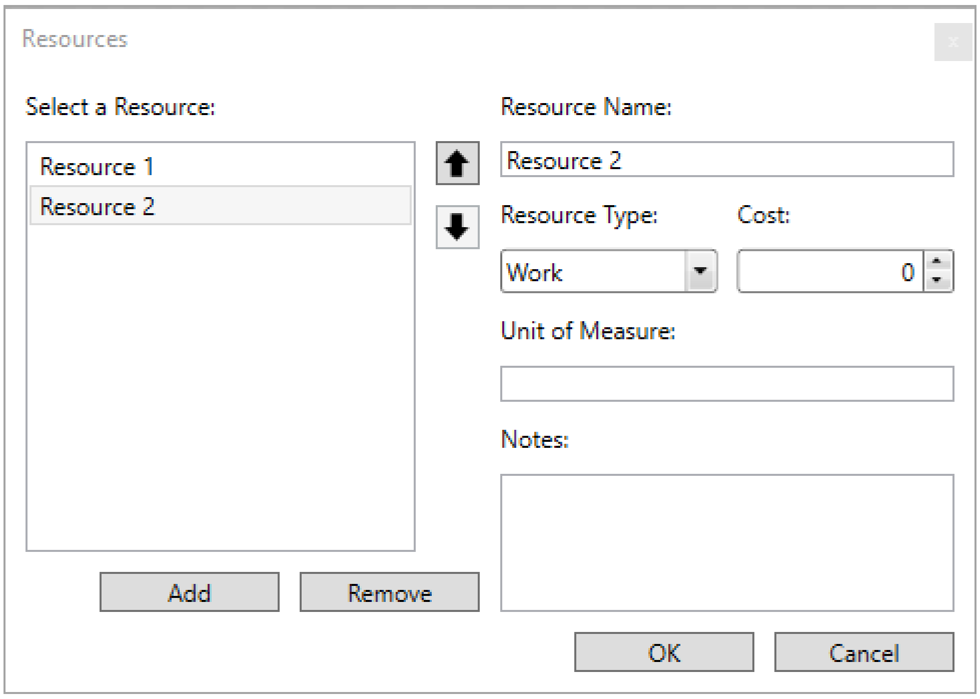
Option	**Purpose**
Status Date	Select the check box to set a date for checking the status from.
Display	Select check box to enable the progress line.
Line Type	Select the type of line to be drawn for the progress line from the drop-down list.
Line Color	Select a color for the progress line from the drop-down menu.

Progress Point Shape	Select the shape of the indicator for the starting point of the progress line from the drop-down list.
Progress Point Color	Select a color for the progress point on the progress line from the drop-down menu.

Resources Dialog

The **Resources** dialog is used for adding and removing resources, and specifying various attributes such as resource name, resource type, cost, and unit of measure. This dialog appears on clicking the **Project Resources** button on the toolbar.

The following image shows the Resources dialog:



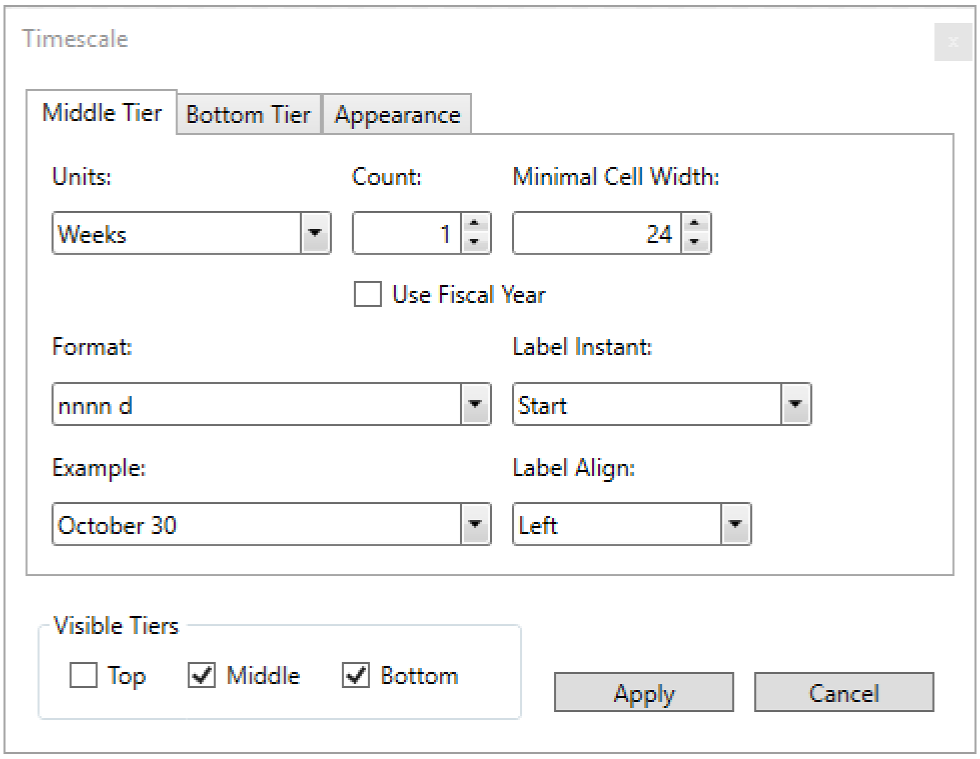
The Resources dialog provides various options as follows:

Option	Purpose
Add	Click the button to add a new resource in the text box.
Remove	Click the button to remove the selected resource from the text box.
Resource Name	Specify the resource name for the added resources in the text box.
Resource Type	Select the resource type such as Cost, Material or Work from the drop-down list.
Cost	Specify the per unit cost of the selected resource type in the numeric box.
Unit of Measure	Specify the unit of measure for the selected resource type in the text box. For example, Cost type resources are usually measured in currency, Material type resources are measured in tons, meters, etc., and Work type resources are usually measured in hours.
Notes	Specify details or notes to understand how the resource is calculated on the basis of the unit of measure in the text box.

Timescale Dialog

The **Timescale** dialog is used to set the visibility of various timescale tiers, setting date and time, and customize appearance of current day, nonworking time and project start/finish line. This dialog appears on clicking the **Timescale** button on the toolbar at runtime.

The Timescale dialog comprises three tabs, namely Middle Tier, Bottom Tier and Appearance tab as shown in the following image:


Timescale

Middle Tier | Bottom Tier | Appearance

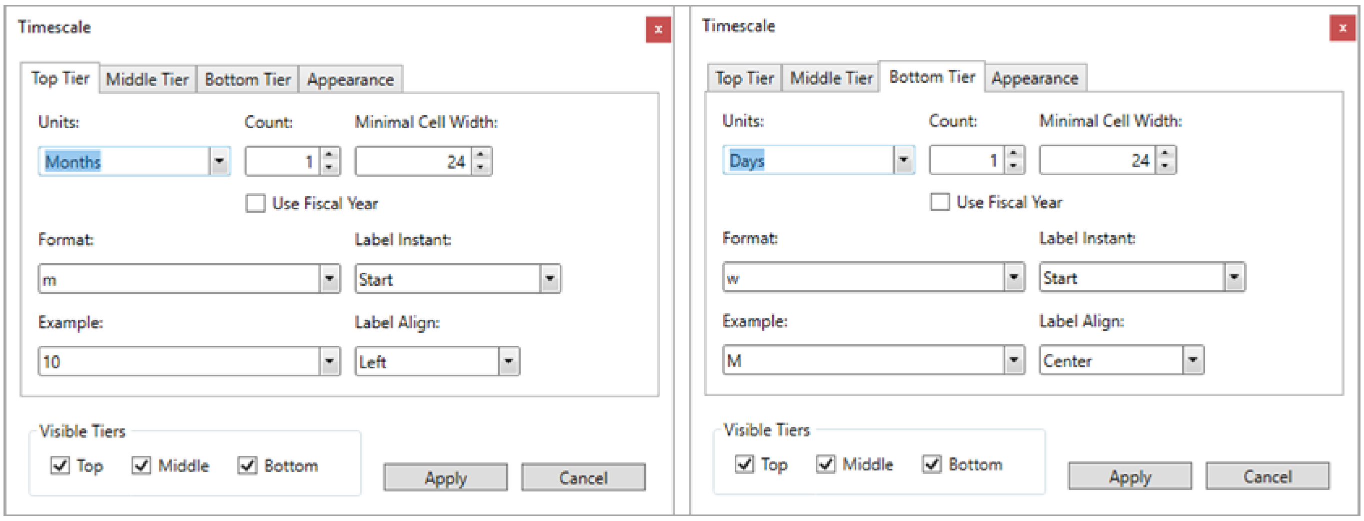
Units: Weeks
Count: 1
Minimal Cell Width: 24
☐ Use Fiscal Year
Format: nnnn d
Label Instant: Start
Example: October 30
Label Align: Left

Visible Tiers: ☐ Top ☑ Middle ☑ Bottom

Apply | Cancel

Top Tier and Bottom Tier Tabs

The following image shows the Top Tier and Bottom Tier tabs in the Timescale dialog:


Timescale

Top Tier | Middle Tier | Bottom Tier | Appearance

Units: Months
Count: 1
Minimal Cell Width: 24
☐ Use Fiscal Year
Format: m
Label Instant: Start
Example: 10
Label Align: Left

Visible Tiers: ☑ Top ☑ Middle ☑ Bottom

Apply | Cancel

Timescale

Top Tier | Middle Tier | Bottom Tier | Appearance

Units: Days
Count: 1
Minimal Cell Width: 24
☐ Use Fiscal Year
Format: w
Label Instant: Start
Example: M
Label Align: Center

Visible Tiers: ☑ Top ☑ Middle ☑ Bottom

Apply | Cancel

The Top, Middle, and Bottoms Tier tabs include various options as follows:

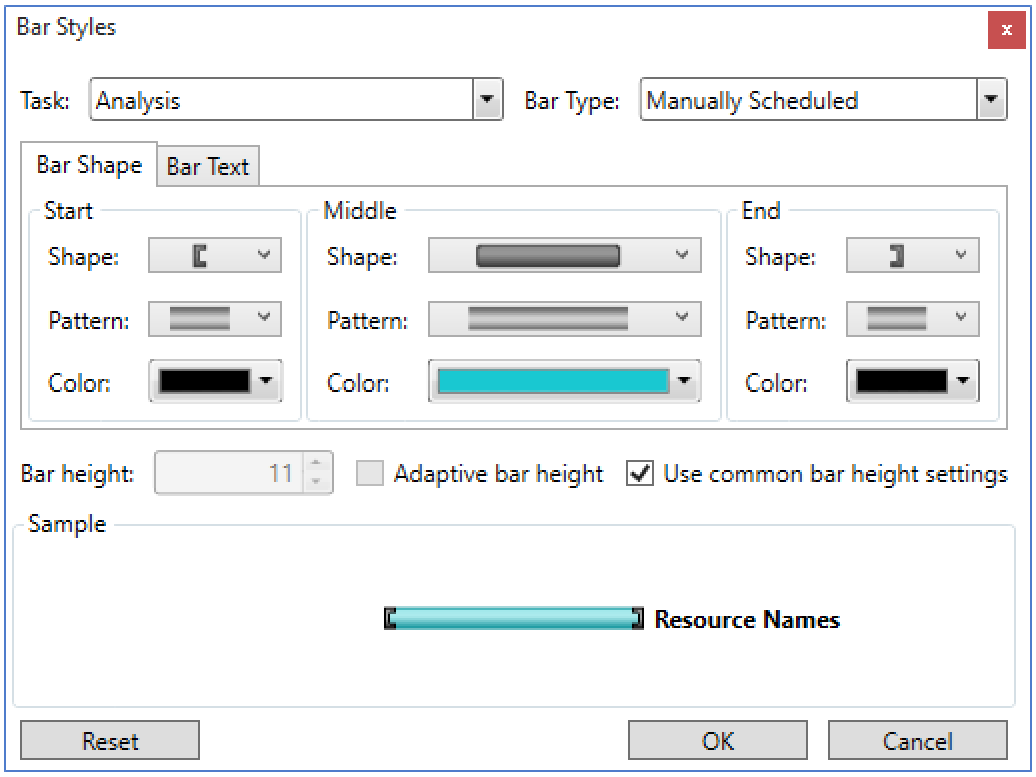
Option	**Purpose**

Units	Select the unit of time from the drop-down list.
Count	Specify the frequency of unit labels appearing on timescale tiers in the numeric box.
Minimal Cell Width	Specify the minimal cell width in pixels for each tier in the numeric box.
Format	Specify the format to display time from the drop-down list. By default, the format is set to nnnn d.
Label Instant	Specify where the instant label is placed: Start, End, Middle, Range, or Overlapped Range from the drop-down list.
Example	Displays results for the date format that you select in the Format drop-down.
Label Align	Specify the alignment for a timescale tier from the drop-down list that provides Left, Center, Right and Justify options.

Bar Styles Dialog

The **Bar Styles** dialog is used for customizing the task bars displayed in the chart view of the control. This dialog appears on clicking the **Bar Styles** button on the toolbar.

The following image shows the Bar Styles dialog:



Task: Analysis
Bar Type: Manually Scheduled
Shape:
Pattern:
Color:
Bar height: 11
☐ Adaptive bar height
☑ Use common bar height settings
Resource Names

The Bar Styles dialog provides various options as follows:

Option	Purpose
Task	Select a particular task for customizing the bar style from the drop-down list.
Bar Type	Select a bar type for the selected task from the drop-down list.

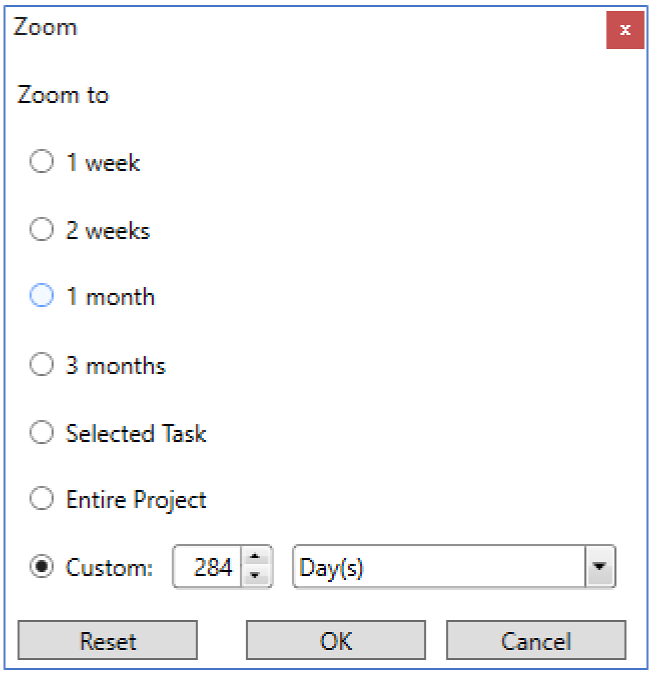
The Bar Styles dialog also includes two tabs namely **Bar Shape** and **Bar Text** to further customize shape, pattern and color of the task bar. The tab provides various customization options as follows:

Option	Purpose
Bar Shape	Set the **Shape**, **Pattern** and **Color** for the Start, Middle and End portions of the task bar from the drop-down lists.
Bar Text	Specify the text that appears against the task bar from the drop-down list.

Zoom Dialog

The **Zoom** dialog is used to zoom-in or zoom-out the scale view of timescale tiers in the chart view. The dialog appears on selecting the **Zoom** option available in the **Zoom** menu on the toolbar.

