



Essential Studio 2013 Volume 4 - v.11.4.0.26

## **Essential ProjIO**



# Contents

<b>1</b>	<b>Overview</b>	<b>5</b>
1.1	Introduction to Essential ProjIO .....	5
1.2	Use Case Scenario .....	5
1.3	Prerequisites and Compatibility .....	5
1.4	Documentation .....	6
<b>2</b>	<b>Installation and Deployment</b>	<b>7</b>
2.1	Installation .....	7
2.2	Where to Find Samples? .....	7
2.2.1	Sample Installation Location .....	7
2.2.2	Viewing Samples .....	7
2.3	Deployment Procedures .....	12
2.3.1	DLLs .....	12
<b>3</b>	<b>Getting Started</b>	<b>13</b>
3.1	Feature Summary .....	13
<b>4</b>	<b>Concepts and Features</b>	<b>14</b>
4.1	Project .....	14
4.1.1	Properties, Methods, and Events Tables for Project .....	14
4.1.1.1	Constructors .....	14
4.1.1.2	Properties .....	14
4.1.1.3	Methods .....	17
4.1.2	Creating a simple project .....	18
4.1.3	Reading a project file .....	19
4.1.4	General Project Properties .....	20
4.1.4.1	Retrieving Project Properties .....	20
4.1.4.2	Setting Project Properties .....	21
4.1.5	Default Project Properties .....	22
4.1.5.1	Retrieving Default Project Properties .....	22
4.1.5.2	Setting Default Project Properties .....	23
4.1.6	Writing Project Summary Information .....	25
4.1.7	Fiscal Year Properties .....	26

4.1.7.1	Retrieving Fiscal Year Properties .....	27
4.1.7.2	Setting Fiscal Year Properties .....	27
4.1.8	Week Day Properties.....	28
4.1.8.1	Retrieving Week Day Properties .....	28
4.1.8.2	Setting Week Day Properties .....	29
4.2	Task .....	30
4.2.1	Properties, Methods, and Events Tables for Task .....	30
4.2.1.1	Constructors .....	30
4.2.1.2	Properties .....	31
4.2.1.3	Methods.....	36
4.2.2	Adding Tasks to a Project .....	36
4.2.3	Creating a summary task.....	37
4.2.4	Creating Task links .....	38
4.2.5	Writing Tasks to Projects.....	39
4.3	Resource .....	41
4.3.1	Properties, Methods, and Events Tables for Task .....	41
4.3.1.1	Constructors .....	41
4.3.1.2	Properties .....	41
4.3.1.3	Methods.....	45
4.3.2	Adding Resources to a Project.....	46
4.3.3	Writing Resources to a Project.....	46
4.4	Assignment.....	48
4.4.1	Properties, Methods, and Events Tables for Task .....	48
4.4.1.1	Constructors .....	48
4.4.1.2	Properties .....	48
4.4.1.3	Methods.....	52
4.4.2	Adding Assignments to a Project .....	52
4.5	Calendar .....	57
4.5.1	Properties, Methods, and Events Tables for Task .....	57
4.5.1.1	Constructors .....	57
4.5.1.2	Properties .....	57
4.5.1.3	Methods.....	58
4.5.2	Creating a Standard Calendar.....	58
<b>5</b>	<b>How To</b>	<b>60</b>

5.1	Does ProjIO require MS Project to be installed on the machine? .....	60
5.2	Will it be possible to view the generated project [.xml file] using ProjIO? .....	60
5.3	Can Essential ProjIO be used to read MS Project files? .....	60
5.4	How to retrieve tasks from a project? .....	60

# 1 Overview

---

## 1.1 Introduction to Essential ProjIO

Essential ProjIO is a 100% native .NET library that enables .NET applications to read and write Microsoft Project xml format documents without utilizing Microsoft Project. It does not use COM Interop and is built from scratch using C#. The ProjIO library can be used in any .NET environment including C# and VB. It is a Non-UI component that is used both on Windows Forms Applications and Web Applications.

## 1.2 Use Case Scenario

Essential ProjIO is used by Managers and Project Leads in companies to manage their projects. It helps them to effectively utilize the resources and increase productivity. It also helps them to complete tasks well before timelines.

## 1.3 Prerequisites and Compatibility

This section covers the requirements that are mandatory for using Essential ProjIO. It also lists operating systems and browsers compatible with the product.

### Prerequisites:

The prerequisites details are tabulated as follows:

*Table 1: Prerequisites*

Development Environments	<ul style="list-style-type: none"><li>• Visual Studio 2010 (Ultimate, Premium, Professional and Express)</li><li>• Visual Studio 2008 (Team System, Professional, Standard &amp; Express)</li><li>• Visual Studio 2005 (Professional, Standard &amp; Express)</li></ul>
.Net Framework Versions	<ul style="list-style-type: none"><li>• .NET 4.0</li><li>• .NET 3.5 SP1</li><li>• .NET 2.0</li></ul>

### Compatibility:

The compatibility details are tabulated as follows:


*Table 2: Compatibility*

Operating Systems	<ul style="list-style-type: none"> <li>Windows Server 2008 (32 bit and 64 bit)</li> <li>Windows 7 (32 bit and 64 bit)</li> <li>Windows Vista (32 bit and 64 bit)</li> <li>Windows XP</li> <li>Windows 2003</li> </ul>
MS Project Compatibility Support	<ul style="list-style-type: none"> <li>MS Project 2007</li> <li>MS Project 2010</li> </ul>

## 1.4 Documentation

Syncfusion provides the following documentation segments to provide all the necessary information pertaining to Essential ProjIO.

Table 3: Documentation

Type of Documentation	Location
Readme	Windows Forms-[drive:]\Program Files\Syncfusion\Essential Studio\x.x.x.x\Infrastructure\Data\Release Notes\readme.htm
Release Notes	Windows Forms-[drive:]\Program Files\Syncfusion\Essential Studio\x.x.x.x\Infrastructure\Data\Release Notes\readme.htm
User Guide (this document)	<p><b>Online</b></p> <p><a href="http://help.syncfusion.com/resources">http://help.syncfusion.com/resources</a> (Navigate to the ProjIO User Guide.)</p> <p> <b>Note: Click Download as PDF to access a PDF version.</b></p> <p><b>Installed Documentation</b></p> <p>Dashboard -&gt; Documentation -&gt; Installed Documentation.</p>
Class Reference	<p><b>Online</b></p> <p><a href="http://help.syncfusion.com/resources">http://help.syncfusion.com/resources</a> (Navigate to the Reporting User Guide. Select <i>ProjIO</i>, and then click the Class Reference link found in the upper right section of the page.)</p> <p><b>Installed Documentation</b></p> <p>Dashboard -&gt; Documentation -&gt; Installed Documentation.</p>

## 2 Installation and Deployment

---

### 2.1 Installation

For step-by-step installation procedure for the installation of Essential Studio, refer to the Installation topic under Installation and Deployment in the Common UG.

### 2.2 Where to Find Samples?

This section covers the location of the installed samples and describes the procedure to run the samples through the sample browser and online. It also lists the location of utilities, assemblies, and source code.

#### 2.2.1 Sample Installation Location

Sample install locations for different platforms are listed below:

- Windows Forms – The Windows Forms samples are installed in the following location:

...\MyDocuments\Syncfusion\EssentialStudio\VersionNumber\Windows\ProjIO.Windows\Samples\2.0

- WPF - The WPF samples are installed in the following location:

...\MyDocuments\Syncfusion\EssentialStudio\VersionNumber\WPF\ProjIO.WPF\Samples\3.5

#### 2.2.2 Viewing Samples

The samples can be viewed in any of the following three ways:

- **Run Samples** – Click to view the locally installed samples.
- **Online Samples**-Click to view online samples.
- **Explore Samples** – Explore samples on disk.

To view the samples:

1. Click **Start -> All Programs-> Syncfusion -> Essential Studio <x.x.x.x> -> Dashboard**.  
The UI samples are displayed by default.
2. Select **Reporting**.

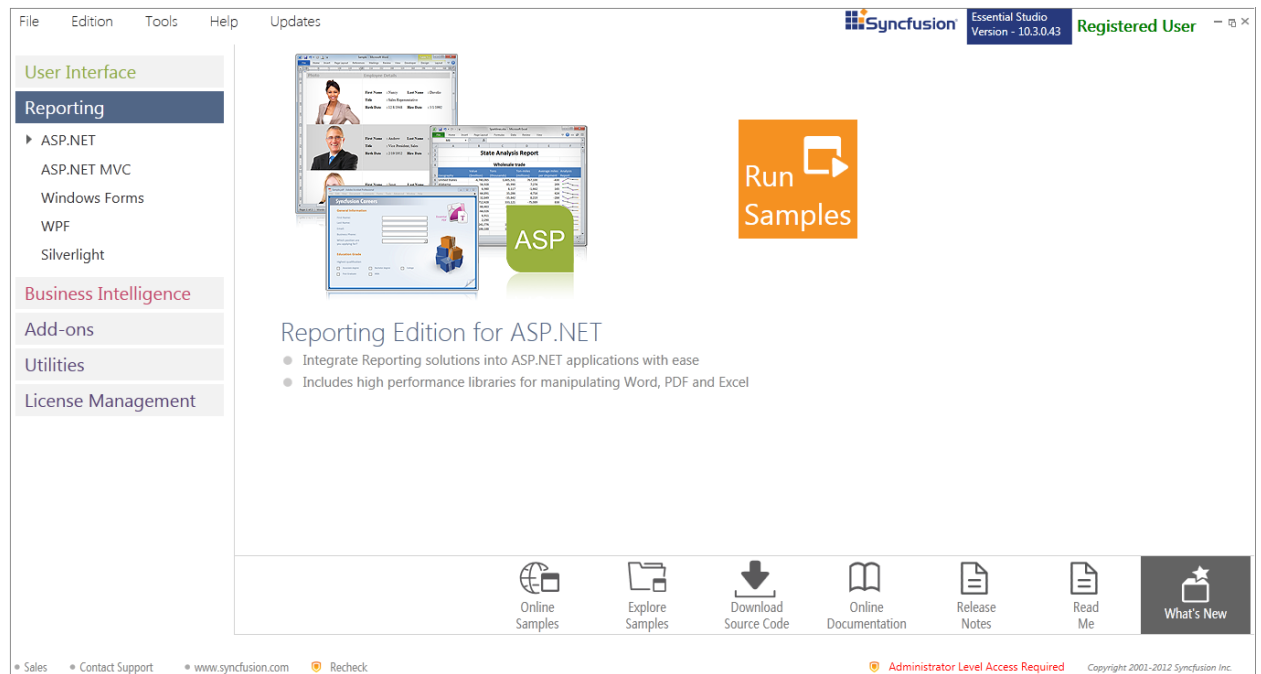


Figure 1: Essential Studio Reporting Dashboard

The steps to view the ProjIO samples in various platforms are discussed below.

