Deltek

Deltek Open Plan® 8.7

Developer Guide

March 8, 2024

Revised: August 28, 2026

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published August 2026.

© Deltek, Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties.

All trademarks are the property of their respective owners.

Contents

Overview 1
Integration 2
Open Plan OLE Automation 2
Objects 2
Collections 2
Properties 2
Methods 3
Accessing OLE Automation Objects 3
Getting an Object Property 3
Setting an Object Property 4
Executing an Object Method 4
Modifying Existing Applications for Use with Open Plan 8.x 4
The Open Plan Object Hierarchy 5
Objects and Collections Properties 11
Properties 33
Methods 99
Examples 189
Import and Export Facilities 197
Integration with Microsoft Project 197
Integration with Primavera P6 207
Integrating with IPMDAR 210
Custom Import/Export Options 210
Custom Import/Export Scripts 212
Importing and Exporting using XML 226
Field Mappings Between MSP and Open Plan 234
Extensibility 240
Adding Utilities to the Open Plan User Interface 240
Tool1… 240
Defining Calculated Fields 242
The Elements of Calculated Field Expressions 243
Calculated Field Functions Reference 248
Standard Calculated Fields 273
Examples of Custom Calculated Fields 304

Processes 308
Time Analysis Calculations 308
Topological Sorting 308
The Calculation of Early Dates 308
The Calculation of Late Dates 311
Float Calculations 312
Criticality 313
Time Analysis and Subprojects 313
Hammock Activities 315
Link/Unlink Activities 315
Logic Trace 315
Schedule Driver Analysis 316
The Effect of Progress Information 317
The Effect of Progress on Time Analysis 317
Activity Status and Remaining Duration 317
Special Cases of Progress Information 319
The Effect of Progress on Risk Analysis 321
The Effect of Progress on Resource Scheduling 322
The Effect of Resource Progressing 323
The Splitting of In-Progress Activities 324
Resource Scheduling Calculations 324
Resource Scheduling Methods 325
The Effect of Activity Attributes 325
The Effect of Resource Attributes 329
Scheduling Intervals and Mixed Time Units 330
Resource Assignment Profiles 332
The Effect of Processing Options 333
Priorities in Resource Scheduling 338
Resource Substitution 340
The Effect of Reservations on Resource Scheduling 342
Resource Usage by Other Projects 343
The Interpretation of Resource Scheduling Results 344
Risk Analysis Calculations 345
The Concept of Probability 346
Interpreting Risk Analysis Data 355
Operating Characteristics of Risk Analysis 357

The Beta Distribution 362
Cost Calculations 364
Cost Data 364
Resource Cost Table (CST) Records 367
Cost Calculations 368
Physical Percent Complete (PPC) 373
Calculated Fields 375
Multi-Project Operations 376
External Subprojects 377
The Effect of the Save As Command 377
Interproject Relationships 377
Project Target Dates 378
Data Merging Issues 378
Baseline Issues 378
Cost Calculations 379
Reporting 380
Customizing Barchart Views 380
Defining Bar Sets 380
Displaying Time-Phased Data 388
Customizing Network Views 393
Custom Box Attributes 394
Combining Box Attributes 395
Custom Box Text 398
Conditional Formatting 400
Customizing Resource Histogram Views 401
Selecting Resources 401
Resource Histogram Display Options 403
Customizing the Date Scale 405
Customizing Risk Histogram Views 406
Selecting a Key Activity 406
Risk Histogram Display Options 406
Customizing the Date Scale 407
Customizing the Web Publisher 407
Customizing Headers and Footers 407
Utilities 409
Open Plan Configuration Files 409

Working with Dockable Web Windows 409
[Open Plan Professional] 409
[System] 410
[SSCE] 411
Windows Registry Entries 411
Sample Tools 412
Open Plan Batch Processor 426
Datasources.dat File 426
Briefcase.dat File 426
Spread Curve Profiles 427
Back Load Profile 427
Double Peak Profile 428
Early Peak Profile 428
Front Load Profile 429
Late Peak Profile 429
Bell (Normal) Profile 429
Linear Profile 430
Date Formats in Open Plan 430
Defining Date Formats 431
Project Utilities 433
Filters 433
Sorts 442
Rollups 449
Global Edits 451
User Defined Validations 457
Calculated Fields 461
Spread Curves 498
Frequently Asked Questions 504
Spreadsheet Views 504
Barchart Views 504
Histogram Views 506
Resources 507
Calculated Fields 508
Durations 508
Miscellaneous 509
Appendix A: Create/Update Baseline Option Flags 510