The following image shows the Zoom dialog:



The Zoom dialog provides various options as follows:

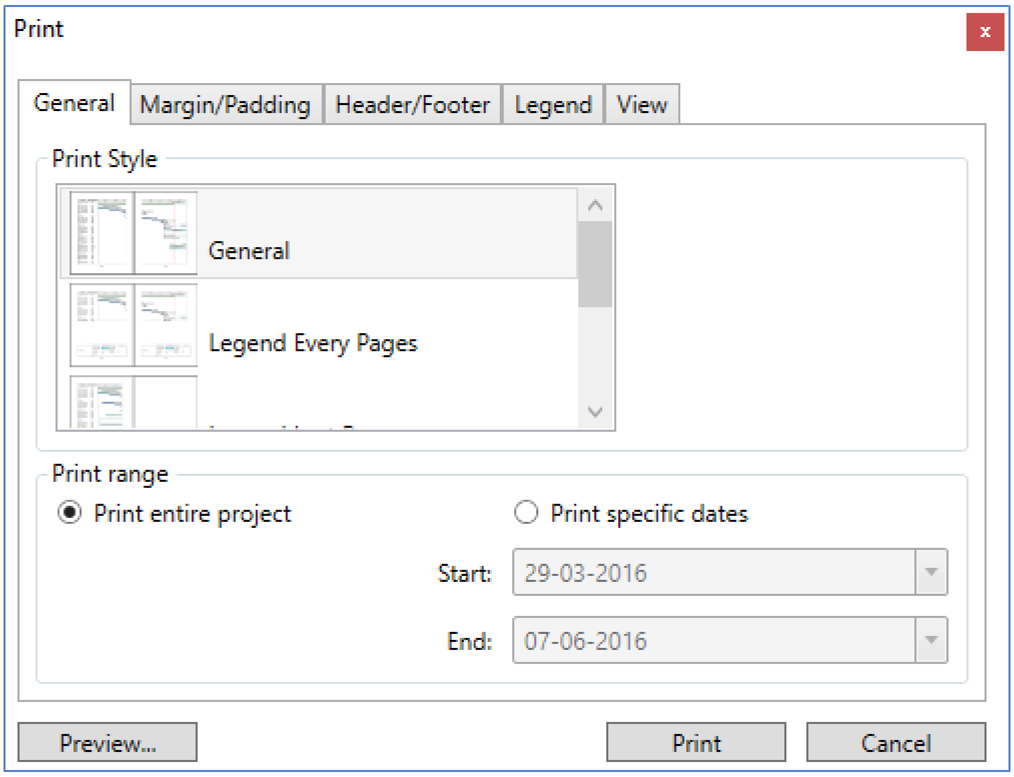
Option	Purpose
Zoom to	Select the radio buttons from the time range to zoom the GanttView.
Custom	Set the custom time range in the form of count in the numeric box and choose the unit from the drop-down list alongside.
Reset	Click the button to reset the selection.
OK	Click the button to apply the selected zoom.
Cancel	Click the button to close the dialog.

Print- Dialog

The **Print** dialog is used for managing print options such as selecting the print style, specifying content and font settings for header, footer and legend, and determining print range. The dialog appears on clicking the **Print** button on the toolbar. The Print dialog shows five tabs namely **General**, **Margin/Padding**, **Header/Footer**, **Legend** and

View.

General


Print

General | Margin/Padding | Header/Footer | Legend | View

Print Style

- General
- Legend Every Pages

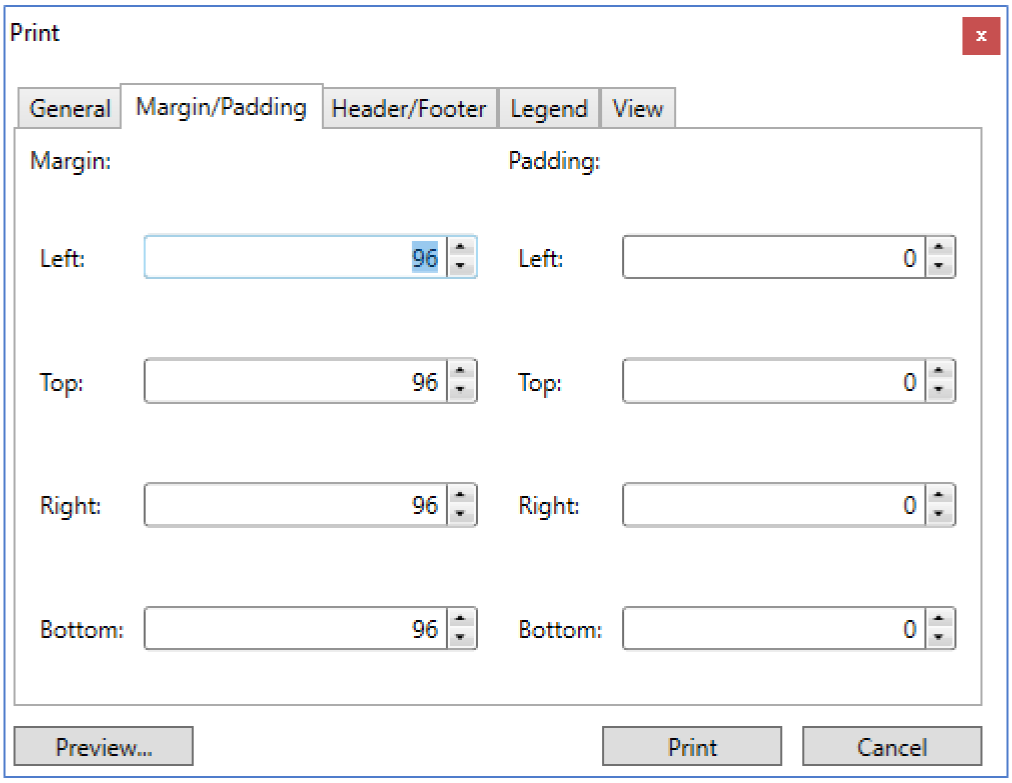
Print range

(● Print entire project) ○ Print specific dates

Start: 29-03-2016

End: 07-06-2016

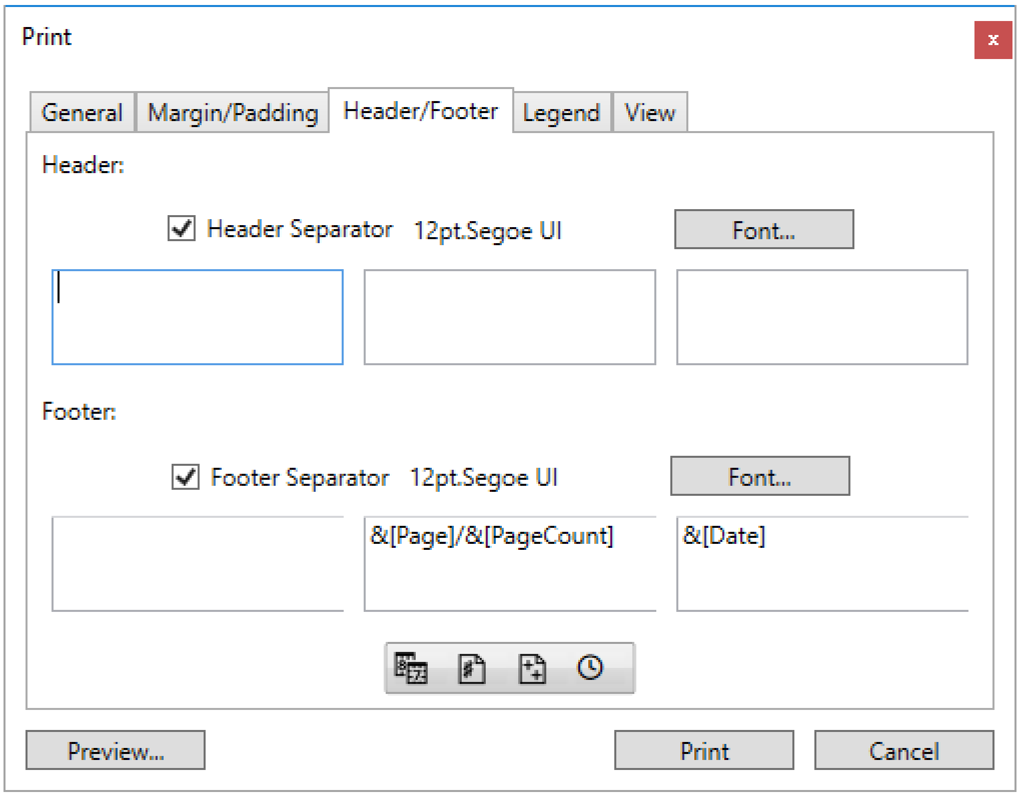
Margin/Padding


Print

General | Margin/Padding | Header/Footer | Legend | View

Margin:		Padding:	
Left:	96	Left:	0
Top:	96	Top:	0
Right:	96	Right:	0
Bottom:	96	Bottom:	0

Header/Footer


Print

General | Margin/Padding | Header/Footer | Legend | View

Header:

☑ Header Separator 12pt.Segoe UI [Font...]

Footer:

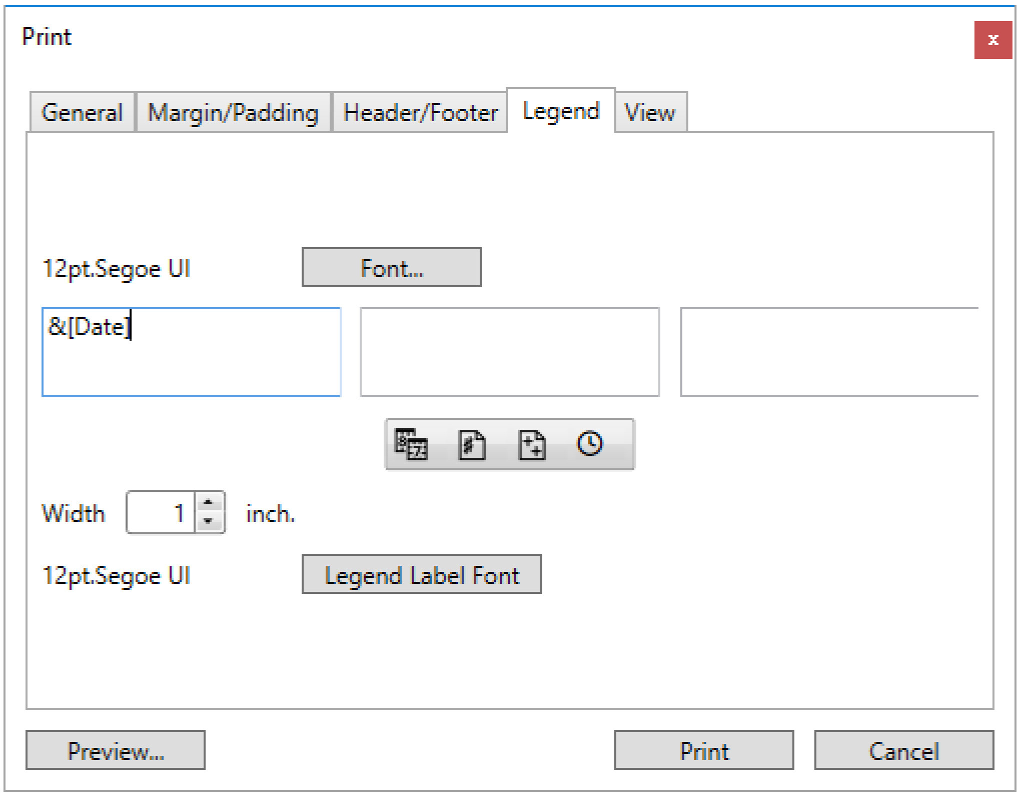
☑ Footer Separator 12pt.Segoe UI [Font...]

&[Page]/&[PageCount]

&[Date]

[Preview...] [Print] [Cancel]

Legend


Print

General | Margin/Padding | Header/Footer | Legend | View

12pt.Segoe UI [Font...]

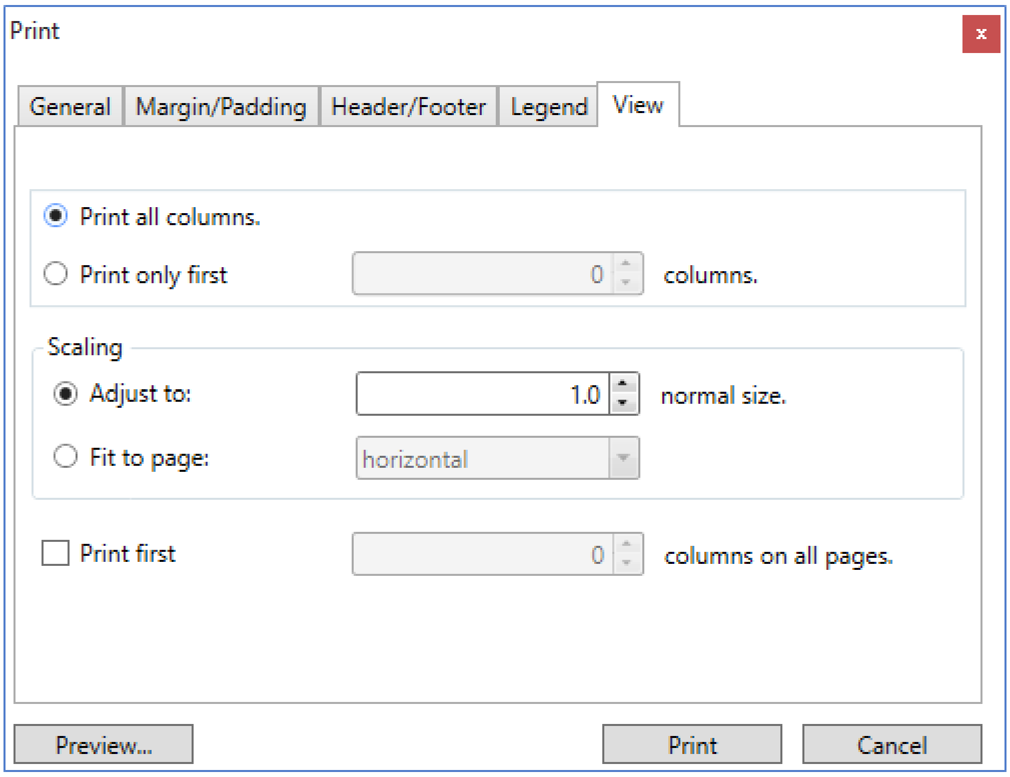
&[Date]

Width 1 inch.

12pt.Segoe UI [Legend Label Font]

[Preview...] [Print] [Cancel]

View



(● Print all columns.)
○ Print only first 0 columns.

Scaling
(● Adjust to:) 1.0 normal size.
○ Fit to page: horizontal

☐ Print first 0 columns on all pages.

Features

GanttView supports a number of features to help users effectively manage and schedule projects. To know more about these features, click the following links:

- Task Management
- Style and Appearance

Task Management

GanttView provides you a number of task management features such as creating custom columns, allocating resources, applying filters, customizing timescales, etc. Learn about these features in detail and how they can be implemented.

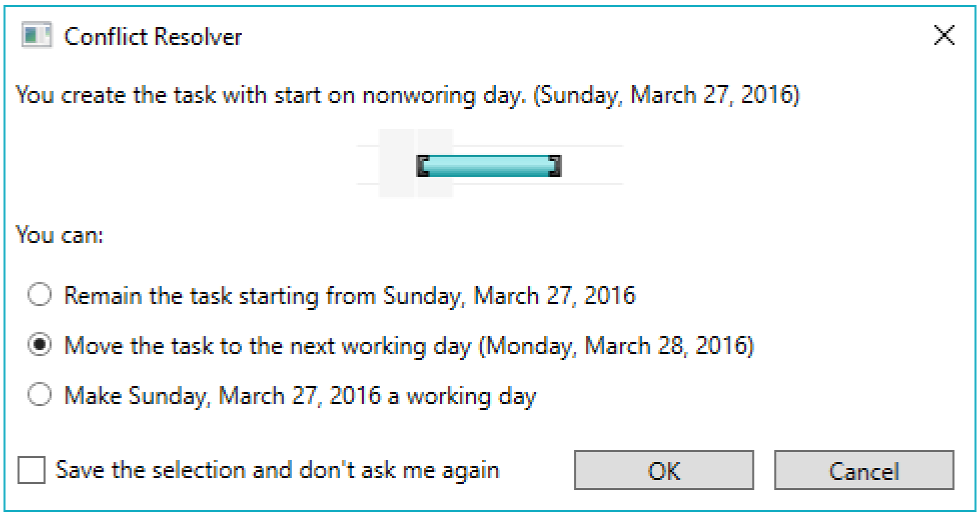
Tasks
    Learn how to create and add tasks in code.
Columns
    Learn how to display default columns available in GanttView.
Custom Columns
    Learn how to create custom columns as per your requirements and add them to GanttView.
Scheduling mode
    Learn how to set scheduling mode for a task in code.
Percent complete
    Learn how to set percent completion status for a task in code.
Task deadline
    Learn how to set deadline for a particular task in code.
Timescale
    Learn how to customize timescale and timescale formats in code.
Milestone task
    Learn how to create a milestone tasks in code.
Progress line
    Learn how to create, customize and progress lines in code.
Resources
    Learn how to create, add and assign resources to a tasks in code.
Constraints
    Learn how to apply constraints in code.
Filtering
    Learn how to filter tasks at runtimes as well as in code.
Sorting
    Learn how to sort tasks by name, start/finish date, duration, etc. and apply custom sort order.
Saving content in XML
    Learn how to save the contents of a gantt view along with layout in an XML file.
Loading content from XML
    Learn how to load the contents from a gantt view along with the layout from an XML file..
Task dependency
    Learn how to create a predecessor task and setting its dependency on another task in code.
Task splitting
    Learn how to split tasks in code.
Project summary task
    Learn how to create summary tasks in code.