## Windows

1. In the Dashboard window, click **Run Samples** for **Windows Forms** under **Reporting** Edition Panel. The **Windows Forms** Sample Browser window is displayed.



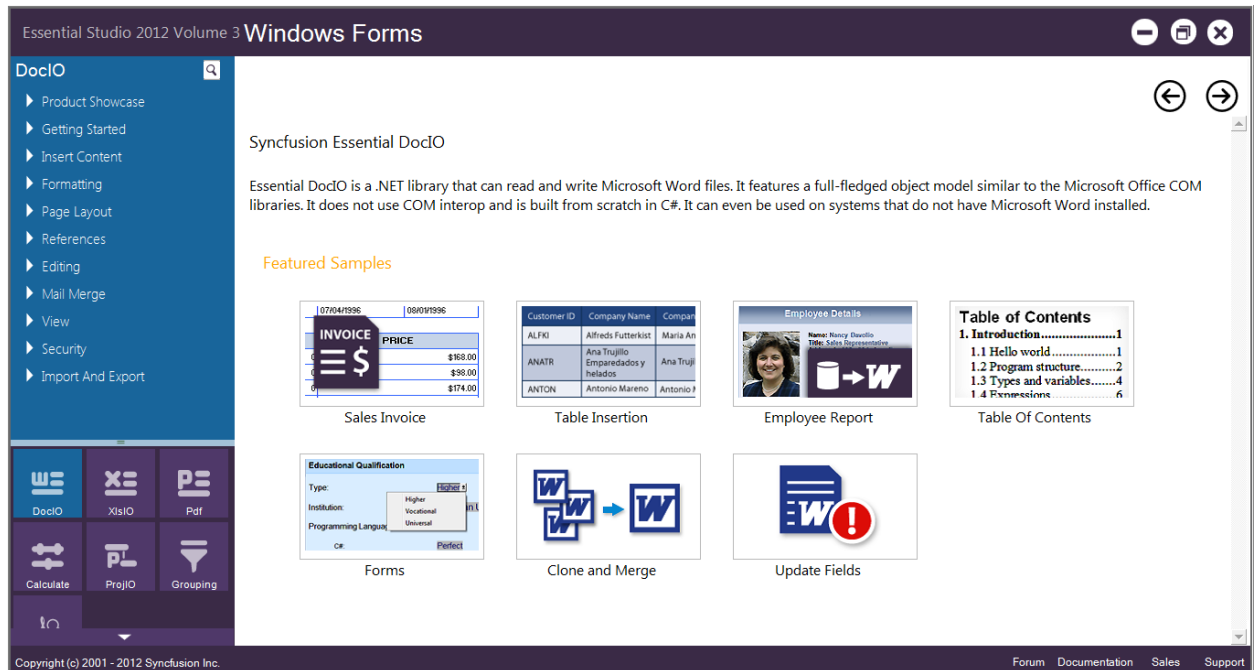


Figure 2: Essential Studio Reporting Windows Forms Dashboard

2. Click **ProjIO** from the bottom-left pane. The ProjIO samples are displayed.

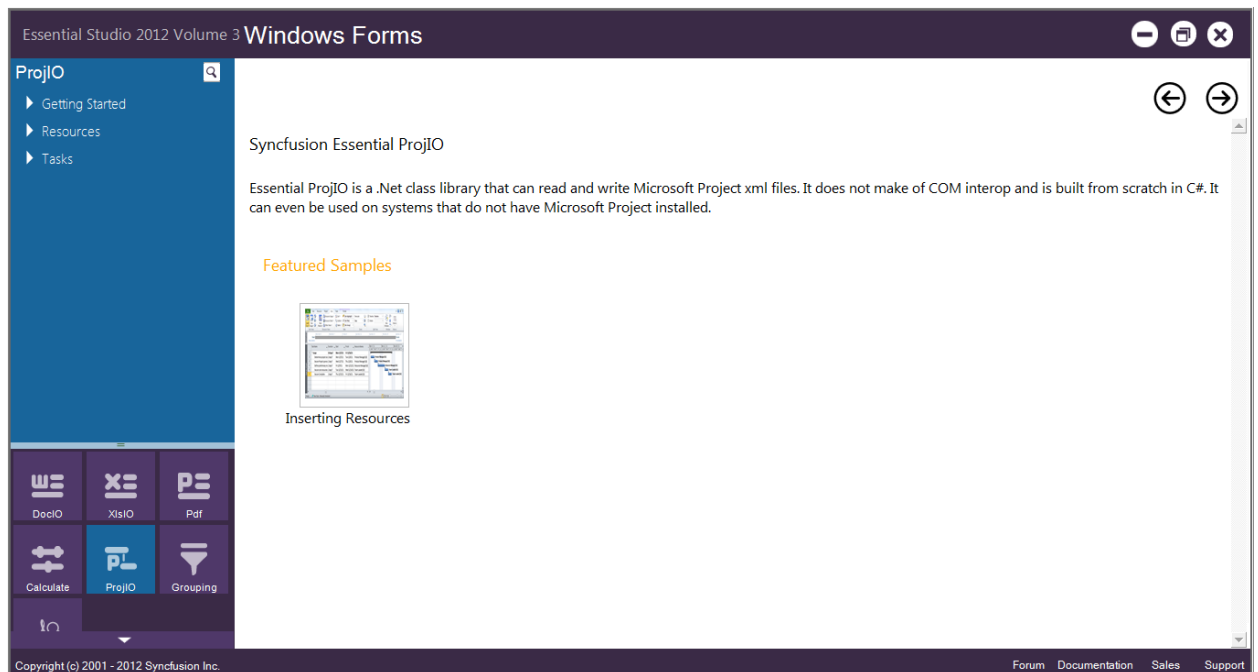


Figure 3: Essential Studio Reporting ProjIO Samples

3. Select any sample and browse through the features.

## WPF

1. In the dashboard window, click **Run Samples** for **WPF** under **Reporting** edition panel.  
The **WPF** Sample Browser window is displayed.