Appendix B: Deltek Open Plan Data Structure 513
System Data 513
Configuration Data 513
Directory Data 526
Security Data 537
View Data 545
User Data 548
Baseline Data 548
Calendar Data 560
Code Data 561
Explorer Data 564
Project Data 566
Resource Data 584
User Defined Fields 589
User Validations 593
Common FW Tables not used by Open Plan 595
Table FW_CFGProcessQueue 595
Table FW_CFGProcessQueueSetup 595
Table FW_DetailJobs 596
Table FW_DetailJobStatus 597
Table FW_DetailProfiles 597
Table FW_MasterJobs 598
Table FW_MasterProfiles 600
Appendix C: If You Need Assistance 603
Customer Services 603
Deltek Support Center 603
Access Deltek Support Center 604
Additional Documentation 604

Overview

With Deltek Open Plan's project management tools, you can provide your organization with reliable estimates and solutions for long and short-term resource and budget allocations. Simplify manual processes with rapid data entry, quickly share and update project plan information through Project Portfolio Management (PPM) integration, and ensure projects get the resources they need.

Integration

Open Plan OLE Automation

OLE automation is a feature of the Windows operating systems family that allows different Windows applications to share data and functionality. OLE automation lets you easily extend the capabilities of Open Plan with powerful, low-maintenance solutions designed with the tools that best suit your needs and abilities.

One of the most powerful features in Open Plan is its support of the robust and flexible technology of OLE automation. Open Plan is designed as an OLE server application, exposing a variety of objects, properties, and methods for use with other Windows applications such as Visual FoxPro, Visual Basic, Microsoft Excel, and many others.

This document discusses the Open Plan object hierarchy and contains information about objects and collections, properties, and methods. The document concludes with tables of Open Plan date formats and examples of incorporating OLE automation in your project.

Objects

The fundamental unit of data or functionality exposed by a server application is called an object. Objects can have constituent parts that are themselves objects. Objects in Open Plan can include the following items:

- The Open Plan application
- A project or collection of projects
- An activity or collection of activities
- Resource definitions or assignments
- Project codes and code files
- Calendar information

Collections

Some objects are collections of other objects. The names of collection objects are usually the plural form of the names of the objects they contain. These collection objects can be manipulated in the same way as other objects. Most collections provide methods for retrieving, counting, adding, or deleting individual members from the collection. Collection objects in Open Plan can include:

- A collection of activity records
- A collection of barchart views
- A collection of resource description records

Properties

Each object created in Open Plan has a set of properties that represent the specific attributes of that object. Some objects have values that can be changed while others have values that can only be retrieved. The following are examples of object properties:

- A field in an activity record
- The number of activities in a project

- The state of a window or view

Methods

Many objects in Open Plan have methods as well as properties. Unlike object properties, which are attributes that you can set or return, object methods are actions that you instruct the object to perform. Methods available for objects in Open Plan include:

- Adding an activity
- Performing time analysis
- Displaying a view
- Opening a project
- Minimizing a window
- Setting a filter

Accessing OLE Automation Objects

Open Plan OLE Automation objects can be easily accessed with a variety of programming environments and client applications. The decision of which tool to use is left solely to the user. Deltek's OLE sample code is written in Microsoft's Visual Basic, which is an excellent tool for both beginners and experienced users to create simple, yet powerful, OLE client applications. This section describes the steps and syntax for accessing Open Plan OLE Automation objects with Visual Basic.

The first step for an external application to access objects belonging to Open Plan is to get the Open Plan application object. The following Visual Basic code is an example of retrieving the Open Plan application object. (In Visual Basic, a line of code that begins with an apostrophe is a comment.)

```
'Declare the Open Plan Application Object variable
Dim OPCreateApplication33 As Object
'Assign the Application object variable
Set OPCreateApplication33 = CreateObject("opp.application")
```

After retrieving the Open Plan application object, its properties and methods then can be accessed in the following ways:

- Getting a property of an object
- Setting a property of an object
- Executing a method of an object

Getting an Object Property

A property of an object is retrieved by setting a variable equal to the object variable, followed by a period (.), and followed by a property name.