Tasks

Any project plan revolves around tasks, which represent the amount of work to be done for accomplishing a set of goals. Tasks define work in terms of duration, dependencies (if any), and resource requirements.

GanttView supports adding tasks to a project plan both at runtime as well as in code.

At runtime, the control lets you add a task through the **Add Task** button available in the toolbar. Clicking the **Add Task** button opens the Task Information dialog to set basic properties of the task such as name, percent complete, start and finish dates, etc. You can also create new tasks by dragging the mouse in the chart view. Creating tasks through mouse drag with start date on a nonworking day such as Sunday or Saturday opens a dialog to resolve the conflict. The image below shows a **Conflict Resolver** dialog.



Conflict Resolver

You create the task with start on nonworing day. (Sunday, March 27, 2016)

You can:

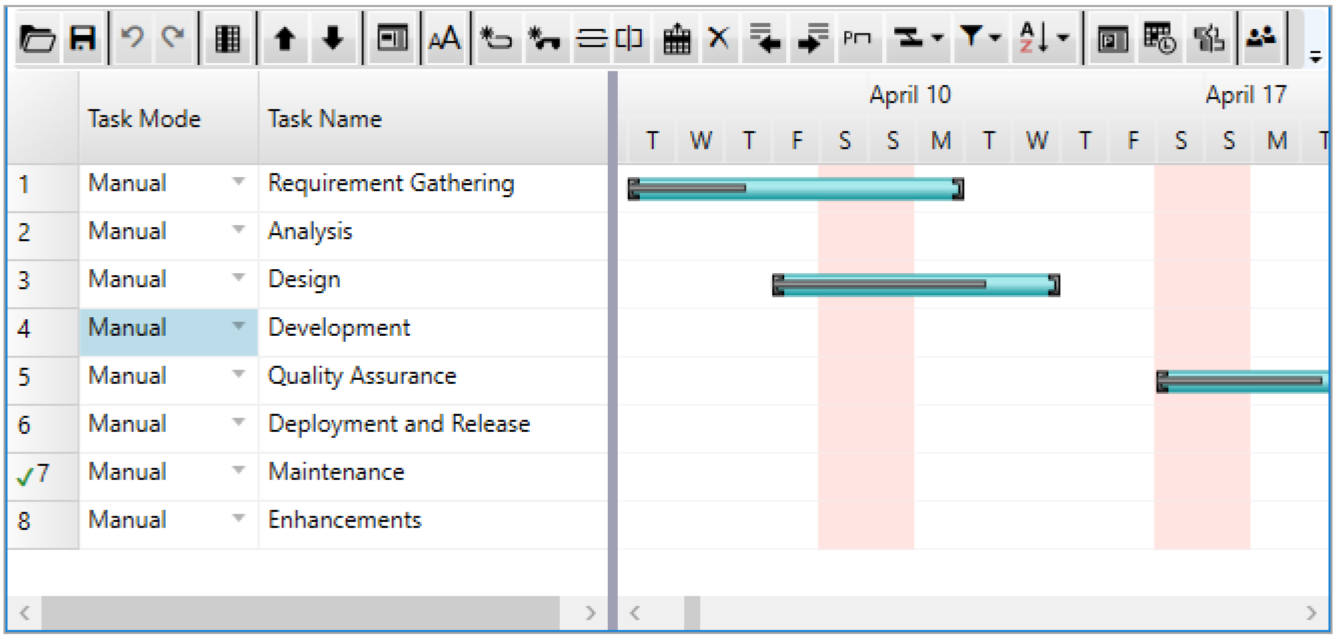
- Remain the task starting from Sunday, March 27, 2016
- [✓ Move the task to the next working day (Monday, March 28, 2016)]
- Make Sunday, March 27, 2016 a working day

☐ Save the selection and don't ask me again

The Conflict Resolver dialog resolves the conflict by providing users three options:

- Start the task from a nonworking day
- Move the task to next working day
- Make Sunday or Saturday a working day

The image below shows tasks added to GanttView.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

In addition, GanttView supports several kinds of tasks such as milestones and summary tasks discussed in Milestone task and Project summary task sections.

Creating and adding tasks in code

In code, tasks can be created by defining an instance of the Task class and setting its basic properties such as name and duration through the Name and Duration properties. A task can be added to a GanttView through the Add method of the TaskCollection class. The following code illustrates creating a task, setting basic properties, and adding it to GanttView.

Visual Basic

```
'Creating a task and setting basic properties
Dim t As New Task()
t.Mode = TaskMode.Manual
t.Name = "Requirement Gathering"
t.Start = New DateTime(2016, 4, 5)
t.Duration = 5

'Adding task to GanttView
gv.Tasks.Add(t)
```

C#

```
//Creating a task and setting basic properties
Task t = new Task();
t.Mode = TaskMode.Manual;
t.Name = "Requirement Gathering";
t.Start = new DateTime(2016, 4, 5);
t.Duration = 5;

//Adding task to GanttView
gv.Tasks.Add(t);
```

Columns

GanttView displays task related attributes such as task name, mode, start and finish date, etc. in task property columns, or simply columns. The control provides 14 default columns to display these attributes in the grid view as follows:

Column Name	**Description**
Task Mode	Displays task mode, that is Manual or Automatic
Task Name	Displays task name
Duration	Displays duration for tasks
Duration Units	Displays duration unit
Start	Displays the start date
Finish	Displays the finish date
% Complete	Displays the completion status of a task in percentage
Constraint Type	Displays the constraint type
Constraint Date	Displays the constraint date

Predecessors	Displays the predecessor task
Deadline	Displays the task deadline
Calendar	Displays the calendar
Resource Name	Displays the name of the resource assigned to a particular task
Notes	Displays the notes corresponding to a particular task

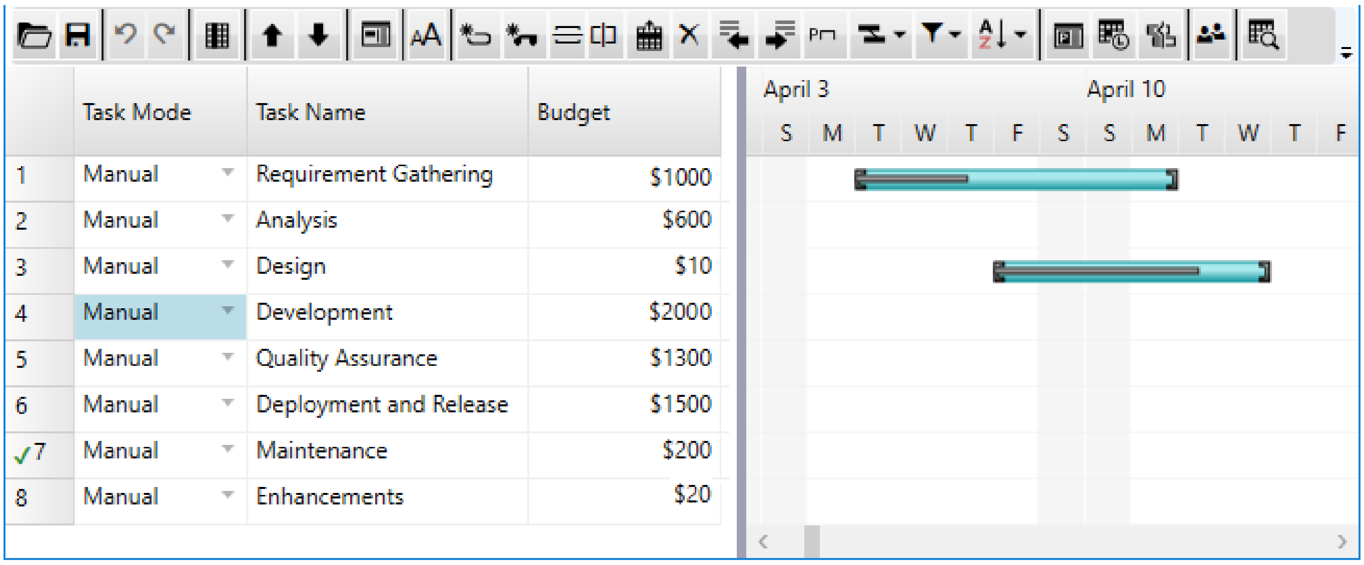
Adding Columns

On placing the GanttView control on the designer, the **Task Mode** and **Task Name** columns are visible by default. To display more columns in the GanttView, you can use the Grid Columns Dialog at runtime.

Custom Columns

GanttView control is not restricted to default task property columns as users can create custom columns as per their requirements. The control provides the CustomFieldColumn class, which can be used to create custom columns. The class also provides a set of properties to specify various attributes of a custom column. For instance, the DataType property can be used to set the data type, while the Format property can be used to specify the format in which the data is to be displayed.

The image below shows a GanttView with a custom column titled 'Budget' that accepts numeric data and displays the same in '$' format.



	Task Mode	Task Name	Budget
1	Manual	Requirement Gathering	$1000
2	Manual	Analysis	$600
3	Manual	Design	$10
4	Manual	Development	$2000
5	Manual	Quality Assurance	$1300
6	Manual	Deployment and Release	$1500
✓7	Manual	Maintenance	$200
8	Manual	Enhancements	$20

Creating custom columns in code

To create custom columns, create an object of the CustomFieldColumn class and set the basic properties such as Name, Caption, Format, and DataType. The following code illustrates creating a custom column titled 'Budget' that accepts numeric data and specifies the currency format in '$'. This example uses the sample created in the Quick Start.

Visual Basic

```
' Creating a Custom Column
Dim customColumn As New CustomFieldColumn()
customColumn.Caption = "Budget"
customColumn.Name = "Budget"
customColumn.DataType = GetType(Decimal)
customColumn.Format = "$#0"
customColumn.Width = 100
gv.Columns.Add(customColumn)
```

C#

```
// Creating a Custom Column
CustomFieldColumn customColumn = new CustomFieldColumn();
customColumn.Caption = "Budget";
customColumn.Name = "Budget";
customColumn.DataType = typeof(decimal);
customColumn.Format = "$#0";
customColumn.Width = 100;
gv.Columns.Add(customColumn);
```

Scheduling Mode

GanttView supports two types of modes to schedule tasks in GanttView, namely **Manual** and **Automatic.**

Manual Mode

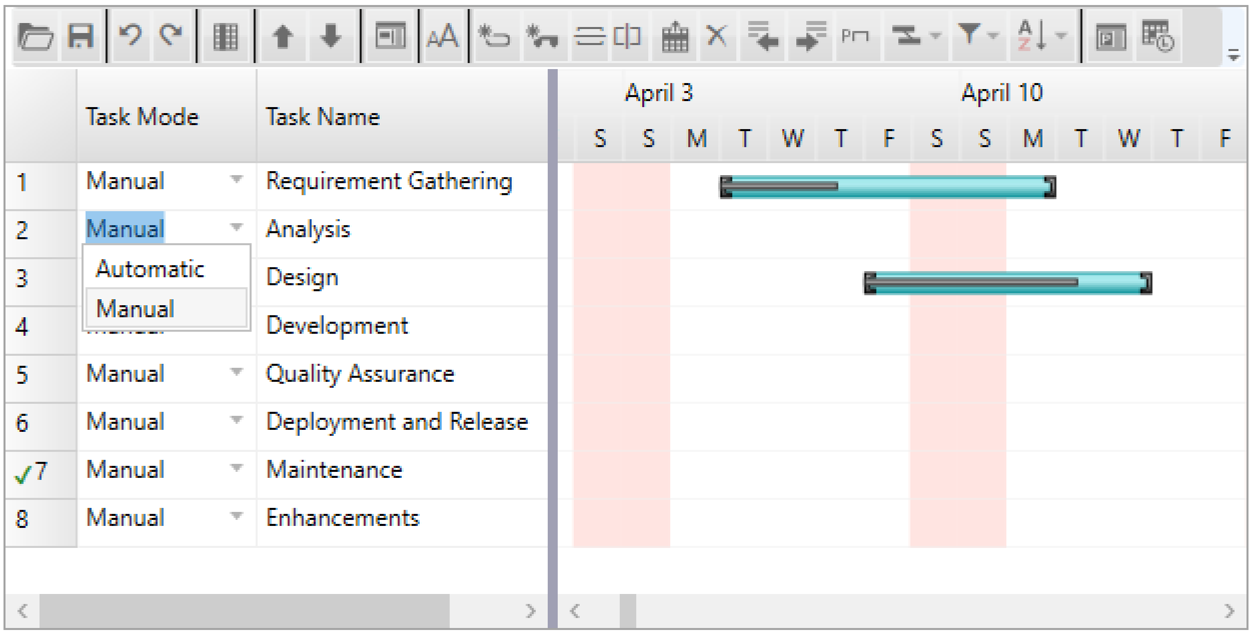
This is the default task mode for tasks in GanttView. The manual mode provides greater flexibility to users in planning and managing project schedules. Manual tasks do not change the scheduling for project resources, etc. This mode is most suited for users not having adequate information related to tasks in a project.

Automatic Mode

The automatic mode provides more structured way to manage project schedules. In automatic mode, the GanttView control calculates the best earliest and latest dates for a task once the user enters information such as task duration, number of resources, etc. In case of any change in the schedule, GanttView automatically adjusts the project schedule for optimum results.

Scheduling modes can be set at runtime through the Task information dialog, which can be accessed by clicking the **Task Information** button from the toolbar.

The following image shows how to set task mode at runtime.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3		Design
4		Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Setting task mode in code

To set the scheduling mode for a task in GanttView, the Task class provides the Mode property that accepts the following values from the TaskMode enumeration.

- **Manual** - Allows you to set the start date for a particular task.
- **Automatic** - Automatically calculates the start date for a task.

The following code illustrates how to set the Mode property for a task.

Visual Basic

```
Dim t As New Task()
t.Mode = TaskMode.Manual
```

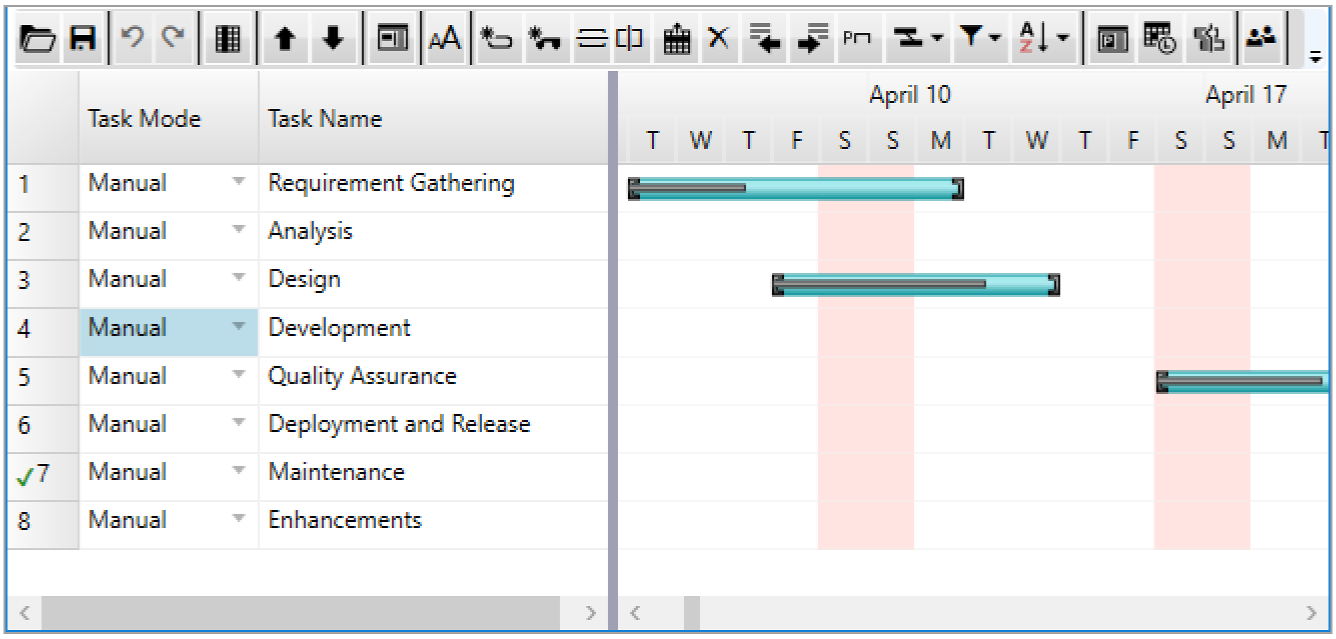
C#

```
Task t = new Task();
t.Mode = TaskMode.Manual;
```

Percent Complete

GanttView provides percent complete feature to specify the completion status of a task in percentage. This feature helps users in keeping a track of all the tasks within a project schedule. The percent complete for a task can be set in GanttView through the Task information dialog, which can be accessed by clicking the **Task Information** button from the toolbar.