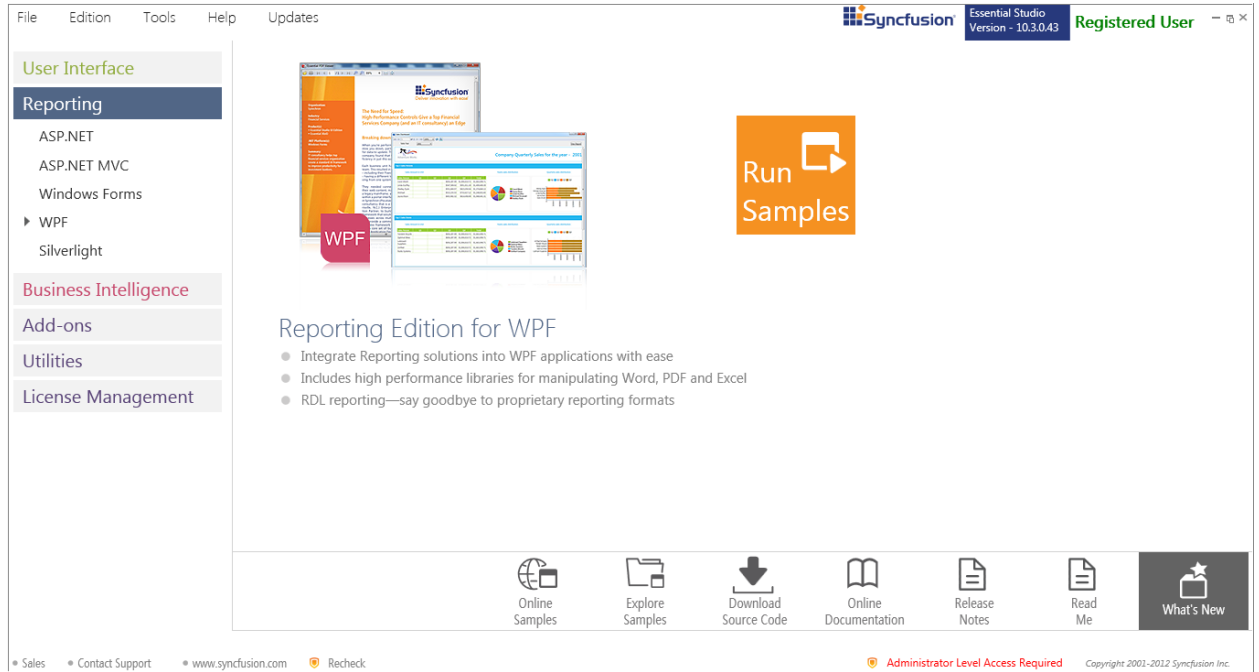


Figure 4: Essential Studio Reporting Dashboard

4. Select **Run Samples**. The default WPF sample will be displayed.

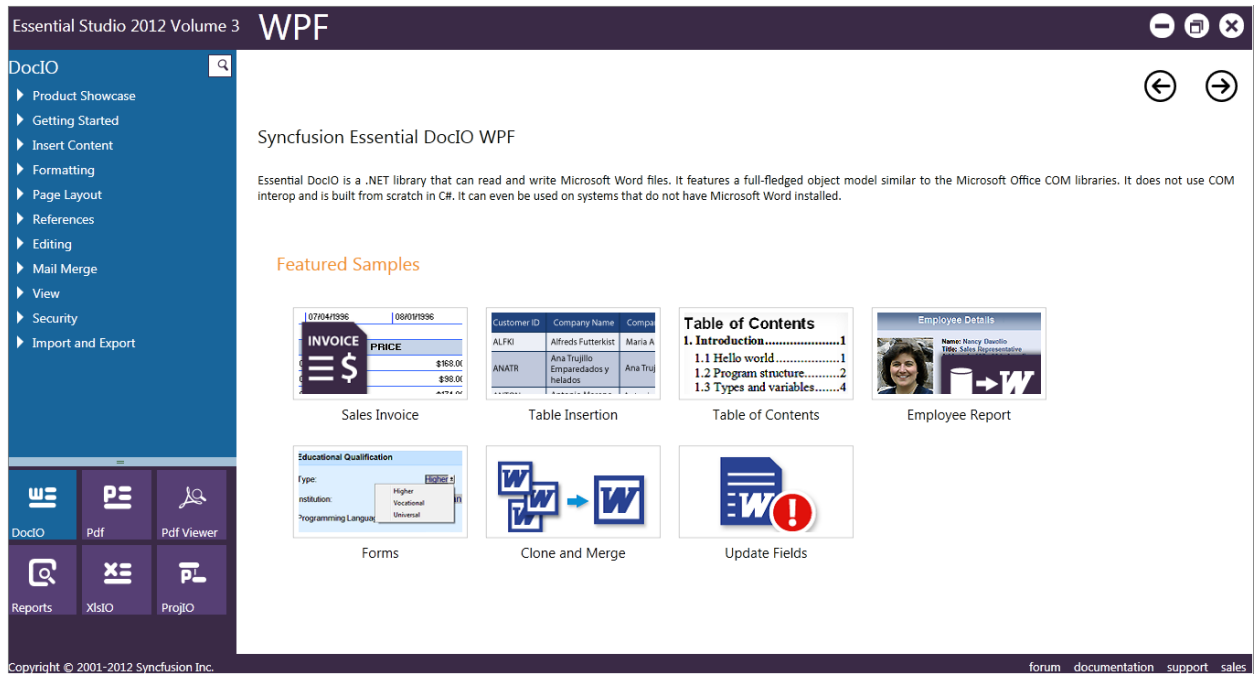


Figure 5: Essential Studio Reporting WPF Dashboard

5. Click **ProjIO** form the bottom-left pane and browse through the features.

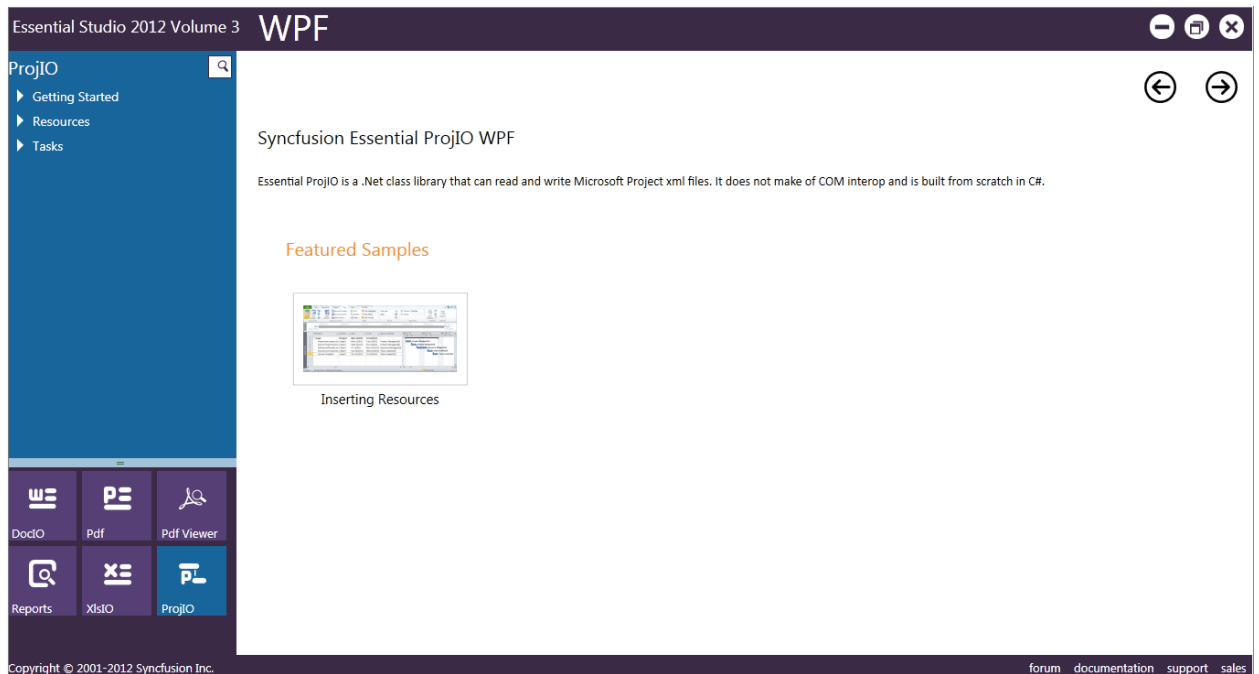


Figure 6: Essential Studio Reporting ProjIO Samples

## 2.3 Deployment Procedures

### DLLs

The following assemblies need to be referenced in your application for the usage of ProjIO.

- Syncfusion.Core.dll
- Syncfusion.ProjIO.Base.dll

## 3 Getting Started

---

### 3.1 Feature Summary

Important Features of Essential ProjIO are:

- Support to create and edit a new Project xml format documents
- Support to modify existing Project documents (XML format alone)
- Support to change project settings like Start and Finish dates
- Support to change default project settings like Default Standard Rate, Default Overtime Rate, Default Fixed Cost Accrual, Default Task Type and so on
- Support to read and write calendars for tasks and resources
- Support to define Calendar exceptions
- Support to manage task baselines
- Support to create and manage links between tasks
- Support to create and manage resources
- Support to assign and manage resources to tasks

## 4 Concepts and Features

---

### 4.1 Project

You can open, modify and create Project files using the **Project** Class. Project class has a structure similar to the MS Project document. Project class is useful in creating MS Projects in XML format. It can also be used to open or modify the existing Project files in XML format.

#### 4.1.1 Properties, Methods, and Events Tables for Project

##### 4.1.1.1 Constructors

Table 4: Project Constructor

Name	Description
Project.Project()	Initializes a new instance of the Project class

##### 4.1.1.2 Properties

Table 5: Project Properties

Property	Description
SaveVersion	Gets or sets the version of Microsoft Office Project from which the project was saved.
UID	Gets or set the unique ID of the project.
Name	Gets or sets the name of the project.
Title	Gets or sets the title of the project.
Subject	Gets or sets the subject of the project.
Category	Gets or sets the category of the project.
Company	Gets or sets the company that owns the project.
Manager	Gets or sets the manager of the project.
Author	Gets or sets the author of the project.
CreationDate	Gets or sets the date when the project was created.
Revision	Gets or sets the number of times a project has been saved.
LastSaved	Gets or sets the date the project was last saved.
ScheduleFromStart	True if the project is scheduled from the start date.
StartDate	Gets or sets the start date of the project. Required if ScheduleFromStart is true.

Property	Description
FinishDate	Gets or sets the finish date of the project. Required if ScheduleFromStart is false.
FYStartDate	Gets or sets the Fiscal Year starting month
CriticalSlackLimit	Gets or sets the number of days past its end date that a task can go before Microsoft Project marks that task as a critical task.
CurrencyDigits	Gets or sets the number of digits after the decimal symbol.
CurrencySymbol	Gets or sets the currency symbol used in the project.
CurrencyCode	Gets or sets the three letter currency character code as defined in ISO 4217. Only USD is supported.
CurrencySymbolPosition	Gets or sets the position of the currency symbol.
CalendarUID	Gets or sets the project calendar UID. Refers to a valid UID in the Calendars collection.
Calendar	Gets or sets the project calendar.
DefaultStartTime	Gets or sets the default start time of new tasks.
DefaultFinishTime	Gets or sets the default finish time of new tasks.
MinutesPerDay	Gets or sets the number of minutes per day.
MinutesPerWeek	Gets or sets the number of minutes per week.
DaysPerMonth	Gets or sets the number of days per month.
DefaultTaskType	Gets or sets the default type of new tasks.
DefaultFixedCostAccrual	Gets or sets the default from where fixed costs are accrued.
DefaultStandardRate	Gets or sets the default standard rate for new resources.
DefaultOvertimeRate	Gets or sets the default overtime rate for new resources.
DurationFormat	Gets or sets the format for expressing the bulk duration.
WorkFormat	Gets or sets the default work unit format.
EditableActualCosts	True if actual costs are editable.
HonorConstraints	True if tasks honor their constraint dates.
EarnedValueMethod	Gets or sets the default method for calculating earned value.

Property	Description
InsertedProjectsLikeSummary	True if subtasks are calculated as summary tasks.
MultipleCriticalPaths	True if multiple critical paths are calculated.
NewTasksEffortDriven	True if new tasks are effort driven.
NewTasksEstimated	True to show the estimated duration by default.
SplitsInProgressTasks	True if in-progress tasks can be split.
SpreadActualCost	True if actual costs are spread to the status date.
SpreadPercentComplete	True if percent complete is spread to the status date.
TaskUpdatesResource	True if updates to tasks, update resources.
FiscalYearStart	True to use fiscal year numbering.
WeekStartDay	Gets or sets the Start day of the week.
MoveCompletedEndsBack	True if the end of completed portions of tasks scheduled to begin after the status date but begun early should be moved back to the status date.
MoveRemainingStartsBack	True if the beginning of remaining portions of tasks scheduled to begin after the status date but began early, should be moved back to the status date.
MoveRemainingStartsForward	True if the beginning of remaining portions of tasks scheduled to begin late should be moved up to the status date.
MoveCompleteEndsForward	True if the end of completed portions of tasks scheduled to get completed before the status date but began late should be moved up to the status date.
BaselineForEarnedValue	Gets or sets the specific baseline used to calculate Variance values.
AutoAddNewResourcesAndTasks	True to automatically add new resources to the resource pool.
StatusDate	Gets or sets the date used for calculation and reporting.
CurrentDate	Gets or sets the system date that the XML was generated.
MicrosoftProjectServerURL	True if the project was created by a Project Server user as opposed to an NT user.
Autolink	True to auto link inserted or moved tasks.



Property	Description
NewTaskStartDate	Gets or sets the default date for new tasks start.
DefaultTaskEVMethod	Gets or sets the default earned value method for tasks.
ProjectExternallyEdited	True if the project XML was edited.
ExtendedCreationDate	Gets or sets date used for calculation and reporting.
ActualInSync	True if all actual work has been synchronized with the project.
RemoveFileProperties	True to remove all file properties on save.
AdminProject	True if the project is an administrative project.
BaselineCalendar	Gets or sets the name of the Baseline Calendar.
NewTasksAreManual	True if new tasks should be made in Manual mode.
UpdateManuallyScheduledTasksWhenEditingLinks	True to update manually scheduled tasks when editing links.
KeepTaskOnNearestWorkingTimeWhenMadeAutoScheduled	True if tasks moving from Manual to Auto Scheduled should be moved to the nearest working time.
OutlineCodes	Gets or sets the collection of outline code definitions associated with the project.
WBSMasks	Gets or sets the table of entries that define the outline code mask.
ExtendedAttributes	Gets or sets the collection of extended attribute (custom field) definitions associated with the project.
Calendars	Gets or sets the collection of calendars that are associated with the project.
Tasks	Gets or sets the collection of tasks that make up the project.
Resources	Gets or sets the collection of resources that make up the project.
Assignments	Gets or sets the collection of assignments that make up the project.