Syntax: Variable = ObjectVariable.PropertyName

Example:
```
Dim MyActivities As Object
Dim ActivityCount As Long
ActivityCount = MyActivities.Count
```

Setting an Object Property

A property of an object is set by setting the object variable, followed by a period (.), and followed by a property name equal to a variable that represents the values of the property.

Syntax: ObjectVariable.PropertyName = Variable

Example:
```
Dim MyActivities As Object
Dim ActID As String
ActID = MyActivities.ID
```

Executing an Object Method

An OLE Automation method can either use no parameters, or one or more parameters. It can return either nothing or a value. For methods that return a value, a variable can be assigned to the returning value by placing the variable on the left side of the equal sign (=). Some methods return objects. For these methods, the returned value is set to an object variable.

Methods with no parameters

Syntax: ObjectVariable.MethodName

Example: MyCodeView.Minimize

Methods with one or more parameters

Syntax: ObjectVariable.MethodName Parameter1, Parameter2

Example: MyProject.ResourceSchedule “Time Limited”, True

Methods that return a value

Syntax: Variable = ObjectVariable.MethodName(Parameter1, Parameter2)

Example: SuppressInListsFlag = AResource.Getfield(“NO_LIST”)

Methods that return an object

Syntax: ObjectVariable1 = ObjectVariable2.MethodName.Item(Parameter1)

Example: Set MyCalendarFile = OPCreateApplication33.Calendars.Item(“MYCAL”)

Modifying Existing Applications for Use with Open Plan 8.x

Because of features that were added to Open Plan 8.x, some applications may have to be modified before you can use them with Open Plan.

Objects, Properties, and Methods No Longer Supported

Some objects, properties, and methods are no longer supported in Open Plan 8.x. Objects not supported include:

- OPFCProcesses
- OPFCStructures

Properties no longer supported in Open Plan 8.x are:

- OPPFormat
- ResidueFile

Methods not supported in Open Plan 8.x are:

- FileOpenSpecialOpenPlan
- OpenSpecialODBC
- GlobalDir
- Rights
- Processes
- Structures

Naming Conventions

Applications using early binding need to be modified to use the new application object's name. OPCreateApplication is also referred to as OPApplication object in this guide.

Security Features

A security feature added to Open Plan 8.x requires users to login upon launching the application. For this reason, automation applications that are intended to launch Open Plan must call the Login method immediately after instantiating the Open Plan application.

If a user's security permissions in Open Plan do not permit the use of a given feature, that feature will not be available for use by automation applications. Instead of disabling tab commands in Deltek EPM Security Administrator, set the menu items to invisible and leave them enabled. The user will not be able to access them, but automation applications will.

Add-Ins Tab

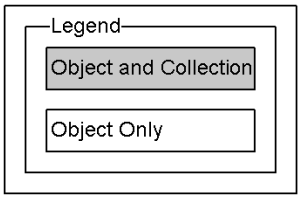
In Open Plan 8.x, more parameters are available to use with applications added to the **Add-Ins** tab. These parameters can be used to return context-specific data to your application or to control how an **Add-Ins** tab tool is launched.

The data returned by Open Plan's special command line parameters in the **Add-Ins** tab is wrapped in quotes. It may be necessary to modify your applications to handle this.

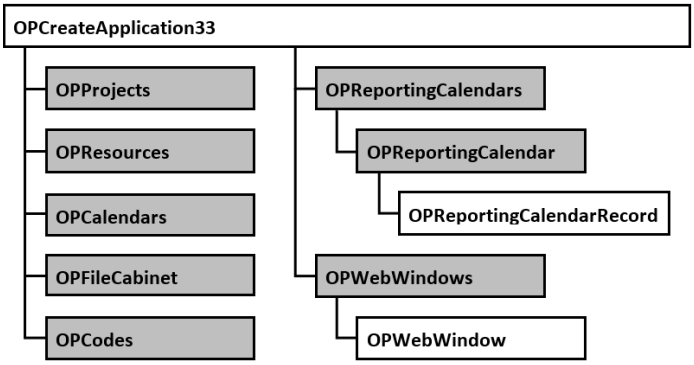
> **Attention:** For a list of the command line parameters and information about the **Add-Ins** tab, see Open Plan Configuration Files.