The task percent complete visually appears as a dark colored bar drawn inside the task bar.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Setting percent complete in code

To set percent completion status for a task, the Task class provides the PercentComplete property. This property accepts any positive double type integer value between 0 and 1, where 0 denotes that the task is not started while 1 denotes the task is complete.

The following code illustrates how to set the PercentComplete property for a task.

Visual Basic

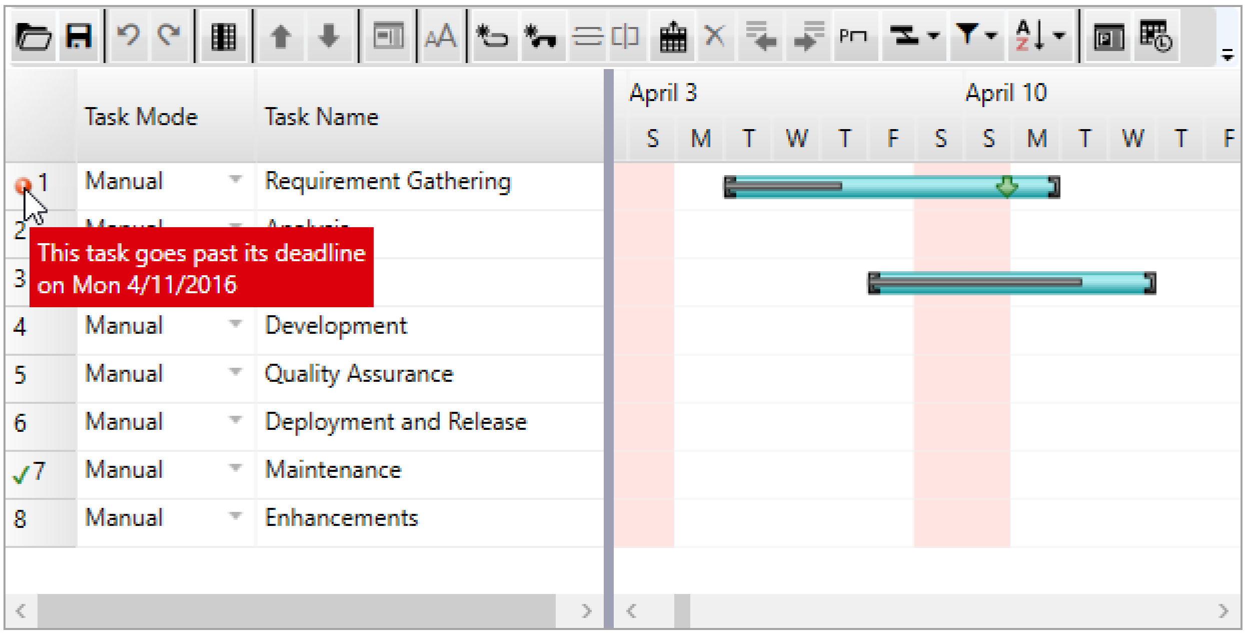
```
' PercentComplete value 0.6 means tha task is 60% complete
Dim t As New Task()
t.PercentComplete = 0.6
```

C#

```
// PercentComplete value 0.6 means the Task is 60% complete
Task t = new Task();
t.PercentComplete = 0.6;
```

Task Deadline

Task deadline enables users to monitor each and every activity involved within a project. GanttView also supports deadlines that allow users to keep a track of tasks exceeding their set deadline dates. Tasks exceeding their deadline dates get highlighted by a red indicator in the GanttView and shows a message box on hovering as shown in the following image.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3		
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

This task goes past its deadline on Mon 4/11/2016

Task deadlines can be set at runtime through the Task information dialog, which can be accessed by clicking the **Task Information** button from the toolbar.

Setting task deadline in code

To set the deadline for a particular task, you can use the Deadline property provided by the Task class. The Deadline property accepts an object of the DateTime class to set the deadline date. The deadline for a task can be set in code by creating an instance of the Task class and setting its Deadline property to an object of DateTime class.

The following example illustrates how to set the deadline date for a task. This example uses the sample created in the Quick Start.

Visual Basic

```
'Setting deadline
Dim t As New Task()
```

```
t.Name = "Requirement Gathering"
t.Start = New DateTime(2016, 4, 5)
t.Deadline = New DateTime(2016, 4, 11)
```

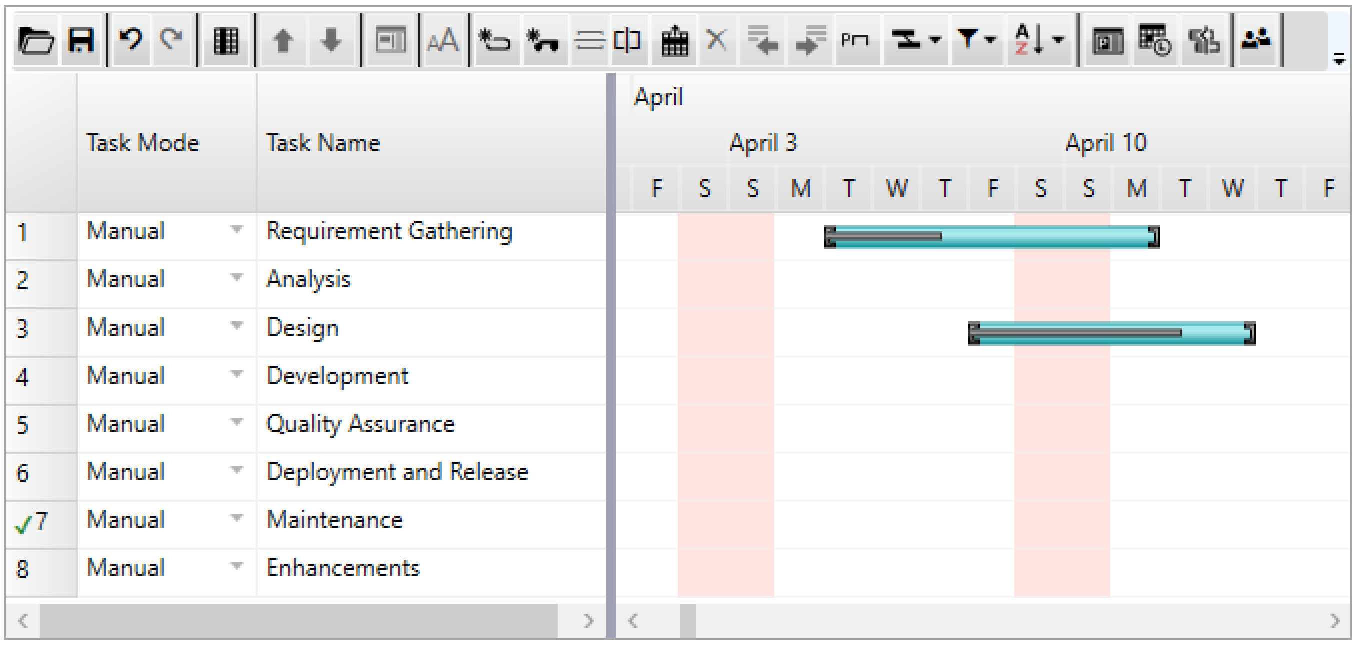
C#

```
//Setting deadline
Task t = new Task();
t.Name = "Requirement Gathering";
t.Start = new DateTime(2016, 4, 5);
t.Deadline = new DateTime (2016, 4, 11);
```

Timescale

GanttView supports a timescale along the horizontal timeline, which is displayed in bands or tiers. The timescale can be customized to display a maximum of three possible tiers namely the bottom tier, the middle tier, and the top tier. The bottom tier displays abbreviated names of days in a week, while the middle tier displays work week. The top tier, which by default is hidden, displays month or year.

The following image shows a GanttView with all the three tiers visible in the timescale.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Setting timescale in code

The GanttView control provides the Timescale class to adjust the display settings of the timescale, and the ScaleTier class to customize the format and units for the three tiers. The Timescale class provides BottomTier, MiddleTier and TopTier properties to specify the settings of the three tiers. The ScaleTier class provides properties such as Visible to set the visibility of timescale tiers, and Format to set the format string of the labels appearing in the three tiers.

The following code illustrates setting the visibility of top tier, and the format string of its label.

Visual Basic

```
'Setting the visibility and format for TopTier
gv.Timescale.TopTier.Visible = True
gv.Timescale.TopTier.Format = "nnnn"
```

C#

```
//Setting the visibility and format for TopTier
gv.Timescale.TopTier.Visible = true;
gv.Timescale.TopTier.Format = "nnnn";
```

Timescale Formats

GanttView supports various formats specifiers for the timescale labels. You can specify these format specifiers through the Format property of the ScaleTier class.

Standard Date and Time Formats

Format	Description
s	s is the standard date/time format specifier. For example, s(x) where 'x' is a placeholder for one of the following letters: d, D, f, g, m, t, y.
sd	Short date pattern (11/11/2016)
sD	Long date pattern (Thursday, November 11, 2016)
sf	Full date and time pattern (Friday, November 11, 2016 10:00 AM)
sg	General date and time pattern (11/11/2016 10:00 AM)
sm	Month day pattern (April 10)
st	Short time pattern (10:00 AM)
sy	Year month pattern (April, 2008)

Year Formats

Format	Description
yy	Represents the year as a two-digit number
yyy	Represents the year with a minimum of three digits
yyyy	Represents the year as a four-digit number

Half Year Formats

Format	Description
h	Represents a half year as a number 1 or 2
h {N1, N2}	Customized half year representation

Quarterly Formats

Format	Description
q	Represents a quarter as a number from 1 through 4
q {N1, N2, N3, N4}	Customized quarter representation

Calendar Month Formats

Format	Description
m	Represents the month as a number from 1 through 12
mm	Represents the month as a number from 01 through 12
n	Single-letter month name
nnn	Use DateTimeFormatInfo.GetAbbreviatedMonthName
nnnn	Use DateTimeFormatInfo.GetMonthName
n {N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11, N12}	Custom month name
e {N1, N2, N3}	Custom thirds-of-month name

Week of the Year Number Formats

Format	Description
k	Represents the week of the year number 1 to 53
kk	Represents the week of the year number 01 to 53

Day of the Month Number Formats

Format	Description
d	Represents the day of the year as a number from 1 through 31
dd	Represents the day of the year as a number from 01 through 31

Day of the Year Number Formats

Format	Description
b	Represents the day of the year as a number from 1 through 366
bbb	Represents the day of the year as a number from 001 through 366

Single Letter Week Day Formats

Format	Description
w	Single-letter week day (For example, S, M, T, W)
ww	Use DateTimeFormatInfo.GetShortestDayName
www	Use DateTimeFormatInfo.GetAbbreviatedDayName
wwww	Use DateTimeFormatInfo.GetDayName
w {N1, N2, N3, N4, N5, N6, N7}	Custom week day name

Time Formats

Format	Description
a	Represents the hour as a number from 1 through 12
aa	Represents the hour as a number from 01 through 12
u	Represents the hour as a number from 0 through 23
uu	Represents the hour as a number from 00 through 23
i	Represents the minute as a number from 0 through 59
ii	Represents the minute as a number from 00 through 59
t	Represents the first character of AM/PM designator
tt	Represents the AM/PM designator

Other Specifiers

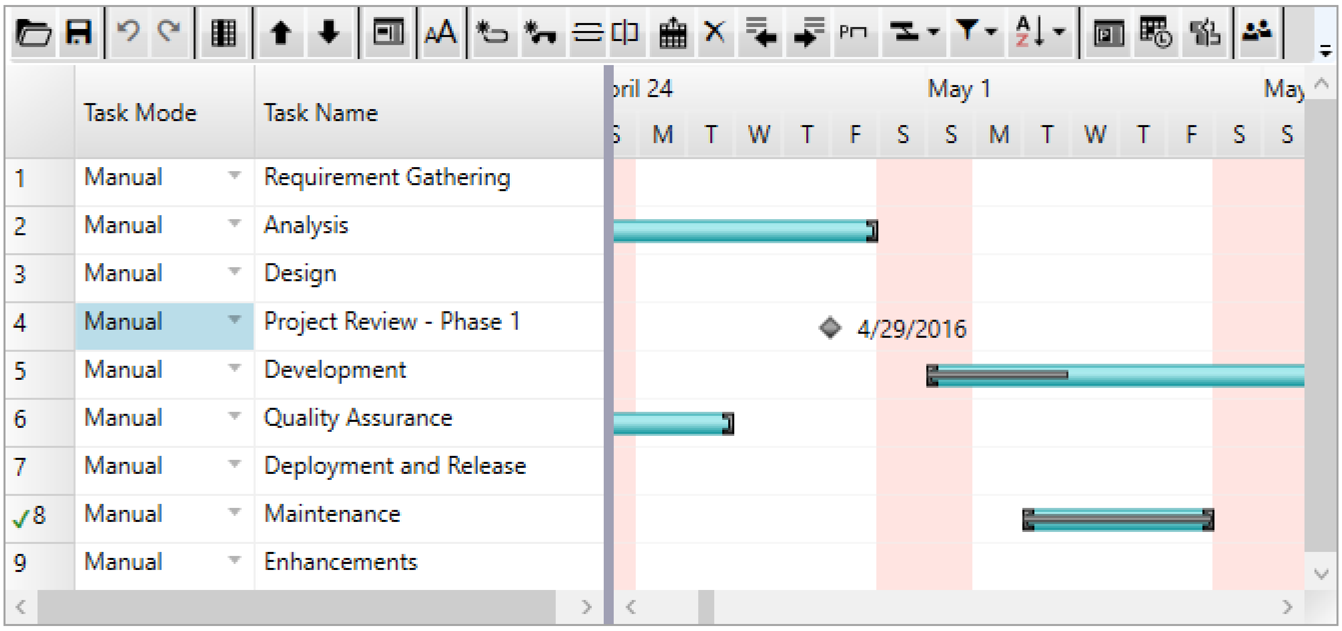
Format	Description
:	Use DateTimeFormatInfo.TimeSeparator
/	Use DateTimeFormatInfo.DateSeparator
"	Represents a quoted string (quotation mark)
'	Represents a quoted string (apostrophe)
\c	Displays the character 'c' as a literal
Other character	Copies to the result string

Milestone Task

A milestone task, or simply a milestone, is used for accounting an important event(s) within a project schedule. This feature helps users in keeping a track of their planned goals. For example, you can create a milestone to mark events such as the completion of a major phase in project schedule or indicate the beginning of the next project phase, etc. Since milestones do not include or account for any work, these are created as tasks with zero duration.

GanttView supports milestones, which can be created as a task with the Duration property set to zero, and the Start and the Finish properties set to the same date. Milestone tasks can also be created at the design-time through the Task Collection Editor available in the properties window. In GanttView, a milestone task is represented by a diamond-shaped item in the chart view.

The following image shows a GanttView displaying a milestone task by the name "Project Review - Phase 1".



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Project Review - Phase 1
5	Manual	Development
6	Manual	Quality Assurance
7	Manual	Deployment and Release
✓8	Manual	Maintenance
9	Manual	Enhancements

Creating a milestone in code

To create a milestone task, you need to define an instance of the Task class, and set the Duration, Start and Finish properties in code. The following example uses the sample created in the Quick Start.