#### 4.1.1.3 Methods

Table 6: Project Methods

Method	Description
--------	-------------

Save	Saves Project instance to disk
CalculateResourceIDs	Recalculates UUIDs and IDs of resources starting from 0
CalculateTaskIDs	Recalculates UUIDs and IDs of tasks starting from 0
Equals	Returns a value indicating whether this instance is equal to a specified object
GetHashCode	Serves as a hash function for Project type
GetType	Gets the type of the current instance
ToString	Returns a string that represents the current object

### 4.1.2 Creating a simple project

**Project** is the main class of Essential ProjIO. We can only create project files in XML format. The following lines of code create a simple project.

[C#]

```
// Creating an instance of Project
Project project = new Project();

// Saving the project - Creates an empty project
project.Save("Empty Project.xml");
```

[VB]

```
' Creating an instance of Project
Dim project As Project = New Project()

' Saving the Project - Creates an empty project
project.Save("Empty Project.xml")
```

The XML project file can be viewed in Microsoft Project using the option **File – Open** and then selecting the XML format (\*.xml) option from the file types. Select '**Project Information**' option from the **Projects menu** and the options will look as follows:

Project Information for 'Empty Project'

Start date: Fri 10/7/11 Current date: Fri 10/7/11

Finish date: Fri 10/7/11 Status date: NA

Schedule from: Project Start Date Calendar: Standard

All tasks begin as soon as possible. Priority: 500

Enterprise Custom Fields

Department:

Custom Field Name	Value

Help Statistics... OK Cancel

Figure 7: Project Information for Empty Project

### 4.1.3 Reading a project file

**Read** method of the **ProjectReader** class is used to read the project files. The Read method has two overloads namely:

- Read(string filename) – opens the file specified by the given file name.
- Read(Stream stream) – opens the file specified by the Stream.

**Read** method returns a **Project** object, which can then be used to retrieve or manipulate project information.

The following code illustrates the use of the **Read** method:

[C#]

```
// Assigning the Project object returned by the Read method
Project P = ProjectReader.Open("SimpleProject.xml");
```

**[VB]**

```
' Assigning the Project object returned by the Read method
Dim P As Project = ProjectReader.Open("SimpleProject.xml")
```

## 4.1.4 General Project Properties

### 4.1.4.1 Retrieving Project Properties

The Project Properties can be retrieved by using the **Project** class. The following code snippet shows how to get the project properties. These properties can be viewed in MS Project by selecting **Project Properties** from the **File - Project** Menu.

**[C#]**

```
// Calling Open method of ProjectReader to get the Project object
Project project = ProjectReader.Open("Sample Project.xml");

// Displaying Project information
Console.WriteLine("Project Start Date: " + project.StartDate);

if (project.ScheduleFromStart)
    Console.WriteLine("Project Finish Date: " + project.StartDate);
else
    Console.WriteLine("Project Finish Date: " + project.FinishDate);

Console.WriteLine("Project Schedule From: " +
    (project.ScheduleFromStart ? "Project Start Date" : "Project Finish
    Date"));

Console.WriteLine("Current Date: " + project.CurrentDate);
Console.WriteLine("Status Date: " + project.StatusDate);
Console.WriteLine("Calendar: " + project.Calendar.Name);
```

**[VB]**

```
' Creating an instance of Project
Dim project As Project = ProjectReader.Open("Sample Project.xml")

' Displaying Project information
Console.WriteLine("Project Start Date: " + project.StartDate)

If (project.ScheduleFromStart) Then
    Console.WriteLine("Project Finish Date: " + project.StartDate)
Else
    Console.WriteLine("Project Finish Date: " + project.FinishDate)
End If

Console.WriteLine("Project Schedule From: " +
If(project.ScheduleFromStart = True, "Project Start Date", "Project
Finish Date"))

Console.WriteLine("Current Date: " + project.CurrentDate)
Console.WriteLine("Status Date: " + project.StatusDate)
Console.WriteLine("Calendar: " + project.Calendar.Name)
```

#### 4.1.4.2 Setting Project Properties

The Project class can be used to set Project properties such as Start Date, Finish Date, Calendar and so on.

The following code snippet shows how to set the Project properties:

**[C#]**

```
// Creating a new instance of the Project object
Project project = new Project();

// Setting Project information
project.ScheduleFromStart = true;
project.StartDate = new DateTime(2011, 7, 9);
```

```
project.CurrentDate = new DateTime(2011, 7, 9);  
project.StatusDate = new DateTime(2011, 7, 9);  
  
// Saving the Project  
project.Save("ProjectProperties.xml");
```

**[VB]**

```
' Creating an instance of Project  
Dim project As Project = New Project()  
  
' Setting Project information  
project.ScheduleFromStart = True  
project.StartDate = New DateTime(2011, 7, 9)  
project.CurrentDate = New DateTime(2011, 7, 9)  
project.StatusDate = New DateTime(2011, 7, 9)  
  
' Saving the Project  
project.Save("ProjectProperties.xml")
```

## 4.1.5 Default Project Properties

The **Project** class is used to get/set Project Default Properties. The default properties of a project can be viewed using the **Tools – Options** menu in MS Project.

### 4.1.5.1 Retrieving Default Project Properties

The following example illustrates how to retrieve default project properties.

**[C#]**

```
// Calling Open method of ProjectReader to get the Project object  
Project project = ProjectReader.Open("Sample Project.xml");  
  
// Retrieving Project Default information  
Console.WriteLine("Default Start Time: " + project.DefaultStartTime);
```

```
Console.WriteLine("Default Finish Time: " + project.DefaultFinishTime);  
Console.WriteLine("Default Standard Rate: " +  
project.DefaultStandardRate);  
Console.WriteLine("Default Overtime Rate: " +  
project.DefaultOvertimeRate);  
Console.WriteLine("Default Task EV Method: " +  
project.DefaultTaskEVMethod);  
Console.WriteLine("Default Cost Accrual: " +  
project.DefaultFixedCostAccrual);
```

[VB]

```
' Creating an instance of Project  
Dim project As Project = ProjectReader.Open("Sample Project.xml")  
  
' Retriving Project information  
Console.WriteLine("Default Start Time: " &  
project.DefaultStartTime.ToString())  
  
Console.WriteLine("Default Finish Time: " &  
project.DefaultFinishTime.ToString())  
  
Console.WriteLine("Default Standard Rate: " &  
project.DefaultStandardRate)  
Console.WriteLine("Default Overtime Rate: " &  
project.DefaultOvertimeRate)  
Console.WriteLine("Default Task EV Method: " &  
project.DefaultTaskEVMethod)  
Console.WriteLine("Default Cost Accrual: " &  
project.DefaultFixedCostAccrual)
```

#### 4.1.5.2 Setting Default Project Properties

The following example shows how to set the default project properties.

[C#]

```
// Creating a new instance of the Project object
Project project = new Project();

// Setting Project Default information
project.DefaultStartTime = new TimeSpan(8, 0, 0);
project.DefaultFinishTime = new TimeSpan(17, 0, 0);
project.DefaultStandardRate = 0f;
project.DefaultOvertimeRate = 0f;
project.DefaultTaskEVMethod = EarnedValueMethod.PercentComplete;
project.DefaultFixedCostAccrual = DefaultFixedCostAccrual.Prorated;

// Saving the Project
project.Save("DefaultProjectProperties.xml");
```

[VB]

```
' Creating an instance of a Project
Dim project As Project = New Project()

' Setting Project information
project.DefaultStartTime = New TimeSpan(8, 0, 0)
project.DefaultFinishTime = New TimeSpan(17, 0, 0)
project.DefaultStandardRate = 0.0F
project.DefaultOvertimeRate = 0.0F
project.DefaultTaskEVMethod = EarnedValueMethod.PercentComplete
project.DefaultFixedCostAccrual = DefaultFixedCostAccrual.Prorated

' Saving the Project
project.Save("DefaultProjectProperties.xml")
```



### 4.1.6 Writing Project Summary Information

Project class contains properties that can get or set the summary information of a project file in XML format. Using this class, the summary information can be updated and the file can be written back in XML format. The following code shows how this can be done.

[C#]

```
// Calling Read method of ProjectReader to get the Project object
Project project = ProjectReader.Open("Sample Project.xml");

// Setting Project Default information
project.SaveVersion = 14;
project.Author = "Sam Anderson";
project.Manager = "John Henson";
project.Company = "Syncfusion";
project.CreationDate = new DateTime(2011, 10, 8);
project.Subject = "Essential ProjIO";
project.Title = "Sample Project";

// Saving the Project
project.Save("Empty Project.xml");
```

[VB]

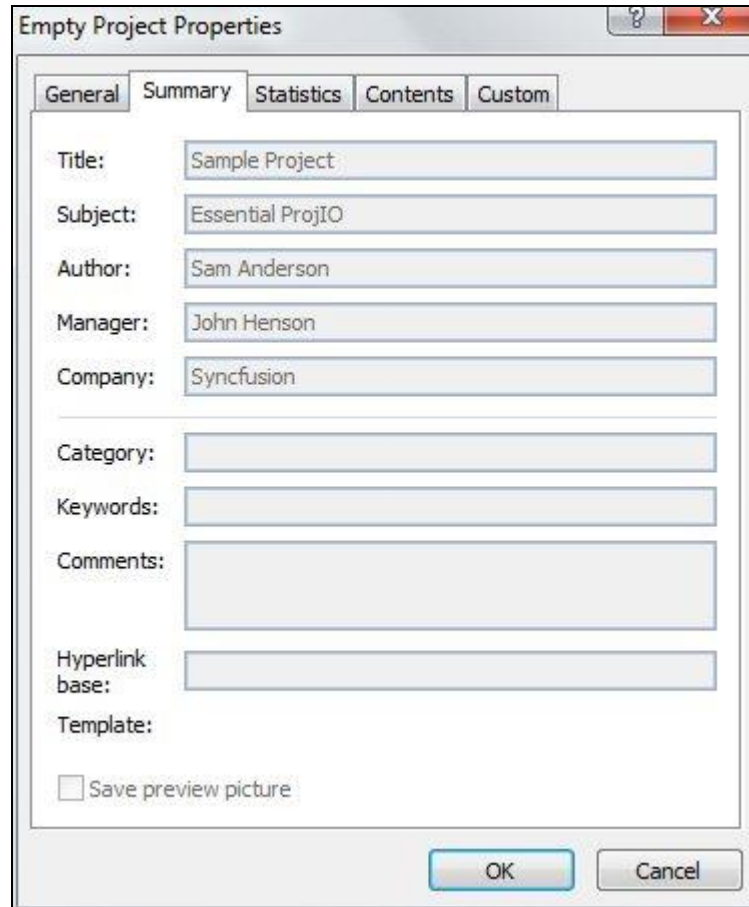
```
' Calling Read method of ProjectReader to get the Project object
Dim project As Project = ProjectReader.Open("Sample Project.xml")

' Retriving Project information
project.SaveVersion = 14
project.Author = "Sam Anderson"
project.Manager = "John Henson"
project.Company = "Syncfusion"
project.CreationDate = New DateTime(2011, 10, 8)
project.Subject = "Essential ProjIO"
```

```
project.Title = "Sample Project"

' Saving the Project
project.Save("Empty Project.xml")
```

The project summary information added through the above code can be viewed by checking the **Project Information – Advanced Properties** in the **File** menu.