The Open Plan Object Hierarchy

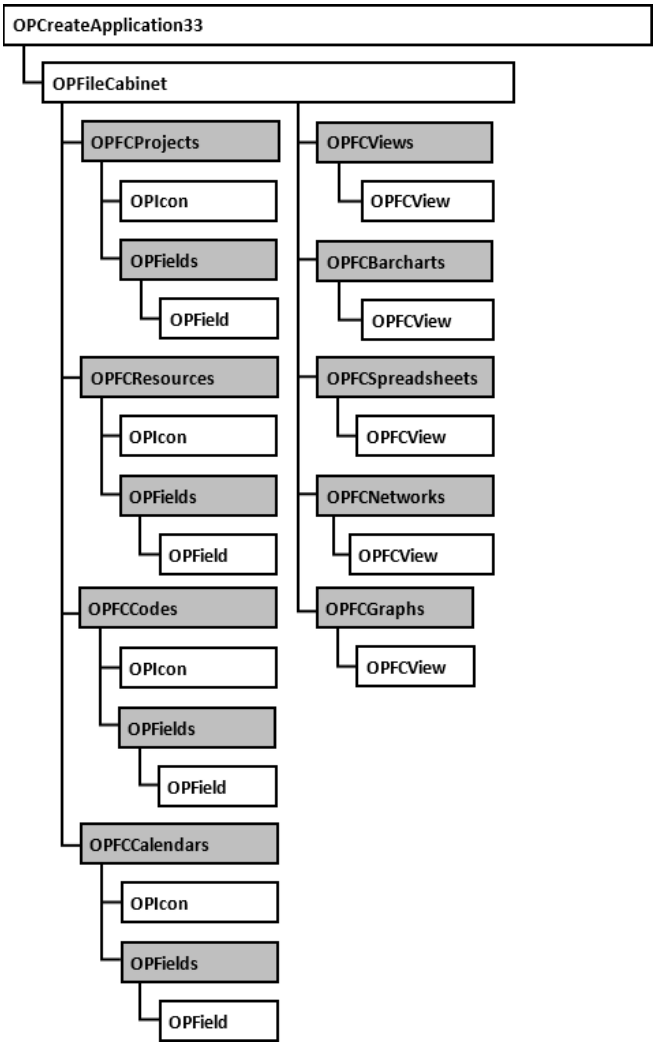
The following diagrams illustrate the hierarchy of objects in Open Plan. The diagrams use the legend:



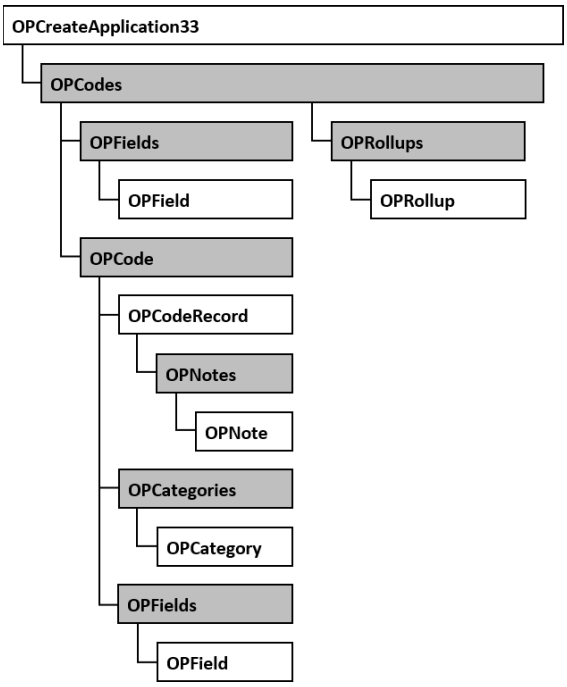
The OPCreateApplication33 Object Hierarchy