Visual Basic

```
'Creating a milestone task
Dim t As New Task()
t.Name = "Project Review - Phase 1"
t.Start = New DateTime(2016, 4, 29)
t.Finish = New DateTime(2016, 4, 29)
t.Duration = 0

'Adding milestone task to gantt view
gv.Tasks.Add(t)
```

C#

```
//Creating a milestone task
Task t = new Task();
t.Name = "Project Review - Phase 1";
t.Start = new DateTime(2016, 4, 29);
t.Finish = new DateTime(2016, 4, 29);
t.Duration = 0;

//Adding milestone task to gantt view
gv.Tasks.Add(t);
```

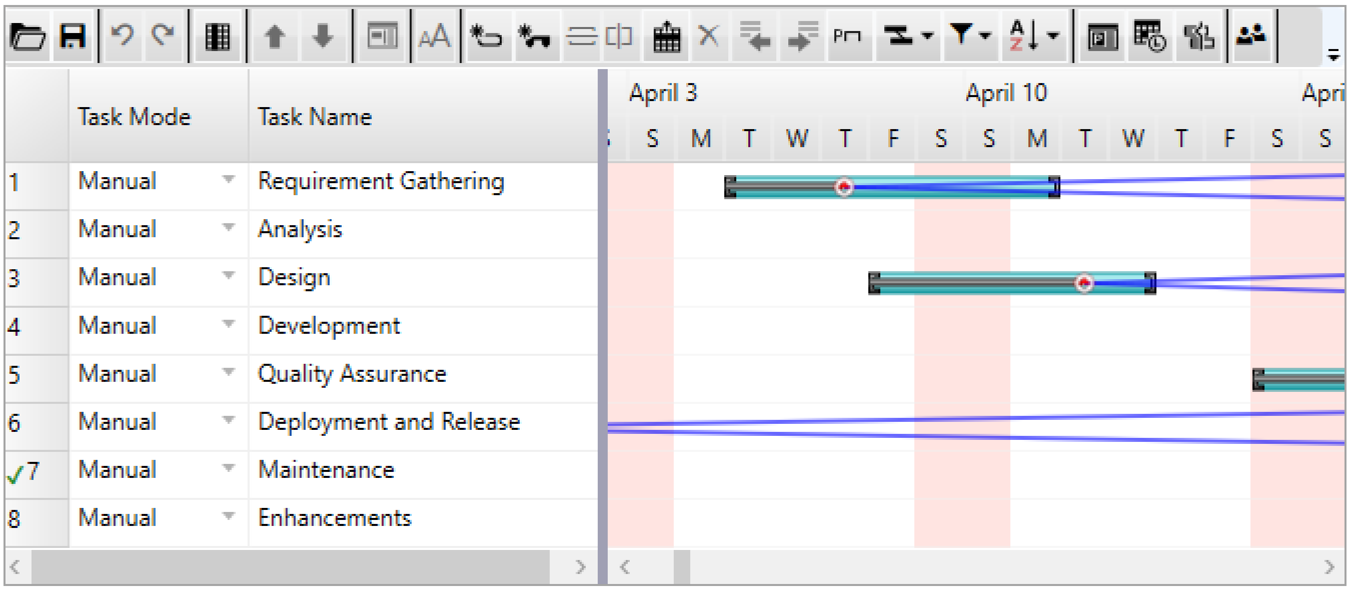
Progress Line

Progress lines are vertical lines drawn to indicate whether a task is behind or ahead of the project schedule with respect to a particular status date. A progress line connects the status date to the point on the task bar that represents its percentage complete. For example, if a task is 50% complete, the progress line connects the status date to the middle of the task bar. A progress line creates a spike either towards the left or right. A progress line that spikes out towards the left indicates that the task is behind the schedule, while a spike towards the right indicates that the task is ahead of the schedule.

Being a project planning and scheduling control, GanttView also supports progress line to indicate a task's status at any given point of time. You can create a progress line at runtime through the **Progress Line** button available on the toolbar. Clicking the button opens the Progress Line dialog, which can be used to select the status date, enable or disable progress line, and customize various elements of the progress line such as its color, shape, and style.

GanttView supports a single progress line at a given instance of time.

The image below shows progress line drawn in GanttView.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Creating a progress line in code

The GanttView control provides the ProgressLine class, which includes various properties for managing progress lines. For example, the class provides the Visible property that accepts Boolean values to enable or disable the visibility of progress line, the StatusDate property to set the status date, the LineColor property to set the color of the progress line, and the LineStyle property to specify the line type. For further customization, the PointColor and PointShape properties can be used to set the color and shape of the point connecting the task bar and the status date.

The following code example illustrates creating and customizing a progress line. This example uses the sample created in the Quick Start.

Visual Basic

```
'Creating a progress line
gv.ProgressLine.Visible = True
gv.ProgressLine.StatusDate = New DateTime(2016, 4, 29)
gv.ProgressLine.LineColor = Colors.Blue
```

C#

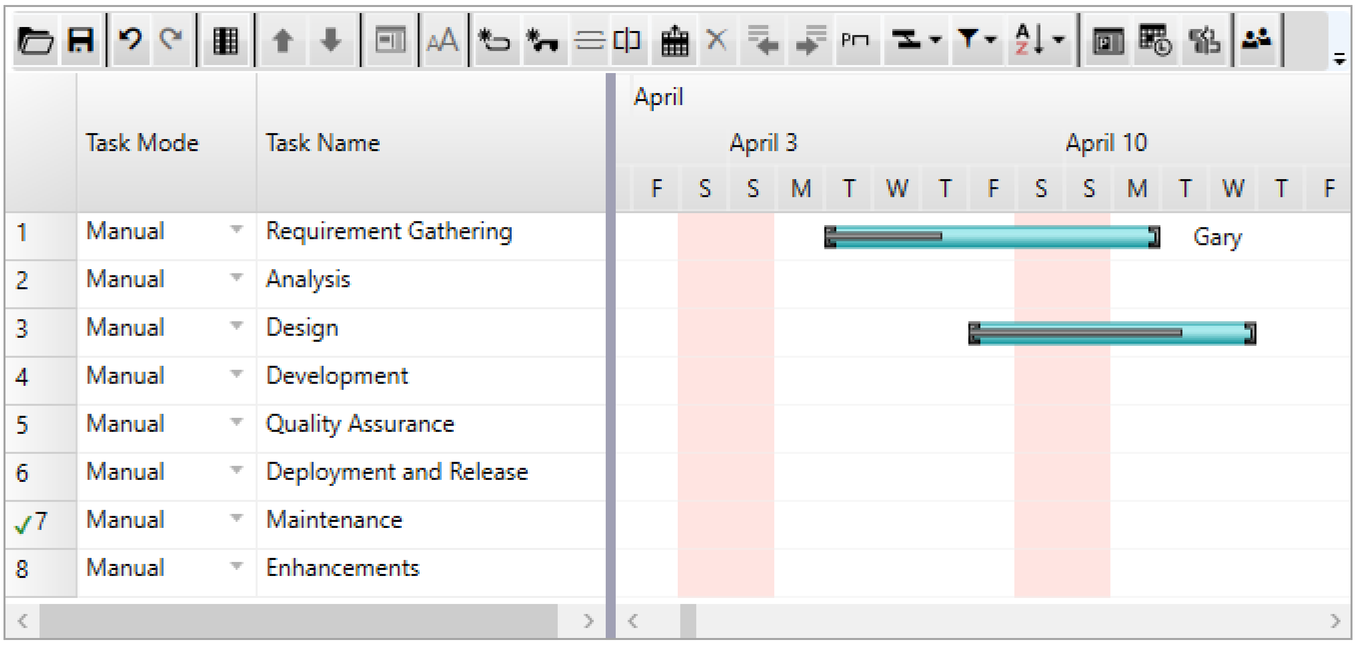
```
//Creating a progress line
gv.ProgressLine.Visible = true;
gv.ProgressLine.StatusDate = new DateTime(2016, 4, 29);
gv.ProgressLine.LineColor = Colors.Blue;
```

Resources

GanttView supports three kinds of resources namely work resource, material resource, and cost resource described further as follows:

Resource Type	Description
Work Resource	Refers to the people and equipment required to accomplish a set of tasks defined in a project.
Material Resource	Refers to the exhaustible raw material and goods required to accomplish a set of task in a project
Cost Resource	Refers to the financial cost associated to a task that needs to be accounted for in a project such as expenditures on travel, entertainment, and other overheads.

The following image shows a resource assigned to a task in GanttView.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Creating, adding and assigning resources to tasks

GanttView provides the Resource class for handling resources. To create and manage resources, you need to create an instance of the Resource class and set its name and cost using the Name and Cost properties, respectively. The ResourceType property can be set to specify the type of resource. The ResourceType property accepts the following values from the ResourceType enumeration:

- **Work** - Specifies a work resource.
- **Material** - Specifies a material resource.
- **Cost** - Specifies a cost resource.

The TaskResources class also provides the Add method to add the resources to a locally-defined resource dictionary. However, simply creating a resource is of no use until it is assigned to a task for which the control provides the ResourceRef class. You need to create an object of the ResourceRef class, set its Resource property, and add the same to the task using the Add method.

The following code example illustrates creating, adding, and assigning resource to a task. This example uses the sample created in the Quick Start.

Visual Basic

```
'Creating a resource
Dim r As New Resource()
r.ResourceType = ResourceType.Work
r.Name = "Gary"
r.Cost = 300D

'Adding the resource to gantt view
gv.TaskResources.Add(r, "Gary")

'Assigning the resource to task 't1'
Dim t1 As Task = gv.Tasks.Search("Requirement Gathering")
If t1 IsNot Nothing AndAlso t1.ResourceRefs.Count = 0 Then
    Dim rRef As New ResourceRef()
    rRef.Resource = r
    rRef.Amount = 0.5
    t1.ResourceRefs.Add(rRef)
End If
```

C#

```
//Creating a resource
Resource r = new Resource();
r.ResourceType = ResourceType.Work;
r.Name = "Gary";
r.Cost = 300m;

//Adding the resource to gantt view
gv.TaskResources.Add(r, "Gary");

//Assigning the resource to task 't1'
Task t1 = gv.Tasks.Search("Requirement Gathering");
if (t1 != null && t1.ResourceRefs.Count == 0)
{
    ResourceRef rRef = new ResourceRef();
    rRef.Resource = r;
    rRef.Amount = 0.5;
    t1.ResourceRefs.Add(rRef);
}
```

Constraints

Constraints define the degree or extent to which a task can be rescheduled in a project schedule. This feature helps users in scheduling automatic tasks on the basis of various factors such as start and finish date, type of constraint, task priority and duration. Constraints cannot be applied to manual tasks. This is because GanttView does not schedule manual tasks, so applying constraints on them has no effect.

You can add constraints in GanttView to schedule automatic tasks by setting the ConstraintDate and ConstraintType properties (available in the Task class) for a particular task. The ConstraintDate property specifies the start or finish date for the applied constraint, while ConstraintType property specifies the type of constraint. The ConstraintType property accepts the following values from the ConstraintType enumeration:

Enumeration Value	Description
Default	Allows you to schedule the task as soon as possible if the project is scheduled from the start date; Or schedules the task as late as possible if the project is scheduled from the finish date.

FinishNoEarlierThan	Allows you to schedule tasks in a way that they must finish on or after the defined constraint date.
FinishNoLaterThan	Allows you to schedule tasks in a way that they must finish on or before the defined constraint date.
MustFinishOn	Allows you to schedule tasks such that they must finish on the defined constraint date.
MustStartOn	Allows you to schedule tasks such that they must start on the defined constraint date.
StartNoEarlierThan	Allows you to schedule tasks such that they must start on or after the constraint date.
StartNoLaterThan	Allows you to schedule tasks such that they must start on or before the constraint date.

Applying constraints in code

To apply constraints, you need to set the ConstraintDate and ConstraintType properties in code. The following example uses the sample created in the Quick Start.

Visual Basic

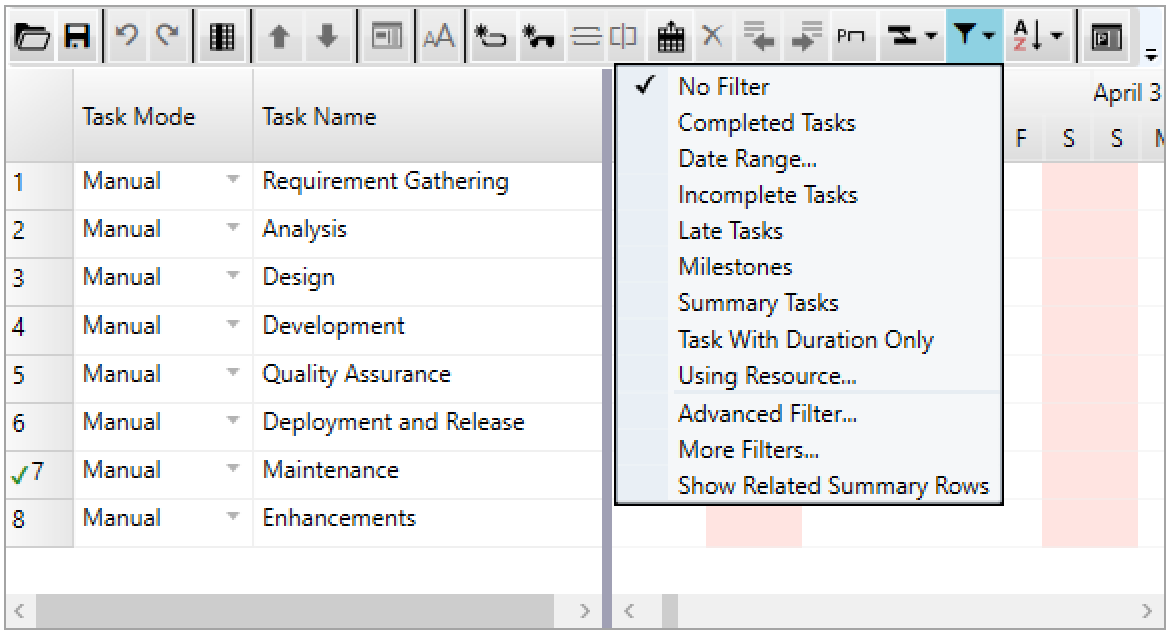
```
'Applying constraints
t.ConstraintDate = New DateTime(2016, 5, 5)
t.ConstraintType = ConstraintType.MustFinishOn
```

C#

```
//Applying constraints
t.ConstraintDate = new DateTime(2016, 5, 5);
t.ConstraintType = ConstraintType.MustFinishOn;
```

Filtering

GanttView supports filtering to help users view specific tasks or resources at runtime. The control lets you apply filters to view complete, incomplete and late tasks, milestones, tasks within a specified time period, tasks using a particular resource. etc. The control provides various filtering options under the **Filter** menu in GanttView Toolbar.


	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

The following table explains the sorting options available in the GanttView.

Filtering Options	Description
No Filter	Lets you remove any existing filter.
Completed Tasks	Lets you filter and display completed tasks.
Date Range	Lets you filter and display tasks within a specified time period.
Incomplete Tasks	Lets you filter and display incomplete tasks.
Late Tasks	Lets you filter and display late tasks.
Milestones	Lets you filter and display milestones in your project schedule.
Tasks with Duration Only	Lets you filter and display tasks with duration only.
Using Resources	Lets your filter tasks assigned to a particular resource.
Advanced Filter	Lets you create and apply custom filters through Advanced dialog.
More Filters	Lets you create new custom filters.

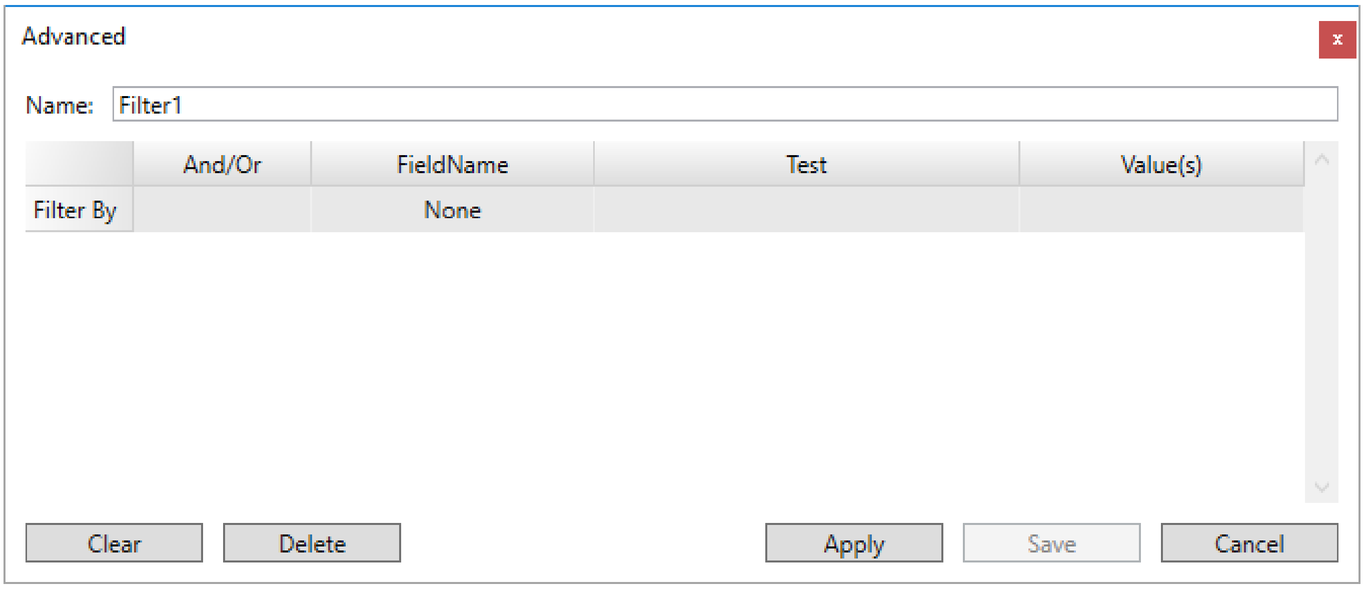
GanttView also allows users to create custom filters according to their requirements. Click the following link to know more about setting custom filters.