The screenshot shows a Windows-style dialog box titled "Empty Project Properties". It has five tabs: "General", "Summary", "Statistics", "Contents", and "Custom". The "Summary" tab is selected. Inside the dialog, there are several text input fields with labels to their left: "Title:" (containing "Sample Project"), "Subject:" (containing "Essential ProjIO"), "Author:" (containing "Sam Anderson"), "Manager:" (containing "John Henson"), "Company:" (containing "Syncfusion"), "Category:" (empty), "Keywords:" (empty), "Comments:" (empty), "Hyperlink base:" (empty), and "Template:" (empty). Below these fields is a checkbox labeled "Save preview picture" which is unchecked. At the bottom right of the dialog are two buttons: "OK" and "Cancel".

Figure 8: Empty Project Properties

### 4.1.7 Fiscal Year Properties

The **Project** class properties **FYStartDate** and **FiscalYearStart** are used to get or set the Fiscal Year properties. **FYStartDate** defines the fiscal year start month and the **FiscalYearStart** property determines whether the fiscal year numbering has been used in the project.

#### 4.1.7.1 Retrieving Fiscal Year Properties

The following code snippets retrieve Fiscal year properties from a project:

[C#]

```
// Calling Open method of ProjectReader to get the Project object
Project project = ProjectReader.Open("Sample Project.xml");

// Retrieving Fiscal Year information
Console.WriteLine("Fiscal Year Start Month: " + project.FYStartDate);

Console.WriteLine(project.FiscalYearStart ? "Fiscal Year Numbering is
used in the Project" : "Fiscal Year Numbering is not used in the
Project");
```

[VB]

```
' Calling Read method of ProjectReader to get the Project object
Dim project As Project = ProjectReader.Open("Sample Project.xml")

' Retrieving Fiscal Year information
Console.WriteLine("Fiscal Year Start Month: " + project.FYStartDate)

Console.WriteLine(If(project.FiscalYearStart, "Fiscal Year Numbering is
used in the Project", "Fiscal Year Numbering is not used in the
Project"))
```

#### 4.1.7.2 Setting Fiscal Year Properties

The following code sets the Fiscal year properties for a project.

[C#]

```
// Creating a new instance of Project object
Project project = new Syncfusion.ProjIO.Project();
```

```
// Setting Fiscal Year information
project.FYStartDate = FYStartDate.April;
project.FiscalYearStart = true;

// Saving the Project
project.Save("FiscalProperties.xml");
```

**[VB]**

```
' Creating an instance of Project
Dim project As Project = New Project()

' Setting Fiscal Year information
project.FYStartDate = FYStartDate.April
project.FiscalYearStart = True

' Saving the Project
project.Save("FiscalProperties.xml")
```

## 4.1.8 Week Day Properties

The Project class contains properties **WeekStartDay**, **DaysPerMonth**, **MinutesPerDay**, **MinutesPerWeek** that can be used to get or set Week day properties of a project.

### 4.1.8.1 Retrieving Week Day Properties

The following code snippets illustrate how to retrieve the Week day properties of a project.

**[C#]**

```
// Opening the project file
Project project = Syncfusion.ProjIO.ProjectReader.Open("Sample
Project.xml");

// Retrieving Week day properties
```

```
Console.WriteLine("Weeks starts on: " + project.WeekStartDay);  
Console.WriteLine("No. of working days per month: " +  
project.DaysPerMonth);  
Console.WriteLine("No. of minutes per day: " + project.MinutesPerDay);  
Console.WriteLine("No. of minutes per week: " +  
project.MinutesPerWeek);
```

[VB]

```
' Opening the project file  
Dim project As Project = ProjectReader.Open("Sample Project.xml")  
  
' Retrieving Week day properties  
Console.WriteLine("Weeks starts on: " + project.WeekStartDay)  
Console.WriteLine("No. of working days per month: " +  
project.DaysPerMonth)  
Console.WriteLine("No. of minutes per day: " + project.MinutesPerDay)  
Console.WriteLine("No. of minutes per week: " + project.MinutesPerWeek)
```

#### 4.1.8.2 Setting Week Day Properties

The following code snippet illustrates how to set the Week day properties of a project.

[C#]

```
// Creating a new Project instance  
Project project = new Project();  
  
// Setting week day properties  
project.WeekStartDay = WeekStartDay.Monday;  
project.DaysPerMonth = 24;  
project.MinutesPerDay = 480;  
project.MinutesPerWeek = 2880;
```

```
//Saving the project
project.Save("WeekDayProperties.xml");
```

[VB]

```
' Creating a new Project instance
Dim project As Project = New Project()

' Setting Week day properties
project.WeekStartDay = WeekStartDay.Monday
project.DaysPerMonth = 24
project.MinutesPerDay = 480
project.MinutesPerWeek = 2880

' Saving the Project
project.Save("WeekDayProperties.xml")
```

## 4.2 Task

Task class exposes ways to create a task for a Project. Task is useful in creating tasks and adding them to the Project. Using Task class, one can create, add and modify tasks. It can also be used to obtain task information.

### 4.2.1 Properties, Methods, and Events Tables for Task

#### 4.2.1.1 Constructors

Table 7: Task Constructors

Name	Description
Task.Task()	Initializes a new instance of the Task class.
Task.Task(string taskName)	Initializes a new instance of the Task class with the task name.

### 4.2.1.2 Properties

Table 8: Task Properties

Property	Description
UID	Gets or sets the unique ID of the task.
ID	Gets or sets the position identifier of the task within the list of tasks.
Name	Gets or sets the name of the task.
Type	Gets or sets the type of task.
IsNull	True if the task is null
CreateDate	Gets or sets the date the task was created.
Contact	Gets or sets the contact person for the task.
WBS	Gets or sets the work breakdown structure code of the task.
WBSLevel	Gets or sets the rightmost WBS level of the task.
OutlineNumber	Gets or sets the outline number of the task.
OutlineLevel	Gets or sets the outline level of the task.
Priority	Gets or sets the priority of the task from 0 to 1000.
Start	Gets or sets the scheduled start date of the task
Finish	Gets or sets the scheduled finish date of the task.
Duration	Gets or sets the planned duration of the task.
DurationFormat	Gets or sets the format for expressing the Duration of the Task.
Work	Gets or sets the amount of scheduled work for the task.
Stop	Gets or sets the date that the task was stopped.
Resume	Gets or sets the date that the task resumed.
ResumeValid	True if the task can be resumed.
EffortDriven	True if the task is effort-driven.
Recurring	True if the task is a recurring task.

Property	Description
OverAllocated	True if the task is overallocated.
Estimated	True if the task is estimated.
Milestone	True if the task is a milestone.
Summary	True if the task is a summary task.
DisplayAsSummary	True if the task should be displayed as a summary task.
Critical	True if the task is in the critical chain.
IsSubProject	True if the task is an inserted project.
IsSubProjectReadOnly	True if the inserted project is read-only.
SubProjectName	Gets or sets the source location of the inserted project.
ExternalTask	True if the task is external.
ExternalTaskProject	Gets or sets the source location and task identifier of the external task.
EarlyStart	Gets or sets the early start date of the task.
EarlyFinish	Gets or sets the early finish date of the task.
LateStart	Gets or sets the late start date of the task.
LateFinish	Gets or sets the late finish date of the task.
StartVariance	Gets or sets the variance of the task start date from the baseline start date as minutes x 1000.
FinishVariance	Gets or sets the variance of the task finish date from the baseline finish date as minutes x 1000.
WorkVariance	Gets or sets the variance of task work from the baseline task work as minutes x 1000.
FreeSlack	Gets or sets the amount of free slack.
StartSlack	Gets or sets the amount of free slack at the start of the task.
FinishSlack	Gets or sets the amount of free slack at the end of the task.
TotalSlack	Gets or sets the amount of total slack.



Property	Description
FixedCost	Gets or sets the fixed cost of the task.
FixedCostAccrual	Gets or sets how the fixed cost is accrued against the task. Values are: 1=Start, 2=Prorated and 3=End.
PercentComplete	Gets or sets the percentage of the task duration completed.
PercentWorkComplete	Gets or sets the percentage of the task work completed.
Cost	Gets or sets the projected or scheduled cost of the task.
OvertimeCost	Gets or sets the sum of the actual and remaining overtime cost of the task.
OvertimeWork	Gets or sets the amount of overtime work scheduled for the task.
ActualStart	Gets or sets the actual start date of the task.
ActualFinish	Gets or sets the actual finish date of the task.
ActualDuration	Gets or sets the actual duration of the task.
ActualCost	Gets or sets the actual cost of the task.
ActualOvertimeCost	Gets or sets the actual overtime cost of the task.
ActualWork	Gets or sets the actual work for the task.
ActualOvertimeWork	Gets or sets the actual overtime work for the task.
RegularWork	Gets or sets the amount of non-overtime work scheduled for the task.
RemainingDuration	Gets or sets the amount of time required to complete the unfinished portion of the task.
RemainingCost	Gets or sets the remaining projected cost of completing the task.
RemainingWork	Gets or sets the remaining work scheduled to complete the task.
RemainingOvertimeCost	Gets or sets the remaining overtime cost projected to finish the task.

Property	Description
RemaningOvertimeWork	Gets or sets the remaining overtime work scheduled to finish the task.
ACWP	Gets or sets the actual cost of work performed on the task to-date.
CV	Gets or sets the earned value cost variance.
ConstraintType	Gets or sets the constraint on the start or finish date of the task.
CalendarUID	Gets or sets the task calendar. Refers to a valid UID in the Calendars collection.
ConstraintDate	Gets or sets the date argument for the task constraint type.
Deadline	Gets or sets the deadline for the task to be completed.
LevelAssignments	True if leveling can adjust assignments.
LevelingCanSplit	True if leveling can split the task.
LevelingDelay	Gets or sets the delay caused by leveling the task.
LevelingDelayFormat	Gets or sets the format for expressing the duration of the delay.
PreLevelStart	Gets or sets the start date of the task before it was leveled.
PreLevelFinish	Gets or sets the finish date of the task before it was leveled.
Hyperlink	Gets or sets the title of the hyperlink associated with the task.
HyperlinkAddress	Gets or sets the hyperlink associated with the task.
HyperlinkSubAddress	Gets or sets the document bookmark of the hyperlink associated with the task.
IgnoreResourceCalendar	True if the task ignores the resource calendar.
Notes	Gets or sets the text notes associated with the task.
HideBar	True if the GANTT bar of the task is hidden when displayed in Microsoft Project.