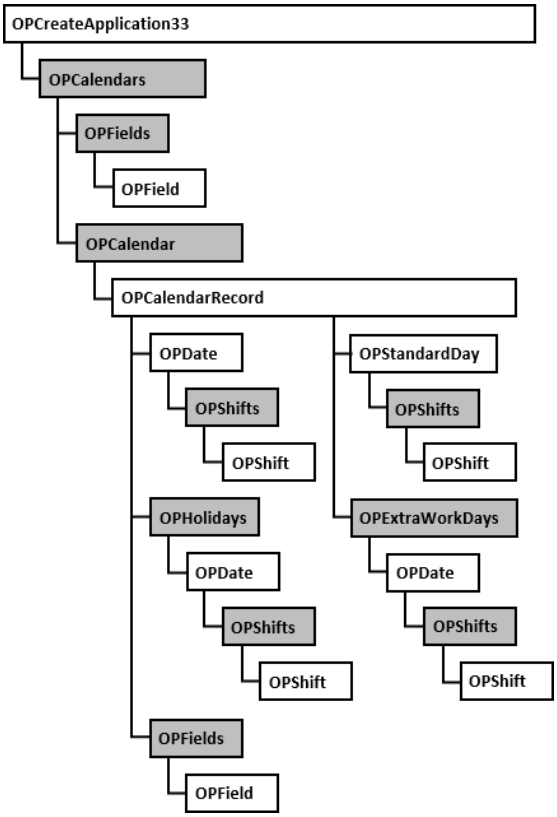
The OPFileCabinet Object Hierarchy



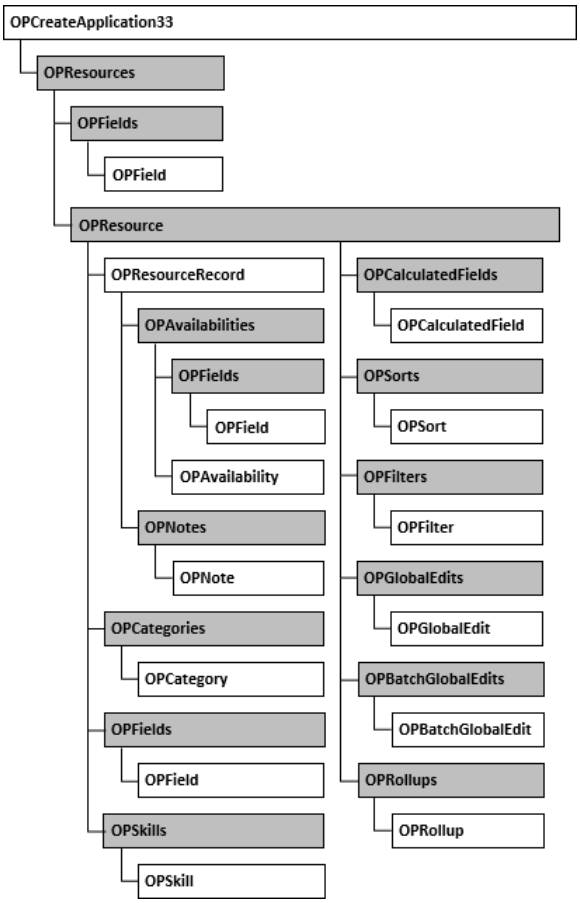
The OPCodes Object Hierarchy



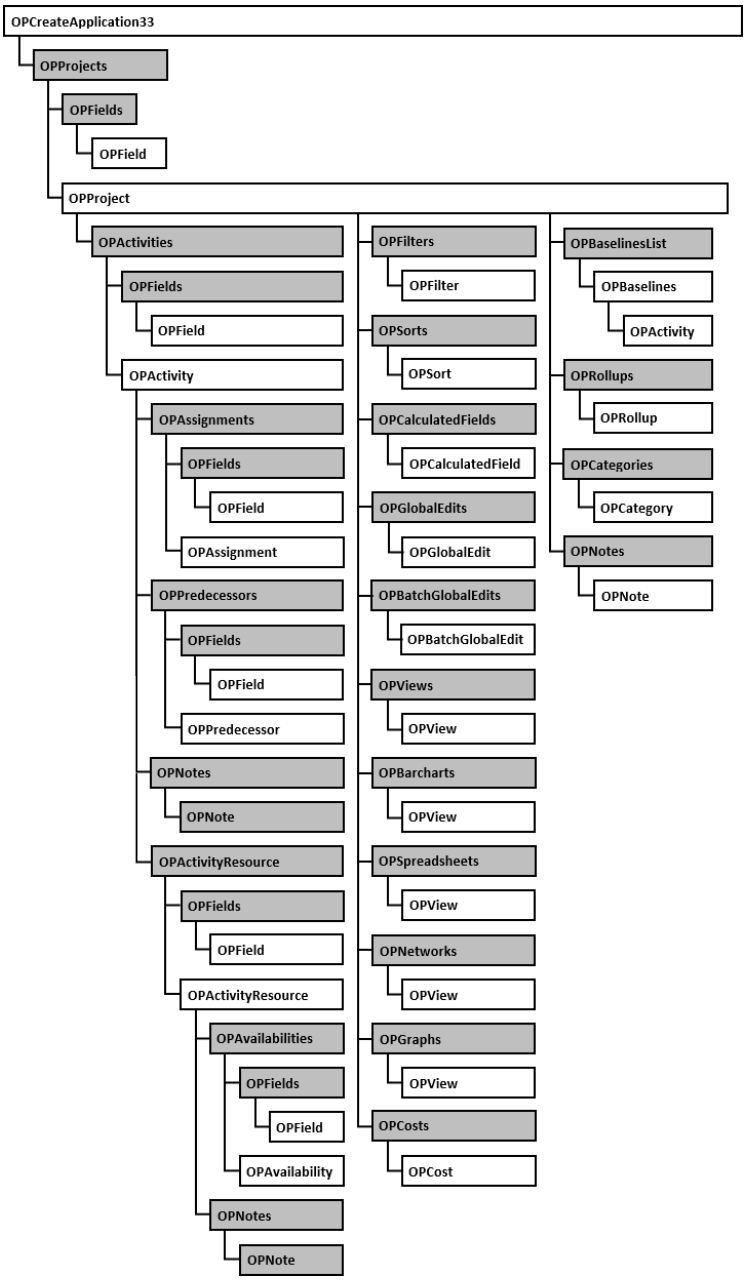
The OPCalendar Object Hierarchy



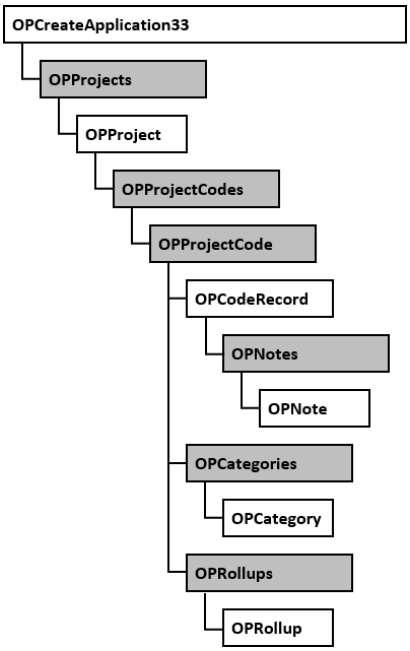
The OPResources Object Hierarchy