- Custom Filtering

Custom Filtering

Besides the default filtering options that you get on the toolbar, the GanttView control also enables users create custom filters to suit their business needs. To achieve this, the control provides Advanced filter dialog, which is available under the **Filter** button on the toolbar. You can create your own filters by specifying fields in the FieldName and test condition in the Test field.

The following image shows how the Advanced filter dialog appears.



	And/Or	FieldName	Test	Value(s)
Filter By		None		

Creating Custom Filters in Code

The C1GanttView class library includes various classes that can be used to create custom filters in code. For example, you can use the LateTasksFilter class to filter late tasks, or use ConditionTasksFilter and AdvancedFilter classes to filter tasks scheduled within a specific date range. Complete the following steps to create such filters in code. The following code example uses the sample created in the Quick Start section.

1. In the XAML view, resize the GanttView control placed onto the MainWindow, and place two standard button controls to the form.
2. Set the basic properties of the buttons in the XAML.

XAML

```
<Button Content="Filter Tasks by Date" Width="110" />
<Button Content="Filter Late Tasks" Width="114"/>
```

3. Subscribe the Click events for the two buttons in the XAML view.

XAML

```
<Button Content="Filter Tasks by Date" Width="110" Click="Button1_Click" />
<Button Content="Filter Late Tasks" Width="114" Click=Button2_Click"/>
```

4. Add the following code to the handler for Button1_Click event in the MainPage.xaml.cs file.

Visual Basic

```
Private Sub Button1_Click(sender As Object, e As RoutedEventArgs)
    Dim filter As New AdvancedFilter()
    ' Filter the tasks starting on April 8th, 2016...
    Dim startCondition As New ConditionTaskFilter()
    startCondition.FilterField = FilterField.Start
    startCondition.TestOperator = TestOperators.IsGreaterThanOrEqualTo
    startCondition.FilterValue = New DateTime(2016, 4, 8)
    filter.Conditions.Add(startCondition)

    ' Filter tasks finishing before or on May 10th, 2016...
    Dim finishCondition As New ConditionTaskFilter()
    finishCondition.FilterField = FilterField.Finish
    finishCondition.TestOperator = TestOperators.IsLessThanOrEqualTo
    finishCondition.FilterValue = New DateTime(2016, 5, 10)
    filter.Conditions.Add(finishCondition)
    gv.ApplyFilter(filter)
```

```
End Sub
```

C#

```
private void Button1_Click(object sender, RoutedEventArgs e)
{
    AdvancedFilter filter = new AdvancedFilter();
    // Filter the tasks starting on April 8th, 2016...
    ConditionTaskFilter startCondition = new ConditionTaskFilter();
    startCondition.FilterField = FilterField.Start;
    startCondition.TestOperator = TestOperators.IsGreaterThanOrEqualTo;
    startCondition.FilterValue = new DateTime(2016, 4, 8);
    filter.Conditions.Add(startCondition);

    // Filter tasks finishing before or on May 10th, 2016...
    ConditionTaskFilter finishCondition = new ConditionTaskFilter();
    finishCondition.FilterField = FilterField.Finish;
    finishCondition.TestOperator = TestOperators.IsLessThanOrEqualTo;
    finishCondition.FilterValue = new DateTime(2016, 5, 10);
    filter.Conditions.Add(finishCondition);
    gv.ApplyFilter(filter);
}
```

The above code filters tasks starting on 8-April-2016 and finishing before or on 10-May-2016.

5. Add the following code to the handler for Button2_Click event in the MainPage.xaml.cs file.

Visual Basic

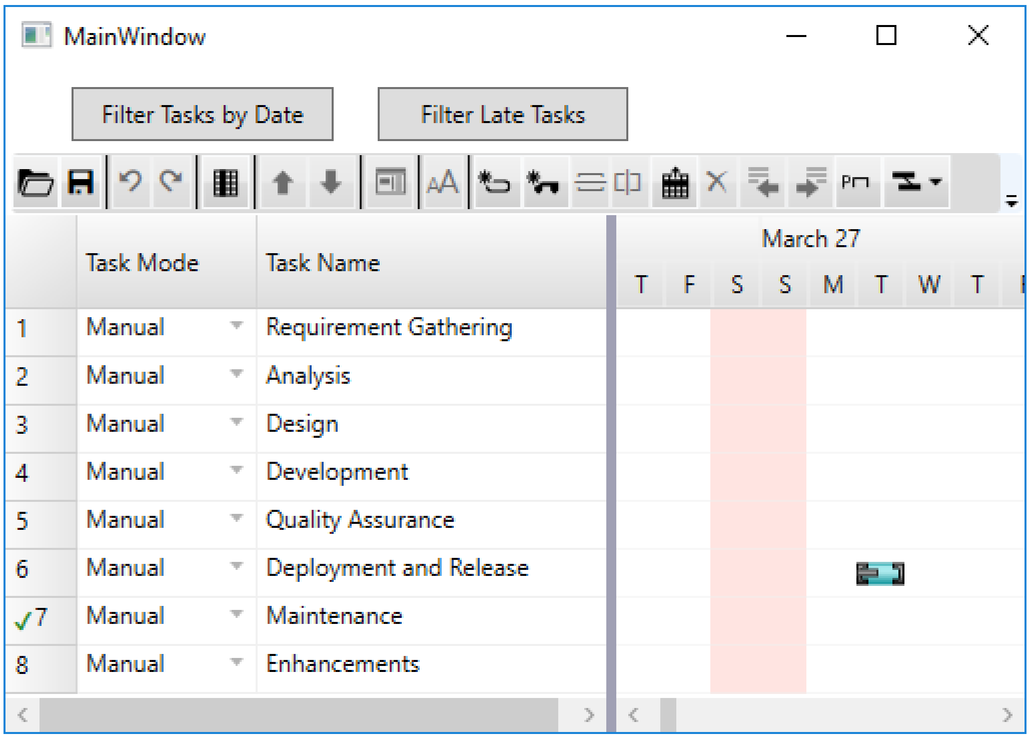
```
Private Sub Button2_Click(sender As Object, e As RoutedEventArgs)
    Dim filter As New LateTasksFilter(New DateTime(2016, 5, 10))
    gv.ApplyFilter(filter)
End Sub
```

C#

```
private void Button2_Click(object sender, RoutedEventArgs e)
{
    LateTasksFilter filter = new LateTasksFilter(new DateTime(2016, 5, 10));
    gv.ApplyFilter(filter);
}
```

The above code filters tasks whose start date is 10-May-2016 or beyond.

6. Press F5 to run the application to see how the custom filter buttons get displayed in the output.

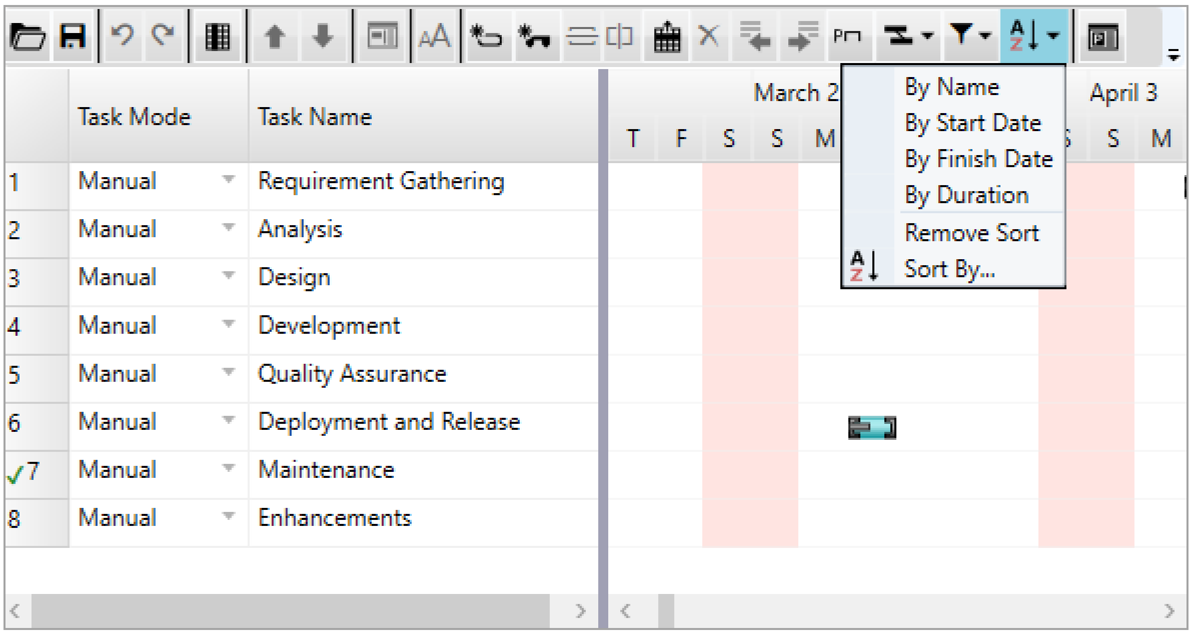


	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

On clicking the **Filter Tasks by Date** button, the GanttView displays tasks starting on 8-April-2016 and finishing before or on 10-May-2016, while on clicking the **Filter Late Tasks**, tasks with start date 10-May-2016 or beyond get displayed.

Sorting

Sorting is an important requirement when it comes to data management, helping users view data in a specified order. The GanttView control lets you sort tasks by task name, start/finish date, and duration through various sorting options available under the **Sort** button in the GanttView Toolbar. The **Sort** menu provides **Sort By** option that helps users achieve multicolumn sorting in tasks.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

The following table explains the sorting options available in the GanttView.

Sort Option	Description
By Name	Lets you sort tasks by their name in alphabetical order.
By Start Date	Lets you sort the tasks by their start date.
By Finish Date	Lets you sort the tasks by their finish date.
By Duration	Lets you sort the tasks by the time duration assigned to each task.
Remove Sort	Lets you remove any existing sort condition.
Sort By	Lets you achieve multicolumn sorting.

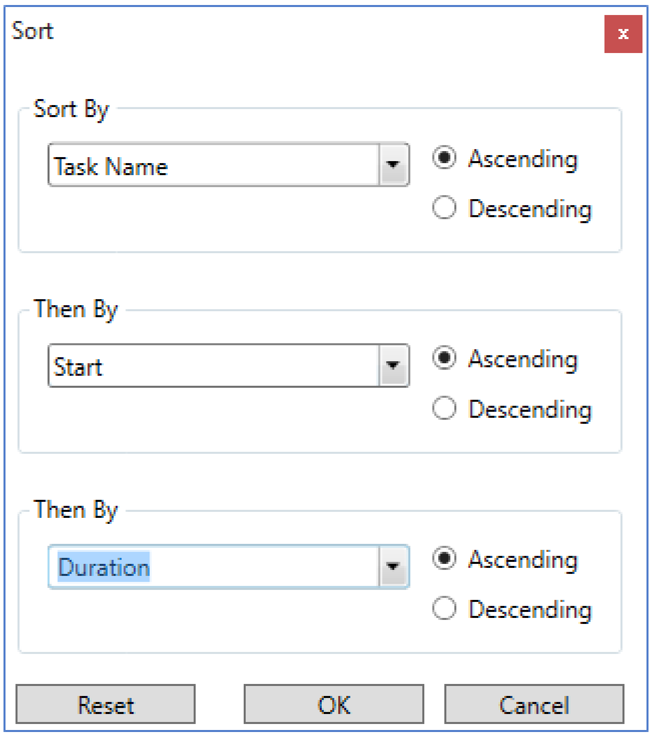
By default, all the sorting options available under the Sort button display data in 'ascending' order. However, you can change the default sort order to 'descending' by using Sort dialog.

You can also achieve multicolumn sorting in GanttView. To know more, see Multicolumn sorting.

Multicolumn Sorting

GanttView allows you to achieve multicolumn sorting through the **Sort By** option available under the **Sort** button on the toolbar. To achieve multicolumn sorting, GanttView provides Sort dialog that lets you set the sort order in multiple columns at runtime.

The following image shows a Sort dialog that creates a sort order on three different columns, namely Task Name, Task Mode and Duration.


Sort

Sort By: Task Name — [✓ Ascending] Descending

Then By: Start — [✓ Ascending] Descending

Then By: Duration — [✓ Ascending] Descending

Saving Content in XML

GanttView allows users to save the contents along with the layout into an XML file. This feature is available to users at runtime through the **Save** button available on the toolbar. Clicking the **Save** button opens the **Save As XML** dialog that lets you save the contents into an XML file.

Saving Content into XML in Code

The functionality to save content into an XML file can also be achieved programmatically through the SaveXml method of the C1GanttView class. The SaveXml method provides three overloads for saving the contents as listed below.

Overloads	Description
SaveXml (String)	To save the contents into an XML file.
SaveXml (Stream)	To save the contents into an IO stream object.
SaveXml (StreamWriter)	To save the contents into an XmlWriter object.

The following code illustrates how the SaveXml method can be used to save the contents into an XML file.

Visual Basic

```
Dim dlg As New OpenFileDialog()
dlg.DefaultExt = ".xml"
dlg.FileName = "gantt"
dlg.Filter = "XML files|*.xml|All files|*.*"
dlg.Title = "Save Gantt View As Xml File"
If dlg.ShowDialog().Value Then
        gv.SaveXml(dlg.FileName)
End If
```

C#

```
OpenFileDialog dlg = new OpenFileDialog();
dlg.DefaultExt = ".xml";
dlg.FileName = "gantt";
dlg.Filter = "XML files|*.xml|All files|*.*";
dlg.Title = "Save Gantt View As Xml File";
if (dlg.ShowDialog().Value)
 {
       gv.SaveXml(dlg.FileName);
 }
```

Loading Content from XML

GanttView allows users to load the contents along with the layout from an XML file. This feature is available to users at runtime through the **Load** button available on the toolbar. Clicking the Load button opens the **Load from XML** file dialog that lets you load the XML file. In addition, GanttView provides ImportFromMsProjectXml method to load project schedules from Microsoft Project.

Loading Content from XML in Code

The functionality to load content from an XML file can also be achieved programmatically through the LoadXml method of the C1GanttView class. The LoadXml method provides three overloads for loading the contents as listed below.

Overloads	Description

LoadXml (String)	To load the contents from an XML file.
LoadXml (Stream)	To save the contents from an IO stream object.
LoadXml (StreamWriter)	To save the contents from an XmlWriter object.

The following code illustrates how the LoadXml method can be used to load the contents from an XML file.

Visual Basic

```
Dim dlg As New OpenFileDialog()
dlg.DefaultExt = ".xml"
dlg.Filter = "XML files|*.xml|All files|*.*"
dlg.Title = "Load Gantt View From Xml File"
If dlg.ShowDialog().Value Then
        gv.LoadXml(dlg.FileName)
Else
        MessageBox.Show("Bad C1GanttView XML.", dlg.Title)
End If
```

C#

```
OpenFileDialog dlg = new OpenFileDialog();
dlg.DefaultExt = ".xml";
dlg.Filter = "XML files|*.xml|All files|*.*";
dlg.Title = "Load Gantt View From Xml File";
if (dlg.ShowDialog().Value)
{
  gv.LoadXml(dlg.FileName);
}
else
{
  MessageBox.Show("Bad C1GanttView XML.", dlg.Title);
}
```

Task Dependency

Tasks created in a project plan are often linked together to create a feasible scheduling relationship. This is called task dependency or dependency. For example, a software development project encompasses various tasks including requirement gathering, product analysis, design, development, quality assurance, release, etc. aligned in phases. In this scenario, every task is fairly dependent on the completion of the other. For instance, quality assurance is dependent on product development, while product development is dependent on requirement and product analysis. Tasks related to quality assurance can only be started once the product development is over. This creates a "finish-to-start" dependency, where product development becomes a predecessor task since it precedes the tasks that depend on it, while quality assurance becomes the successor task as it follows or succeeds the task(s) on which it depends.