Property	Description
Rollup	True if the task is rolled up.
BCWS	Gets or sets the budgeted cost of work scheduled for the task.
BCWP	Gets or sets the budgeted cost of work performed on the task to-date.
PhysicalPercentComplete	Gets or sets the percentage complete value entered by the Project Manager.
EarnedValueMethod	Gets or sets the method for calculating earned value.
PredecessorLink	Gets or sets the predecessor task of the task that contains it.
ActualWorkProtected	Gets or sets the duration through which actual work is protected.
ActualOvertimeWorkProtected	Gets or sets the duration through which actual overtime work is protected.
ExtendedAttribute	Gets or sets the value of an extended attribute.
Baseline	Gets or sets the collection of baseline values of the task.
OutlineCode	Gets or sets the value of an outline code.
IsPublished	True if the task is published.
StatusManager	Gets or sets the name of the task status manager.
CommitmentStart	Gets or sets the start date of the deliverable.
CommitmentFinish	Gets or sets the finish date of the deliverable.
CommitmentType	Gets or sets the commitment type of the deliverable.
Active	True if the task is active.
Pinned	True if the task is in manually scheduled mode.
PinnedStart	Gets or sets text displayed in start field when the task is in Manually Scheduled mode.
PinnedFinish	Gets or sets text displayed in finish field when the task is in Manually Scheduled mode.

Property	Description
PinnedDuration	Gets or sets text displayed in duration field when the task is in Manually Scheduled mode.
TimePhasedData	Gets or sets the time phased data block associated with the task.

### 4.2.1.3 Methods

Table 9: Task Methods

Method	Description
GetHashCode	Serves as a hash function for Task type.
GetType	Gets the type of the current instance.
ToString	Returns a string that represents the current object.

## 4.2.2 Adding Tasks to a Project

Tasks can be created in one or more ways as given below.

- By a default constructor

Creating a task instance without using any parameter as shown in the following code snippet:

[C#]

```
Task task1 = new Task();
task1.Name = "Main Task";
task1.Start = DateTime.Now;
task1.Finish = DateTime.Now;
```

[VB]

```
Dim task1 As Task = New Task()
task1.Name = "Main Task"
task1.Start = DateTime.Now
task1.Finish = DateTime.Now
```

- By name

Creating a task instance by passing the task name as shown in the following code snippet:

**[C#]**

```
Task task1 = new Task("Main Task");  
task1.Start = DateTime.Now;  
task1.Finish = DateTime.Now;
```

**[VB]**

```
Dim task1 As Task = New Task("Main Task")  
task1.Start = DateTime.Now  
task1.Finish = DateTime.Now
```

### 4.2.3 Creating a summary task

To make a task as the summary task, you need to make use of the **IsSummary** property of the **Task** class.

The following example illustrates making a task as Summary task.

**[C#]**

```
Task task1 = new Task("Main Task");  
task1.Start = DateTime.Now;  
task1.Finish = DateTime.Now;  
task1.IsSummary = true;
```

**[VB]**

```
Dim task1 As Task = New Task("Main Task")  
task1.Start = DateTime.Now  
task1.Finish = DateTime.Now  
task1.IsSummary = True
```

The summary task created using the above code will look like as shown below when viewed in Microsoft Project.

	<div><div><div><div></div><div>i</div></div></div><div>Task Mode</div></div>	Task Name	Duration	Start	Finish	Oct 1, '11							Oct 2, '11							Oct 9, '11						
						T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W					
1	<div><div></div><div></div><div></div></div>	<div><div></div><div>Task1</div></div>	1 day?	Fri 10/7/11	Fri 10/7/11																					
2	<div><div></div><div></div><div></div></div>	Task2	1 day?	Fri 10/7/11	Fri 10/7/11																					
			</																							

Figure 9: Summary Task Created

#### 4.2.4 Creating Task links

A task link is created using the default constructor of the **TaskLink** class. It accepts three parameters. The first parameter defines the predecessor **Task**, second parameter defines the successor **Task** and third parameter defines the task link type from values specified by **TaskLinkType** enumeration type.

The following example illustrates how to create links between two tasks.

[C#]

```
// Creating two tasks that are to be linked
Task task1 = new Task("Task1");
Task task2 = new Task("Task2");

// Link task1 and task2
TaskLink link = new TaskLink(task1, task2, TaskLinkType.FinishToStart);
```

[VB]

```
' Creating tasks that are to be linked
Dim task1 As Task = New Task("Task1")
Dim task2 As Task = New Task("Task2")

' Creating a link between task1 and task2
Dim link As TaskLink = New TaskLink(task1, task2,
TaskLinkType.FinishToStart)
```

## 4.2.5 Writing Tasks to Projects

**RootTask** property of the **Project** class contains the **Children** property that returns the list of **Task** objects. The **Children** property is used to update the tasks.

The following code snippet demonstrates writing tasks to a project.

[C#]

```
// Creating an instance of the Project
Project P = new Project();

// Creating two tasks to be added to the project
Task task1 = new Task("Task1");
task1.Duration = new TimeSpan(8, 0, 0);
Task task2 = new Task("Task2");
task2.Duration = new TimeSpan(8, 0, 0);

// Adding the tasks to the RootTask of project
P.RootTask.Children.Add(task1);
P.RootTask.Children.Add(task2);

// Calculating Task IDs and UUIDs
P.CalculateTaskIDs();

// Link "Task1" and "Task2"
TaskLink link = new TaskLink(task1, task2, TaskLinkType.FinishToStart);

// Saving the project
P.Save("ProjectWithTasks.xml");
```

**[VB]**

```
' Creating an instance of the Project
Dim P As Project = New Project()

' Creating tasks that are to be linked
Dim task1 As Task = New Task("Task1")
task1.Duration = new TimeSpan(8, 0, 0)
Dim task2 As Task = New Task("Task2")
task2.Duration = new TimeSpan(8, 0, 0)

' Adding the tasks to the RootTask of the Project
P.RootTask.Children.Add(task1)
P.RootTask.Children.Add(task2)

' Calculating Task IDs and UIDs
P.CalculateTaskIDs()

' Creating a link between task1 and task2
Dim link As TaskLink = New TaskLink(task1, task2,
TaskLinkType.FinishToStart)

' Saving the project
P.Save("ProjectWithTasks.xml")
```

The project file created using above code will look as shown in the following screenshot.



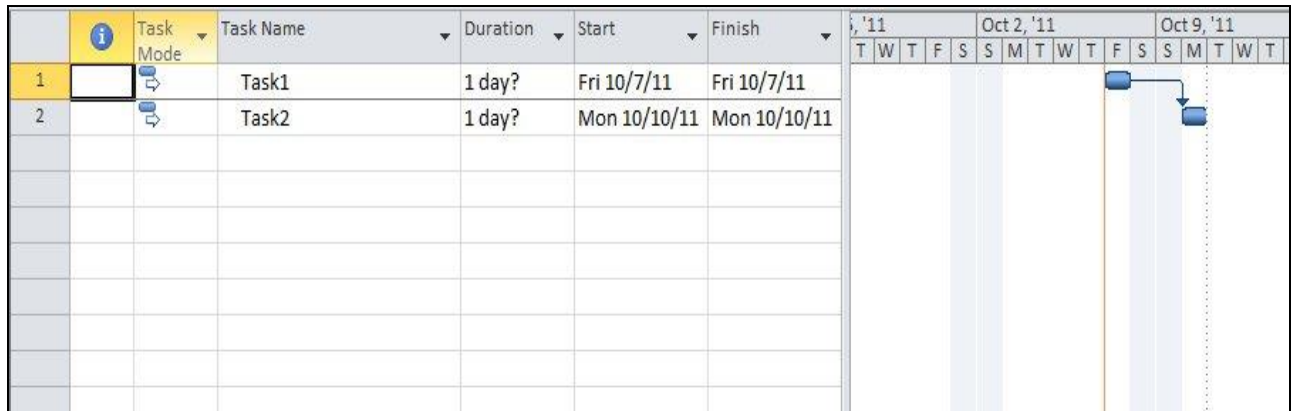


Figure 10: Project File Created

## 4.3 Resource

Resources class can be used to create resources and add them to Projects. However, creating resources do not assign them to tasks. One has to explicitly assign the resources to tasks using Assignment class (4.4). Using the Resources class, one can create, add, and modify resources to projects. It can also be used to retrieve resource information.

### 4.3.1 Properties, Methods, and Events Tables for Task

#### 4.3.1.1 Constructors

Table 10: Resource Constructor

Name	Description
Resource.Resource()	Initializes a new instance of the Resource class.

#### 4.3.1.2 Properties

Table 11: Resource Properties

Property	Description
UID	Gets or sets the unique identifier of the resource
ID	Gets or sets the position identifier of the resource within the list of resources
Name	Gets or sets the name of the resource
Type	Gets or sets the type of resource
IsNull	True if the resource is null
Initials	Gets or sets the initials of the resource
Phonetics	Gets or sets the phonetic spelling of the resource

Property	Description
	name. For use with Japanese only.
NTAccount	Gets or sets the NT account associated with the resource
MaterialLabel	Gets or sets the unit of measure for the material resource
Code	Gets or sets the code or other information about the resource
Group	Gets or sets the group to which the resource belongs
WorkGroup	Gets or sets the type of workgroup to which the resource belongs
EmailAddress	Gets or sets the email address of the resource
Hyperlink	Gets or sets the title of the hyperlink associated with the resource
HyperlinkAddress	Gets or sets the hyperlink associated with the resource
HyperlinkSubAddress	Gets or sets the document bookmark of the hyperlink associated with the resource
MaxUnits	Gets or sets the maximum number of units that the resource is available
PeakUnits	Gets or sets the largest number of units assigned to the resource at any time
OverAllocated	True if the resource is over allocated.
AvailableFrom	Gets or sets the first date that the resource is available.
AvailableTo	Gets or sets the last date the resource is available.
Start	Gets or sets the scheduled start date of the resource.
Finish	Gets or sets the scheduled finish date of the resource.
CanLevel	True if the resource can be leveled.