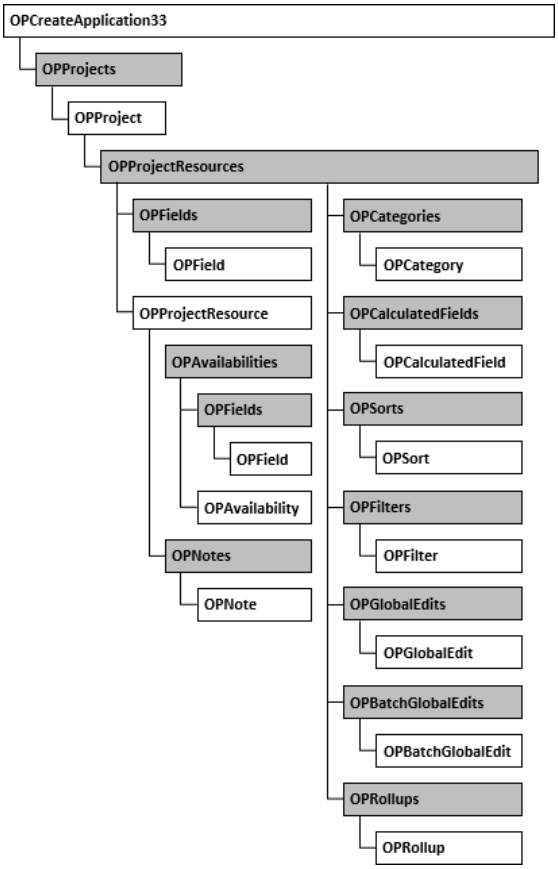
The OPProjects Object Hierarchy



The OPProjectCodes Object Hierarchy



The OPProjectResources Object Hierarchy



Objects and Collections Properties

OPAccessControlList Collection

Description The **OPAccessControlList** collection contains the list of all OPAccess Control List objects.