Based on all possible relationships, any two tasks can have the following dependencies:

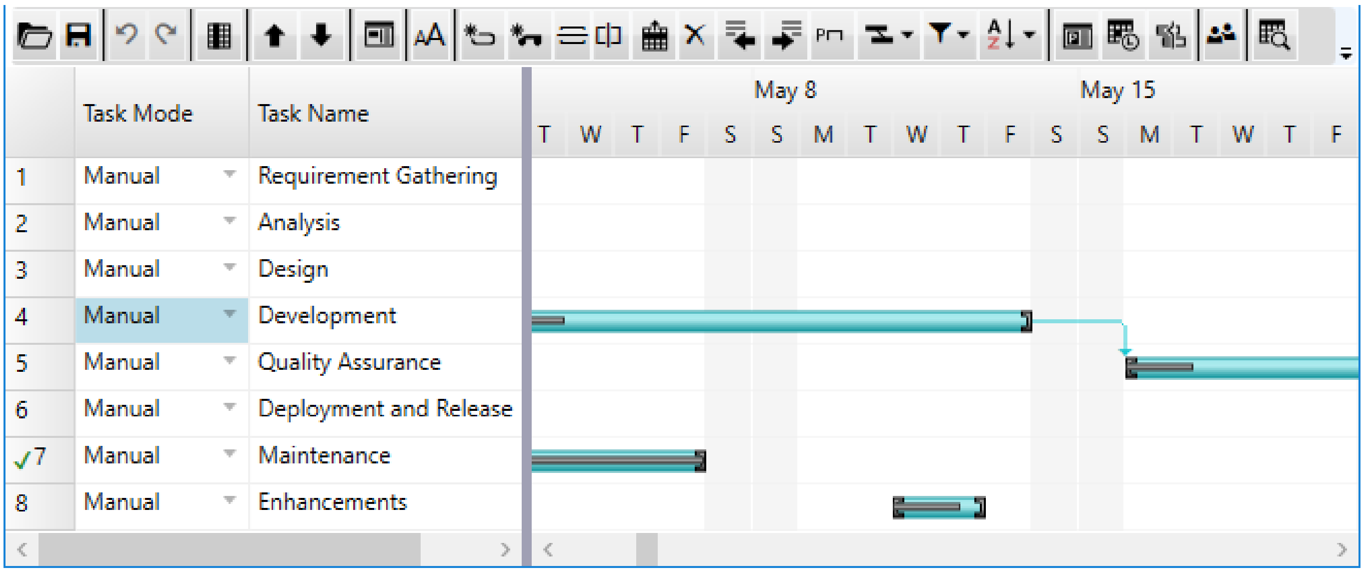
Dependency	**Description**
Finish-to-start	The finish date of the predecessor task determines the start date of the successor task.
Finish-to-finish	The finish date of the predecessor task determines the finish date of the

	successor task.
Start-to-finish	The start date of the predecessor task determines the finish date of the successor task.
Start-to-start	The start date of the predecessor task determines the start date of the successor task.

GanttView supports task dependencies both at runtime as well as in code. The Predecessors tab available in the Task information dialog can be used to create task dependencies at runtime. In addition, dependent tasks can be linked through mouse at runtime. Creating and handling task dependencies in code is explained later in the topic.

In GanttView, task dependencies can be visualized through lines connecting the dependent tasks. The grid area of the GanttView also displays predecessor tasks and dependencies in the **Predecessors Column**, provided its visibility is enabled. The Predecessors column displays the serial number of the predecessor task along with dependency type in abbreviated form (like SS, SF, or FF), except in case for default dependency, that is Finish-to-start.

The following image shows tasks dependencies created in a GanttView.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Creating task dependencies in code

To create and handle task dependencies, the GanttView control provides the Predecessor class, which can be used to specify the dependence of one or more tasks on the other. The class provides PredecessorTask and PredecessorType properties to set task precedence and dependency between tasks. A delay in days can also be set between the dependent tasks by using the Lag property. The PredecessorType property accepts the following values from the PredecessorType enumeration to indicate the type of dependence:

- **FinishToStart** - This is the default value that indicates that the dependent task cannot start until the predecessor task finishes.
- **FinishToFinish** - Indicates that the dependent task can finish only after the predecessor task has finished.
- **StartToFinish** - Indicates that the dependent task can finish only after the predecessor task has started.
- **StartToStart** – Indicates that the dependent task can start only after the parent task has started.

The following code example illustrates creating a predecessor task, and setting its dependence on another task and its type. This example uses the sample created in the Quick start.

Visual Basic

```
Dim task1 As Task = gv.Tasks.Search("Development")
Dim task2 As Task = gv.Tasks.Search("Quality Assurance")
```

```
If task1 IsNot Nothing AndAlso task2 IsNot Nothing AndAlso task2.Predecessors.Count
= 0 Then
    'switch to auto-scheduling mode
    task2.Mode = TaskMode.Automatic

    Dim p As New Predecessor()
    p.PredecessorTask = task1
    p.PredecessorType = PredecessorType.FinishToStart
    task2.Predecessors.Add(p)

    'restore the manual mode
    task2.Mode = TaskMode.Manual
End If
```

C#

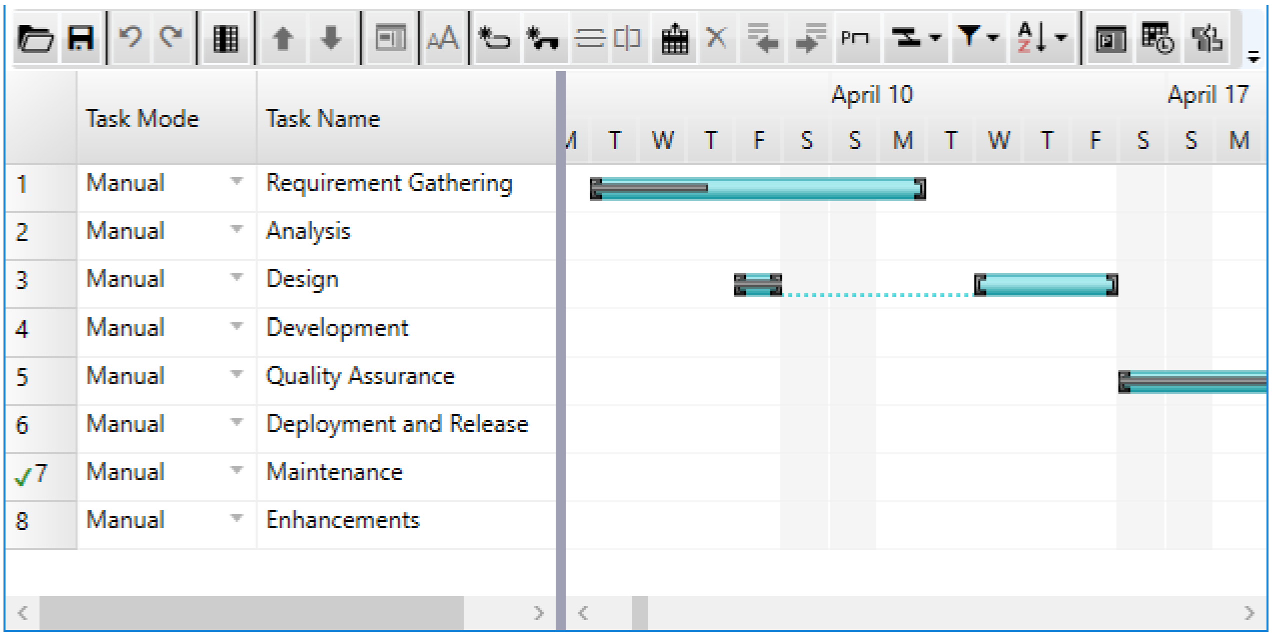
```
Task task1 = gv.Tasks.Search("Development");
Task task2 = gv.Tasks.Search("Quality Assurance");

if (task1 != null && task2 != null && task2.Predecessors.Count == 0)
{
    //switch to auto-scheduling mode
    task2.Mode = TaskMode.Automatic;

    Predecessor p = new Predecessor();
    p.PredecessorTask = task1;
    p.PredecessorType = PredecessorType.FinishToStart;
    task2.Predecessors.Add(p);

    //restore the manual mode
    task2.Mode = TaskMode.Manual;
}
```

Task Splitting

GanttView provides task splitting feature to account for unanticipated work interruptions because of various factors, such as, addition of new tasks in existing plan, unavailability of resources, change in task priority or unplanned leaves. Using this feature, user can easily put a task on hold in the middle so that a part of it can be started later or another high priority task can be taken up. In addition, user can also split a task as many times according to the requirement.

GanttView enables users to split task at runtime as well as through code. The **Split Task** button available in the toolbar splits a task into two or more segments.

The following image shows task split into two segments in a GanttView.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

To split a task at runtime

1. In the toolbar, click the **Split Task** button once.
   A screen tip appears, and the mouse pointer changes.
2. Move the mouse pointer over the task bar of the task to be split.
3. Move the mouse pointer to the date where you want the split segment to start.
4. Click the mouse pointer at the desired start date.
5. Drag the split segment to another date as per the schedule.

Note: The screen tip contains the date at which the split segment is scheduled to start. As you move the mouse pointer along the task bar, the scheduled start date in the screen tip changes.

A task can also be split at runtime by right clicking a specific point (date) on the task bar and selecting the **Split Task** option from the context menu. The **Split Task** option splits the task into two segments and adds a gap of one day in between. The split task segments can then be dragged to another date as per the user requirement.

To split a task through code

A task can be split through code by invoking the **SplitTask** method of the Task class. You can specify the start date of the task segment and the interruption duration in terms of number of days in the method to split a task. The following example illustrates splitting a task through code. This example uses the sample created in the Quick start.

Visual Basic

```
'Splitting a task into two segments
gv.Tasks(2).SplitTask(gv.Tasks(2).Start.Value.AddDays(2), 2)
```

C#

```
//Splitting a task into two segments
gv.Tasks[2].SplitTask(gv.Tasks[2].Start.Value.AddDays(2), 2);
```

Limitations:

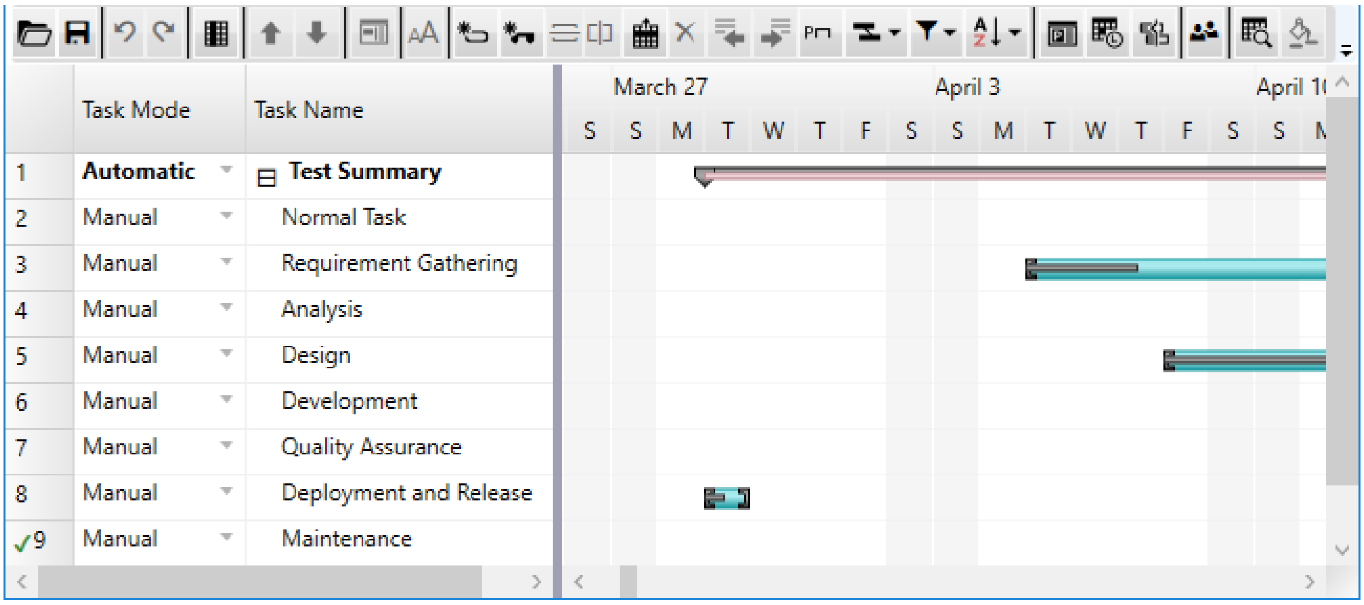
By design, the task splitting feature of GanttView does not apply in the following conditions:

1. If the split segment is scheduled to begin before the start date, or after the finish date of the task.
2. If the duration of the task to be split is less than or equal to 1 day.
   If the DurationUnit is 'Days' and the duration is less than 1 day, then the task cannot be split.
   If the DurationUnit is 'Hours' and the duration is less than 24 hours, then the task cannot be split.
3. If the main task begins on a nonworking day, then the split segment should begin on a working day.

Project Summary Task

Project summary task represents the summary of tasks indented below it, which are called subtasks or child tasks. You can create summary tasks in GanttView to better organize tasks into phases and depict a clear outline of the project plan. A summary task must contain at least one child task. By default, a project summary task is always automatically scheduled since its duration depends on the duration of its subtasks. This means the duration of a project summary task gets automatically adjusted on the basis of the earliest start date and the latest finish date of the underlying subtasks.

The following image shows a project summary task in GanttView.



	Task Mode	Task Name
1	**Automatic**	**Test Summary**
2	Manual	Normal Task
3	Manual	Requirement Gathering
4	Manual	Analysis
5	Manual	Design
6	Manual	Development
7	Manual	Quality Assurance
8	Manual	Deployment and Release
✓9	Manual	Maintenance

Creating summary tasks

To create a summary task above a subtask, the toolbar provides **Add Summary Task** button. In order to view the overall project summary, click the **Show Project Summary** button available in the toolbar.

In code, summary tasks are created just like any other task and assigning an ID to them by setting the ID property. To create a subtask of a particular summary task, assign the ID of the project summary task to the OutlineParentID property of the subtask.

The following example illustrates creating summary task in code. This example uses the sample created in the Quick start.

Visual Basic

```
'Creating project summary task
Dim summary As New Task()
summary.Name = "Test Summary"
summary.ID = 1
summary.Mode = TaskMode.Automatic

Dim subtask1 As New Task()
```

```
subtask1.Name = "Normal Task"
subtask1.ID = 2
subtask1.Mode = TaskMode.Manual
subtask1.OutlineParentID = 1
subtask1.Start = DateTime.Today
subtask1.Duration = 5

gv.Tasks.Add(summary)
gv.Tasks.Add(subtask1)
```

C#

```
//Creating project summary task
Task summary = new Task();
summary.Name = "Test Summary";
summary.ID = 1;
summary.Mode = TaskMode.Automatic;

Task subtask1 = new Task();
subtask1.Name = "Normal Task";
subtask1.ID = 2;
subtask1.Mode = TaskMode.Manual;
subtask1.OutlineParentID = 1;
subtask1.Start = DateTime.Today;
subtask1.Duration = 5;

gv.Tasks.Add(summary);
gv.Tasks.Add(subtask1);
```

Style and Appearance

GanttView provides various options to customize the default appearance of its elements such as task bars, fields, as well as the control itself. Learn about these features and how to implement them in code.