Property	Description
AccrueAt	Gets or sets how cost is accrued against the resource.
Work	Gets or sets the total work assigned to the resource across all assigned tasks.
RegularWork	Gets or sets the amount of non-overtime work assigned to the resource.
OvertimeWork	Gets or sets the amount of overtime work assigned to the resource.
ActualWork	Gets or sets the amount of actual work performed by the resource.
RemainingWork	Gets or sets the amount of remaining work required to complete all assigned tasks.
ActualOvertimeWork	Gets or sets the amount of actual overtime work performed by the resource.
RemainingOvertimeWork	Gets or sets the amount of remaining overtime work required to complete all tasks.
PercentWorkComplete	Gets or sets the percentage of work completed across all tasks.
StandardRate	Gets or sets the standard rate of the resource. This value is as of the current date if a rate table exists for the resource.
StandardRateFormat	Gets or sets the units used by Microsoft Project to display the standard rate.
Cost	Gets or sets the total project cost for the resource across all assigned tasks.
OvertimeRate	Gets or sets the overtime rate of the resource. This value is as of the current date if a rate table exists for the resource.
OvertimeRateFormat	Gets or sets the units used by Microsoft Project to display the overtime rate.
OvertimeCost	Gets or sets the total overtime cost for the resource including actual and remaining overtime costs.

Property	Description
CostPerUse	Gets or sets the cost per use of the resource. This value is as of the current date if a rate table exists for the resource.
ActualCost	Gets or sets the actual cost incurred by the resource across all assigned tasks.
ActualOvertimeCost	Gets or sets the actual overtime cost incurred by the resource across all assigned tasks.
Remaining Cost	Gets or sets the remaining projected cost of the resource to complete all assigned tasks.
RemainingOvertimeCost	Gets or sets the remaining projected overtime cost of the resource to complete all assigned tasks.
WorkVariance	Gets or sets the difference between the baseline work and the work as minutes x 1000.
CostVariance	Gets or sets the difference between the baseline cost and the cost.
SV	Gets or sets the earned value schedule variance, through the project status date.
CV	Gets or sets the earned value cost variance, through the project status date.
ACWP	Gets or sets the actual cost of the work performed by the resource for the project to-date.
CalendarUID	Gets or sets the resource calendar. Refers to a valid UID in the Calendars collection.
Notes	Gets or sets the text notes associated with the resource.
BCWS	Gets or sets the budget cost of work scheduled for the resource.
BCWP	Gets or sets the budgeted cost of the work performed by the resource for the project to-date.
IsGeneric	True if the resource is generic.
IsInactive	True if the resource is set to inactive.
IsEnterprise	True if the resource is an Enterprise resource.

Property	Description
BookingType	Gets or sets the booking type of the resource.
ActualWorkProtected	Gets or sets the duration through which actual work is protected.
ActualOvertimeWorkProtected	Gets or sets the duration through which actual overtime work is protected.
ActiveDirectoryGUID	Gets or sets the Active Directory GUID for the resource.
CreationDate	Gets or sets the date the resource was created.
ExtendedAttribute	Gets or sets the value of an extended attribute.
Baseline	Gets or sets the baseline values for the resources.
OutlineCode	Gets or sets the value of an outline code.
IsCostResource	True if the resource is a cost resource.
AssnOwner	Gets or sets the name of the assignment owner.
AssnOwnerGuid	Gets or sets the GUID of the assignment owner.
IsBudget	True if the resource is a budget resource.
AvailabilityPeriods	Gets or sets the collection of periods during which the resource is available.
Rates	Gets or sets the collection of periods and the rates associated with each one.
TimePhasedData	Gets or sets the time phased data.

#### 4.3.1.3 Methods

Table 12: Resource Methods

Method	Description
Equals	Returns a value indicating whether this instance is equal to a specified object.
GetHashCode	Serves as a hash function for Resource type.
GetType	Gets the type of the current instance.

ToString	Returns a string that represents the current object.
----------	------------------------------------------------------

### 4.3.2 Adding Resources to a Project

**Resources** property exposed by **Project** class represents the list of all the **Resource** objects in a project. This property can be used to add resources.

The following code snippet illustrates adding resources to a project.

[C#]

```
// Creating an instance of the Project
Project project = new Project();

// Creating resource
Resource resource = new Resource();
resource.Name = "Resource1";

// Adding resource to project
project.Resources.Add(resource);
```

[VB]

```
' Creating an instance of the Project
Dim project As Project = New Project()

' Creating resource
Dim resource As Resource = New Resource()
resource.Name = "Resource1"

' Adding resource to Project
project.Resources.Add(resource)
```

### 4.3.3 Writing Resources to a Project

**Resources** property exposed by **Project** class represents the list of all the **Resource** objects in a project. This property can be used to update resources in a project.

The following code shows how to write resources to a project.

**[C#]**

```
// Creating an instance of the Project
Project project = new Project();

// Creating resource
Resource resource = new Resource();
resource.Name = "Resource1";

// Adding resource to project
project.Resources.Add(resource);

// Calculating Resource IDs and UUIDs
project.CalculateResourceIDs();

// Saving the project
project.Save("Project With Resources.xml");
```

**[VB]**

```
' Creating an instance of the Project
Dim project As Project = New Project()

' Creating resource
Dim resource As Resource = New Resource()
resource.Name = "Resource1"

' Adding resource to Project
project.Resources.Add(resource)

' Calculating Resource IDs and UUIDs
project.CalculateResourceIDs()
```

```
' Saving the project  
project.Save(@"D:\Project With Resources.xml")
```

## 4.4 Assignment

Assignment class is used to bind resources to tasks. It can also be used to retrieve the details of the task and resource that are bound together.

### 4.4.1 Properties, Methods, and Events Tables for Task

#### 4.4.1.1 Constructors

Table 13: Assignment Constructor

Name	Description
Assignment.Assignment()	Initializes a new instance of Assignment class.

#### 4.4.1.2 Properties

Table 14: Assignment Properties

Property	Description
UID	Gets or sets the unique identifier of the assignment.
TaskUID	Gets or sets the unique identifier of the task.
ResourceUID	Gets or sets the unique identifier of the resource.
PercentWorkComplete	Gets or sets the amount of work completed on the assignment.
ActualCost	Gets or sets the actual cost incurred on the assignment.



Property	Description
ActualFinish	Gets or sets the actual finish date of the assignment.
ActualOvertimeCost	Gets or sets the actual overtime cost incurred on the assignment.
ActualOvertimeWork	Gets or sets the actual amount of overtime work incurred on the assignment.
ActualStart	Gets or sets the actual start date of the assignment.
ActualWork	Gets or sets the actual amount of work incurred on the assignment.
ACWP	Gets or sets the actual cost of work performed on the assignment to-date.
Confirmed	Whether the Resource has accepted all of his or her assignments.
Cost	Gets or sets the projected or scheduled cost of the assignment.
CostRateTable	Gets or sets the cost rate table used for the assignment.
CostVariance	Gets or sets the difference between the cost and baseline cost for a resource.
CV	Gets or sets the earned value cost variance.
Delay	Gets or sets the amount that the assignment is delayed.
Finish	Gets or sets the scheduled finish date of the assignment.
FinishVariance	Gets or sets the variance of the assignment finish date from the baseline finish date.
Hyperlink	Gets or sets the title of the hyperlink associated with the assignment.
HyperlinkAddress	Gets or sets the hyperlink associated with the assignment.
HyperlinkSubAddress	Gets or sets the document bookmark of the hyperlink associated with the assignment.

Property	Description
WorkVariance	Gets or sets the variance of assignment work from the baseline work as minutes x 1000.
HasFixedRateUnits	Whether the Units are Fixed Rate.
FixedMaterial	Whether the consumption of the assigned material resource occurs in a single, fixed amount.
LevelingDelay	Gets or sets the delay caused by leveling.
LevelingDelayFormat	Gets or sets the format for expressing the duration of the delay.
LinkedFields	Whether the Project is linked to another OLE object.
MileStone	Whether the assignment is a milestone.
Notes	Gets or sets the text notes associated with the assignment.
Overallocated	Whether the assignment is over allocated.
OvertimeCost	Gets or sets the sum of the actual and remaining overtime cost of the assignment.
OvertimeWork	Gets or sets the scheduled overtime work scheduled for the assignment.
PeakUnits	Gets or sets the largest number of units that a resource is assigned for a task.
RateScale	Gets or sets the time unit for the usage rate of the material resource assignment.
RegularWork	Gets or sets the amount of non-overtime work scheduled for the assignment.
RemainingCost	Gets or sets the remaining projected cost of completing the assignment.
RemainingOvertimeCost	Gets or sets the remaining projected overtime cost of completing the assignment.
RemainingOvertimeWork	Gets or sets the remaining overtime work scheduled to complete the assignment.
RemainingWork	Gets or sets the remaining work scheduled to complete the assignment.

Property	Description
ResponsePending	Whether a response has been received for a TeamAssign message.
Start	Gets or sets the scheduled start date of the assignment.
Stop	Gets or sets the date that the assignment was stopped.
Resume	Gets or sets the date that the assignment resumed.
StartVariance	Gets or sets the variance of the assignment start date from the baseline start date.
Summary	Whether the task is a summary task.
SV	Gets or sets the earned value schedule variance, through the project status date.
Units	Gets or sets the number of units for the assignment.
UpdateNeeded	Whether the resource assigned to a task needs to be updated as to the status of the task.
VAC	Gets or sets the difference between baseline cost and total cost.
Work	Gets or sets the amount of scheduled work for the assignment.
WorkContour	Gets or sets the work contour of the assignment.
BCWS	Gets or sets the budgeted cost of work on the assignment.
BCWP	Gets or sets the budgeted cost of work performed on the assignment to-date.
BookingType	Gets or sets the booking type of the assignment.
ActualWorkProtected	Gets or sets the duration through which actual work is protected.
ActualOvertimeWorkProtected	Gets or sets the duration through which actual overtime work is protected.
CreationDate	Gets or sets the date the assignment was created.

Property	Description
AssnOwner	Gets or sets the name of the assignment owner.
AssnOwnerGuid	Gets or sets the GUID of the assignment owner
BudgetCost	Gets or sets the budgeted amount for cost resources on this assignment.
BudgetWork	Gets or sets the budgeted work amount for work or material resources on this assignment.
ExtendedAttribute	Gets or sets the value of an extended attribute.
Baseline	Gets or sets the collection of baseline values associated with the assignment.
F404000 – F4040C8	Gets or sets Project 2010 assignment enterprise custom field elements.
TimePhasedData	Gets or sets the time phased data associated with the assignment.