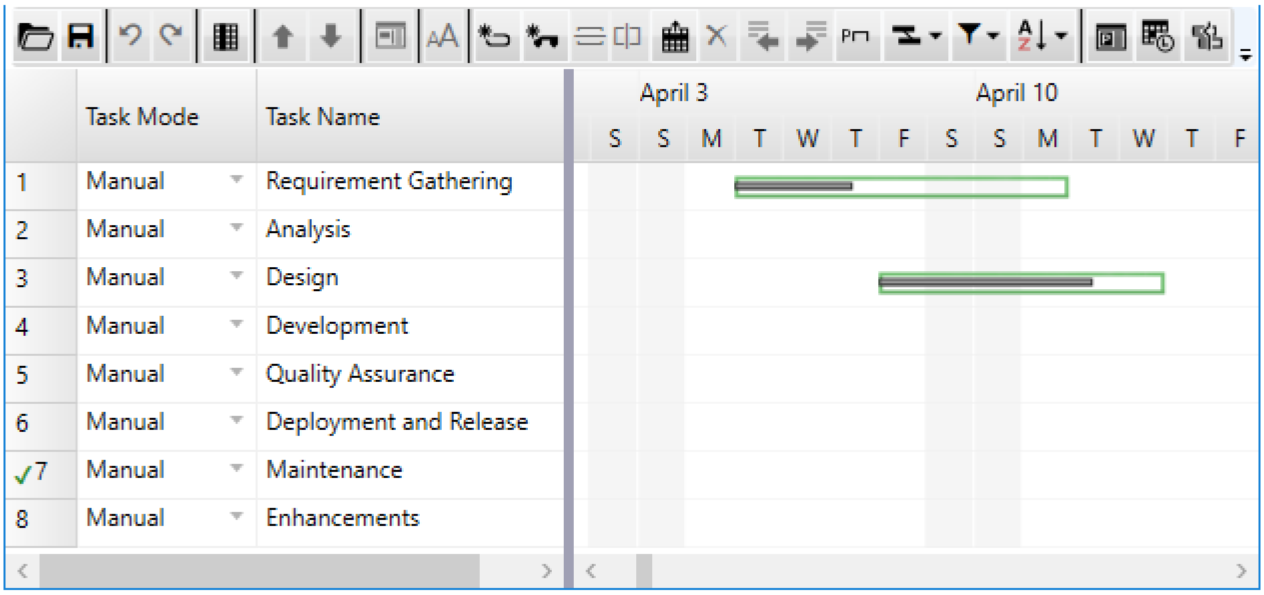
Custom bar styles
    Learn how to customize task bars in code.
Custom field styles
    Learn how to apply custom styles to fields in code.
Applying themes
    Learn how to apply ComponentOne themes to GanttView in code.
Set back color of day of week
    Learn how to apply back color to a specific day(s) in code.

Custom Bar Styles

In GanttView, all the tasks in a project plan are represented through task bars, which depict attributes like start and finish date, duration, percent complete, current status, etc. The visual appeal of these bars can be enhanced by customizing their color, shape, and hatch style.

This is possible in GanttView through the BarStyle class, which includes various properties to set the background color, shape and hatch style of task bars. Task bars can also be customized at runtime through the Bar Styles dialog, which can be accessed by clicking the **Bar Styles** button available on the toolbar. In addition, the control lets you adjust the height of the task bar according to the font applied to GanttView. The C1GanttView class provides the AdaptiveBarHeight property, which if set to **true** automatically adjusts the height of the task bar according to GanttView's font size. You can also set the bar height explicitly by setting the AdaptiveBarHeight property to **false** and then setting the height using the BarHeight property.

The following image shows a GanttView with customized task bars.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

Properties available in the BarStyle class for customizing the task bars are as follows:

Property	Purpose
BarType	To specify the bar(s) on which the customization is to be applied.
BarColor	To set the background color of the task bar.
BarShape	To set the shape of the task bar.
BarPattern	To set the hatch style for filling the task bar.
EndPattern	To set the hatch style for filling the shape at the end of the task bar.

The following code illustrates how to customize task bars. This example uses the sample created in the Quick Start.

Visual Basic

```
'Creating an instance of bar style
Dim bs As New BarStyle()

'Applying customization to manual tasks only
bs.BarType = BarType.ManualTask
bs.BarColor = Colors.LightGreen
bs.BarShape = BarShape.Frame
bs.BarPattern = HatchPattern.Gradient
bs.EndPattern = HatchPattern.Empty

'Adding the bar style object to gantt view
gv.BarStyles.Add(bs)
```

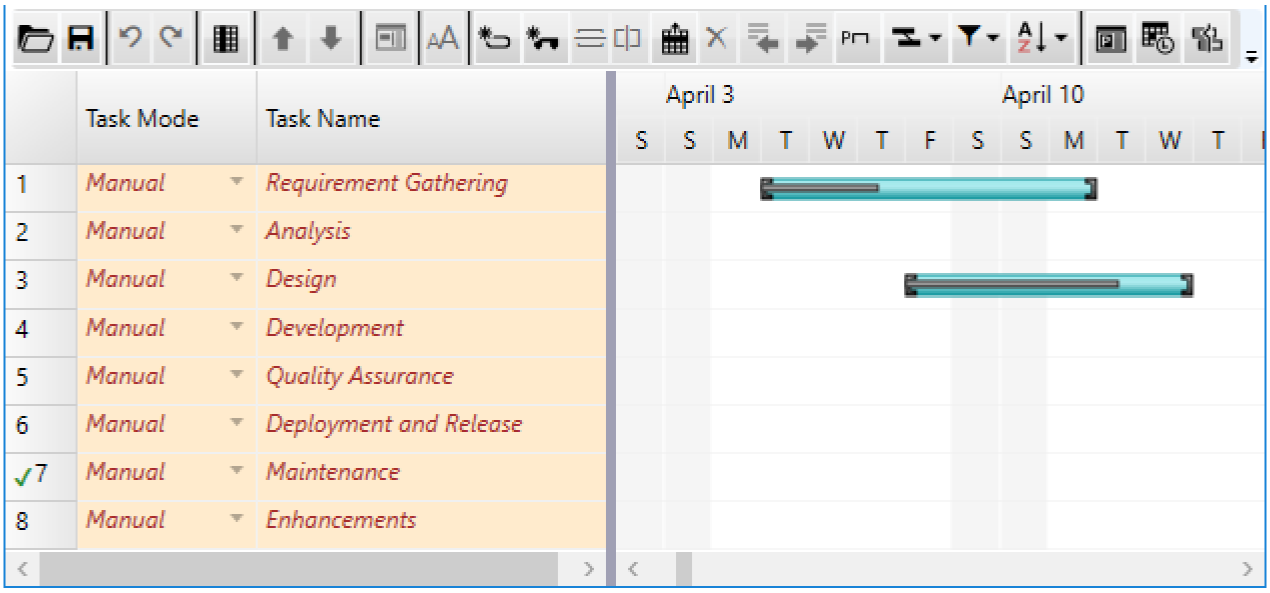
C#

```
//Creating an instance of bar style
BarStyle bs = new BarStyle();

//Applying customization to manual tasks only
bs.BarType = BarType.ManualTask;
bs.BarColor = Colors.LightGreen;
bs.BarShape = BarShape.Frame;
bs.BarPattern = HatchPattern.Gradient;
bs.EndPattern = HatchPattern.Empty;

//Adding the bar style object to gantt view
gv.BarStyles.Add(bs);
```

Custom Field Styles

GanttView provides users the ability to customize the style and appearance of tasks fields in order to match with the user interface requirements of an application. Users can format the fields in the grid view by using various properties of the FieldStyle class such as Field, FieldName, FontStyle, ForegroundColor, BackgroundColor, and Underline.

The following image shows a GanttView with custom styles applied to the fields.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

The following code illustrates how to apply custom styles to fields. This example uses the sample created in the Quick start.

Visual Basic

```
Dim fs As New FieldStyle()
fs.Field = StyleField.All
fs.ForegroundColor = System.Windows.Media.Colors.Brown
fs.BackgroundColor = System.Windows.Media.Colors.BlanchedAlmond
fs.FieldName = "Task Name"
fs.FontStyle = FontStyles.Italic
For Each task As Task In gv.Tasks
    Dim fsc As FieldStyleCollection = task.FieldStyles
    fsc.Add(fs)
Next
```

C#

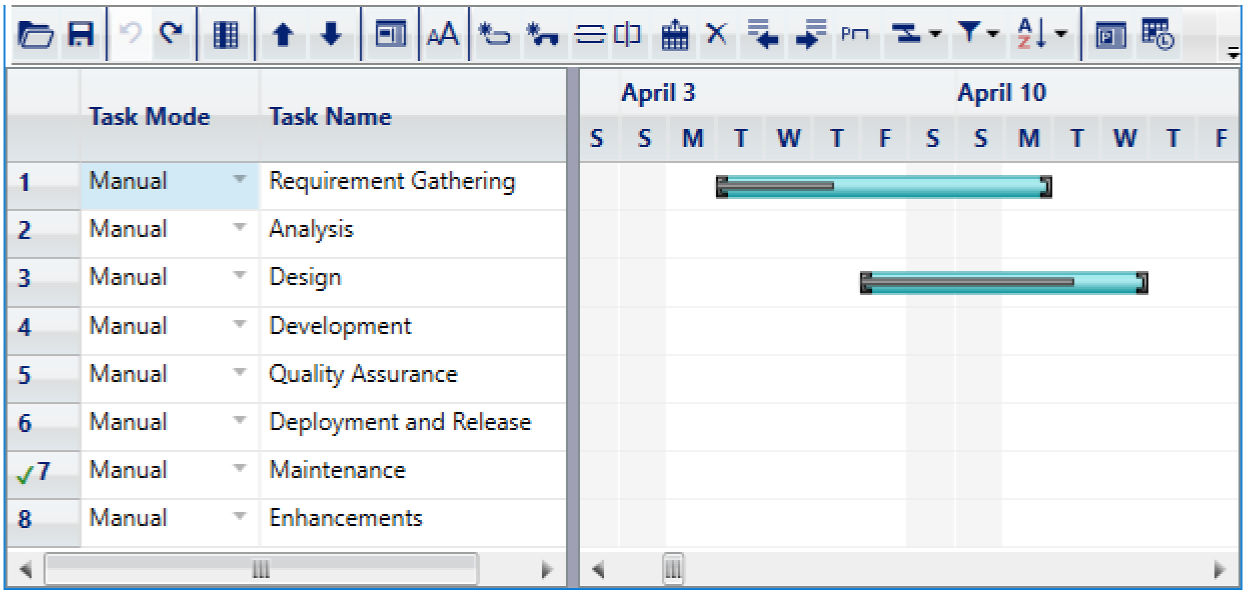
```
FieldStyle fs = new FieldStyle();
fs.Field = StyleField.All;
fs.ForegroundColor = System.Windows.Media.Colors.Brown;
fs.BackgroundColor = System.Windows.Media.Colors.BlanchedAlmond;
fs.FieldName = "Task Name";
fs.FontStyle = FontStyles.Italic;
```

```
foreach (Task task in gv.Tasks)
{
    FieldStyleCollection fsc = task.FieldStyles;
    fsc.Add(fs);
}
```

Apply Themes

GanttView supports built-in ComponentOne themes available in **C1.WPF.Theming** library. You can apply any of the available C1 themes to the GanttView control and provide a customized and consistent appearance to the application. To understand how themes can be used effectively, see Theming.

The following image shows a GanttView with **C1WhistlerBlue** theme applied to it.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

The following steps illustrate how to apply C1 themes to GanttView. This example uses the sample created in the Quick start.

1. Create a Quick start application and add the following .dll files to your application.
   - C1.WPF.Theming
   - C1.WPF.Theming.WhistlerBlue
2. Switch to the MainWindow.xaml.cs file and add the following import statements.

   Visual Basic

   ```
   Imports C1.WPF.Theming
   Imports C1.WPF.Theming.WhistlerBlue
   ```

   C#

   ```
   using C1.WPF.Theming;
   using C1.WPF.Theming.WhistlerBlue;
   ```

3. Create a class, MyTheme, to create an object of the C1Themes class.

   Visual Basic

   ```
   Public Class MyThemes
       Private Shared _myTheme As C1Theme = Nothing
       Public Shared ReadOnly Property MyTheme() As C1Theme
           Get
   ```

```
                If _myTheme Is Nothing Then
                    _myTheme = New C1ThemeWhistlerBlue()
                End If

                Return _myTheme
            End Get
        End Property
End Class
```

C#

```
public class MyThemes
{
    private static C1Theme _myTheme = null;
    public static C1Theme MyTheme
    {
        get
        {
            if (_myTheme == null)
            {
                _myTheme = new C1ThemeWhistlerBlue();
            }

            return _myTheme;
        }
    }
}
```

4. Add the following code in the loaded event to apply ComponentOne theme to the GanttView control.

Visual Basic

```
'Apply C1 theme to GanttView
C1Theme.ApplyTheme(gv, MyThemes.MyTheme)
```

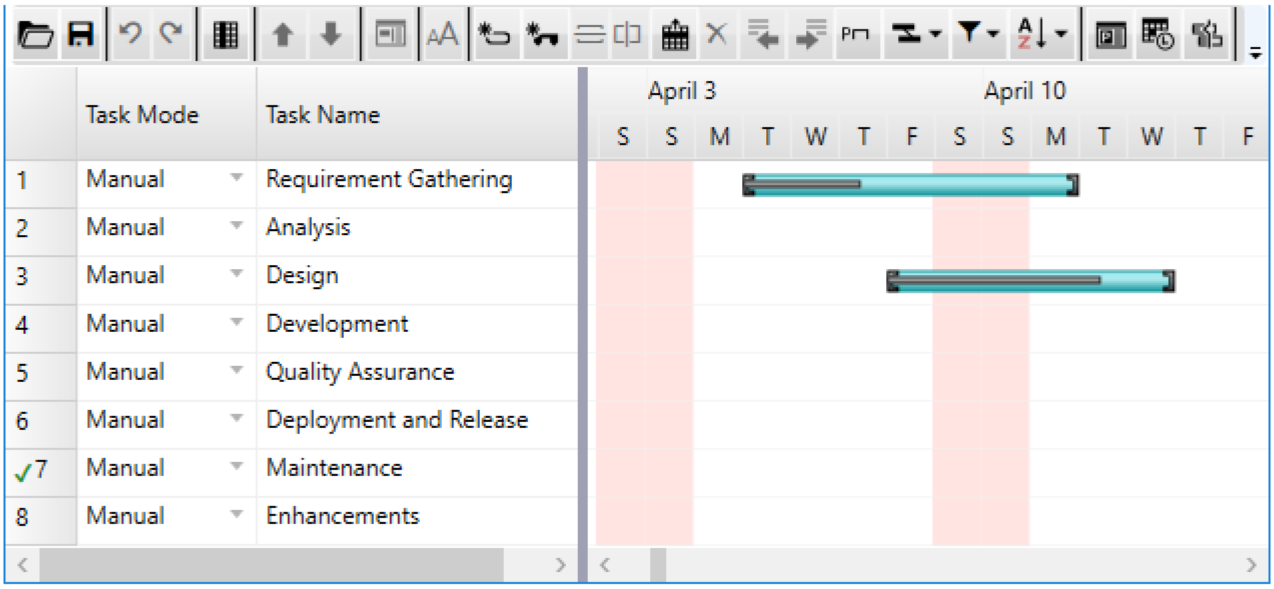
C#

```
//Apply C1 theme to GanttView
C1Theme.ApplyTheme(gv, MyThemes.MyTheme);
```

Set Back Color of Day

GanttView enables users to highlight a specific day of week by setting its back color. For instance, the weekends in a gantt view can be highlighted in a different color to distinguish them from workweek. This can be achieved in code by subscribing the PaintDay event and setting the BackColor property to color specific days.

The following image shows a GanttView with weekends highlighted by a different color.



	Task Mode	Task Name
1	Manual	Requirement Gathering
2	Manual	Analysis
3	Manual	Design
4	Manual	Development
5	Manual	Quality Assurance
6	Manual	Deployment and Release
✓7	Manual	Maintenance
8	Manual	Enhancements

The following code illustrates how to set back color of weekend. This example uses the sample created in the Quick start.

Visual Basic

```
Private Sub gv_PaintDay(sender As Object, e As PaintDayEventArgs)
    If e.[Date].DayOfWeek = DayofWeek.Saturday OrElse e.[Date].DayOfWeek =
DayofWeek.Sunday Then
        e.BackColor = Colors.MistyRose
    End If
End Sub
```

C#

```
//Set back color for a specific day of week
private void gv_PaintDay(object sender, PaintDayEventArgs e)
{
    if (e.Date.DayOfWeek == DayOfWeek.Saturday || e.Date.DayOfWeek ==
DayOfWeek.Sunday)
    {
        e.BackColor = Colors.MistyRose;
    }
}
```