#### 4.4.1.3 Methods

Table 15: Assignment Methods

Method	Description
Equals	Returns a value indicating whether this instance is equal to a specified object
GetHashCode	Serves as a hash function for Assignment type
GetType	Gets the type of the current instance
ToString	Returns a string that represents the current object

#### 4.4.2 Adding Assignments to a Project

**Assignments** are used to bind the task and resources. **Project** class exposes **Assignments** collection that represents the list of all the **Assignment** objects in a project. This property can be used to add assignment.

The code below shows adding assignments to a project:

**[C#]**

```
// Creating an instance of the Project
Project project = new Project();

// Creating a task
Task task = new Task();
task.Name = "Task1";

// Adding the task to project
project.RootTask.Children.Add(task);

// Calculating Task ID
project.CalculateTaskIDs();

// Creating a resource
Resource resource = new Resource();
resource.Name = "Resource1";

// Adding resource to project
project.Resources.Add(resource)

// Calculating Resource ID
project.CalculateResourceIDs()

// Creating an instance of Assignment
Assignment assignment = new Assignment();
assignment.UID = 1;

// Assigning resource to task
assignment.Task = task;
assignment.Resource = resource;

// Adding resource to project
```

```
project.Resources.Add(resource);
```

**[VB]**

```
' Creating an instance of Project
Dim project As Project = New Project()

' Creating an instance of Task
Dim task As Task = New Task()
task.Name = "Task1"

' Adding the task to project
project.RootTask.Children.Add(task)

' Calculating Task ID
project.CalculateTaskIDs()

' Creating an instance of Resource
Dim resource As Resource = New Resource()
resource.Name = "Resource1"

' Adding resource to project
project.Resources.Add(resource)

' Calculating Resource ID
project.CalculateResourceIDs()

' Creating an instance of Assignment
Dim assignment As Assignment = New Assignment()
assignment.UID = 1

' Assigning tasks and resource to assignment
assignment.Task = task
assignment.Resource = resource

' Adding an assignment to a project
```

```
project.Assignments.Add(assignment)
```

The project created using the above code will look as shown in the following Microsoft Project screenshot.

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Name	Oct 25, '11							Oct 2, '11							Oct 9, '11							Oct 16, '11							
								M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M
1		Task1	1 day?	Fri 10/7/11	Fri 10/7/11		Resource1[8]																													
																														</						



## 4.5 Calendar

Calendar class is used to create calendars and add them to the project. Using the Calendar class, one can create calendar exceptions (holidays, different working times and working days), define working days, working times, and so on. It can also add a standard calendar to the project. Calendar class contains properties that can be used to retrieve information of all calendars present in a project.

### 4.5.1 Properties, Methods, and Events Tables for Task

#### 4.5.1.1 Constructors

Table 16: Calendar Constructors

Name	Description
Calendar.Calendar()	Initializes a new instance of Calendar class.
Calendar.Calendar(string calendarName)	Initializes a new instance of Calendar class with the calendar name.

#### 4.5.1.2 Properties

Table 17: Calendar Properties

Property	Description
UID	Gets or sets the unique identifier of the calendar.
Name	Gets or sets the name of the calendar.
IsBaseCalendar	True if the calendar is a base calendar.
IsBaselineCalendar	True if the calendar is a baseline calendar.
BasecalendarUID	Gets or sets the unique identifier of the base calendar on which this calendar depends. Only applicable if the calendar is not a base calendar.
WeekDays	Gets or sets the collection of weekdays that defines this calendar.
Exceptions	Gets or sets the collection of exceptions that is associated with the calendar.
WorkWeeks	Gets or sets the collection of effective work weeks associated with the calendar.

### 4.5.1.3 Methods

Table 18: Calendar Methods

Method	Description
Equals	Returns a value indicating whether this instance is equal to a specified object
GetHashCode	Serves as a hash function for Calendar type
GetType	Gets the type of the current instance
ToString	Returns a string that represents the current object
Calendar.StandardCalendar()	Creates a standard calendar
Calendar.StandardCalendar(string calendarName)	Creates a standard calendar

### 4.5.2 Creating a Standard Calendar

The static method `StandardCalendar` is used to create a standard calendar and add it to the project.

This method contains two overloads namely:

- `StandardCalendar()` – Creates a standard calendar
- `StandardCalendar(string calendarName)` – Creates a standard calendar by passing the calendar name

The following code snippet shows how to make use of this method:

[C#]

```
// Creating a standard calendar
Calendar calendar = Calendar.StandardCalendar();

// Creating a standard calendar by passing the calendar name
Calendar calendar1 = Calendar.StandardCalendar("Standard");
```

[VB]

```
' Creating a standard calendar
Dim calendar As Calendar = Calendar.StandardCalendar()
```

```
' Creating a standard calendar by passing the calendar name  
Dim calendar1 As Calendar = Calendar.StandardCalendar("Standard")
```

## 5 How To

---

### 5.1 Does ProjIO require MS Project to be installed on the machine?

No. ProjIO is a 100% Native .Net Library that does not depend on MS Project.

### 5.2 Will it be possible to view the generated project [.xml file] using ProjIO?

No. ProjIO does not have a Graphical User Interface to view the generated project files. MS Project is required to view the project files generated by using ProjIO.

### 5.3 Can Essential ProjIO be used to read MS Project files?

Essential ProjIO provides support for reading MS Project XML format files. Currently ProjIO does not support reading and writing files in .mpp or .mpt format.

### 5.4 How to retrieve tasks from a project?

The tasks present in a project can be retrieved using the **GetTaskByUID** method.

The following code snippet illustrates retrieving tasks using this method:

```
[C#]

// Opening the project file
Project project = ProjectReader.Open(@"D:\ProjectWithTasks.xml");

// Retrieving a task by UID
Task task = project.GetTaskByUID(2);

// Viewing retrieved task information
Console.WriteLine("Task Name: " + task.Name);
Console.WriteLine("Task Start Date: " + task.Start);
Console.WriteLine("Task Finish Date: " + task.Finish);
Console.WriteLine("No. of Sub Tasks: " + task.Children.Count);
```

**[VB]**

```
' Opening the project file
Dim project As Project = ProjectReader.Open("ProjectWithTasks.xml")

' Retrieving a task by UID
Dim task As Task = project.GetTaskByUID(2)

' Viewing retrieved task information
Console.WriteLine("Task Name: " + task.Name)
Console.WriteLine("Task Start Date: " + task.Start)
Console.WriteLine("Task Finish Date: " + task.Finish)
Console.WriteLine("No. of Sub Tasks: " + task.Children.Count)
```

## 5.5 How to retrieve resources from a project?

The resources present in a project can be retrieved using the **GetResourceByUID** method.

The following code snippet illustrates how to retrieve tasks using this method:

**[C#]**

```
// Opening the project file
Project project = ProjectReader.Open("ProjectWithResources.xml");

// Retrieving a resource by UID
Resource resource = project.GetResourceByUID(1);

// Viewing retrieved resource information
Console.WriteLine("Resource Name: " + resource.Name);
Console.WriteLine("Type: " + resource.Type);
Console.WriteLine("Work: " + resource.Work);
Console.WriteLine("Remaining Work: " + resource.RemainingWork);
Console.WriteLine("Resource Calendar ID: " + resource.CalendarUID);
```

**[VB]**

```
' Opening the project file
Dim project As Project = ProjectReader.Open("ProjectWithResources.xml")

' Retrieving a resource by UID
Dim resource As Resource = project.GetResourceByUID(1)

' Viewing retrieved resource information
Console.WriteLine("Resource Name: " + resource.Name)
Console.WriteLine("Type: " + resource.Type)
Console.WriteLine("Work: " + resource.Work)
Console.WriteLine("Remaining Work: " + resource.RemainingWork)
Console.WriteLine("Resource Calendar ID: " + resource.CalendarUID)
```

## 5.6 How to retrieve resource assignments from a project?

The resource assignments present in a project can be retrieved using the **GetAssignmentByUID** method.

The following code snippet shows how to use this method:

**[C#]**

```
// Opening the project file
Project project = ProjectReader.Open("SampleProject.xml");

// Retrieving an assignment by UID
Assignment assignment = project.GetAssignmentByUID(1);

//Viewing retrived assignment information
Console.WriteLine("Resource: " + assignment.Resource.Name);
Console.WriteLine("Assigned to: " + assignment.Task.Name);
Console.WriteLine("Booking Type: " + assignment.BookingType);
Console.WriteLine("Cost: $" + assignment.Cost);
```

**[VB]**

```
' Opening the project file
Dim project As Project = ProjectReader.Open("SampleProject.xml")

' Retrieving a resource by UID
Dim assignment As Assignment = project.GetAssignmentByUID(1)

' Viewing retrieved assignment information
Console.WriteLine("Resource: " + assignment.Resource.Name)
Console.WriteLine("Assigned to: " + assignment.Task.Name)
Console.WriteLine("Booking Type: " + assignment.BookingType)
Console.WriteLine("Cost: $" + assignment.Cost)
```

## **Index**

### **A**

Adding Assignments to a Project 52  
Adding Resources to a Project 46  
Adding Tasks to a Project 36  
Assignment 48

### **C**

Calendar 57  
Can Essential ProjIO be used to read MS Project files? 60  
Concepts and Features 14  
Constructors 14, 30, 41, 48, 57  
Creating a simple project 18  
Creating a Standard Calendar 58  
Creating a summary task 37  
Creating Task links 38

### **D**

Default Project Properties 22  
Deployment Procedures 12  
DLLs 12  
Documentation 6  
Does ProjIO require MS Project to be installed on the machine? 60

### **F**

Feature Summary 13  
Fiscal Year Properties 26

### **G**

General Project Properties 20  
Getting Started 13

### **H**

How To 60  
How to retrieve tasks from a project? 60

### **I**

Installation 7  
Installation and Deployment 7  
Introduction to Essential ProjIO 5

### **M**

Methods 17, 36, 45, 52, 58

### **O**

Overview 5

### **P**

Prerequisites and Compatibility 5  
Project 14  
Properties 14, 31, 41, 48, 57  
Properties, Methods, and Events Tables for Project 14  
Properties, Methods, and Events Tables for Task 30, 41, 48, 57

### **R**

Reading a project file 19  
Resource 41  
Retrieving Default Project Properties 22  
Retrieving Fiscal Year Properties 27  
Retrieving Project Properties 20  
Retrieving Week Day Properties 28

### **S**

Sample Installation Location 7  
Setting Default Project Properties 23  
Setting Fiscal Year Properties 27  
Setting Project Properties 21  
Setting Week Day Properties 29

### **T**

Task 30

### **U**

Use Case Scenario 5

### **V**

Viewing Samples 7



**W**

Week Day Properties 28

Where to Find Samples? 7

Will it be possible to view the generated project  
[.xml file] using ProjIO? 60

Writing Project Summary Information 25

Writing Resources to a Project 46

Writing Tasks to Projects 39