Remarks The **OPAccessControlList** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPAccessControlObject

Description The **OPAccessControl** object uses one of the methods listed below to obtain an **OPAccessControlList** collection or object. If the method is called without parameters, it returns an **OPAccessControlList** collection. If the method is used with an index parameter, it returns an **OPAccessControlObject**.

Remarks The **OPAccessControl** object contains the following methods:

Methods	
GetRights	SetRights

OPActivities Collection

Description The **OPActivities** collection contains the list of all **OPActivity** objects belonging to the parent **OPProject** object.

Remarks The **OPActivities** collection object has the following properties and methods:

Properties		
Count	Filter	Sort
Methods		
Add	Remove	SetRiskFields
AssignCurrentFieldSet	SetAssignmentFields	SetSortFields
Categories	SetCalculatedFieldTo	SetSortTo
Fields	SetCollectionGrowth	SetStepFields
GetCalculatedFieldString	SetCostFields	SetSuccFields
GetFilterString	SetFilterTo	SetUsageFields
Item	SetPredFields	Steps

OPActivity Object

Description When retrieved from the OPActivities collection, the **OPActivity** object contains information about a single activity within an open project. When retrieved from the **OPBaselines** collection, the **OPActivity** object contains information about a single baseline activity within a saved baseline belonging to an open project.

Remarks The **OPActivity** object contains the following properties and methods:

Properties		
ActivityLogicFlag	EarlyStartStdDev	ProgressValue

Properties		
ActualFinishDate	ExpectedFinish	RefreshData
ActualStartDate	FinishFreeFloat	ResourceScheduleType
BaselineFinish	FinishTotalFloat	ScheduleActions
BaselineStart	FirstUsage	ScheduledFinish
BudgetCost	FreeFloat	ScheduledStart
Calendar	FreeFloatStdDev	ScheduleDuration
ComputedRemainingDuration	ID	ScheduleFloat
ComputedStatus	KeyActivityStatus	SubprojectFilename
CostToDate	LateFinish	SuppressRequirements
Critical	LateFinishStdDev	TargetFinish
CriticalIndex	LateStart	TargetFinishType
DelayingResource	LateStartStdDev	TargetStart
Description	MaxDuration	TargetStartType
Duration	MaxNoSplits	TotalFloat
DurationDistShape	MinSplitLength	TotalFloatStdDev
EarliestFeasible	OptimisticDuration	TotalResourceCost
EarlyFinish	PercentComplete	Type
EarlyFinishStdDev	PessimisticDuration	
EarlyStart	ProgressFlag	
Methods		
ActivityResources	IsExternalOpen	Selected
Assignments	Notes	SetCurrentFields
GetCurrentFields	Predecessors	SetField
GetField	Remove	Steps
GetFields	ResourceUsages	

OPActivityResource Object

Description The **OPActivityResource** object represents a resource assigned to a specific activity in the current project.

Remarks The **OPActivityResource** object has the following methods:

Properties		
ActivityID	RefreshData	Type
Class	RollUp	UnitCost
Description	Suppress	Units
ID	Threshold	
Methods		
Availabilities	GetFields	Remove
GetCurrentFields	GetResourceCrosstabData	SetCurrentFields
GetEarnedValueCrosstabData	GetResourceCrosstabDataInXML	SetField
GetEarnedValueCrosstabDataInXML	GetResourceDateArray	
GetField	Notes	

OPActivityResources Collection

Description The **OPActivityResources** collection contains the list of all **OPActivityResource** objects belonging to the parent **OPActivity** object.

Remarks The **OPActivityResources** collection object has the following properties and methods:

Properties		
Count		
Methods		
AssignCurrentFieldSet	Fields	Item

OPAssignment Object

Description The **OPAssignment** object contains properties and methods for an assignment record within the **OPAssignments** collection.

Remarks The **OPAssignment** object has the following properties and methods:

Properties		
ActivityID	RefreshData	ResourcePeriod
AlternateResourceID	Remaining	
Level	ResourceID	
LevelType	ResourceOffset	
Methods		
GetCurrentFields	Getfields	Remove
GetEarnedValueCrosstabData	GetResourceCrosstabData	SetCurrentFields
GetEarnedValueCrosstabDataInXML	GetResourceCrosstabDataInXML	Setfield
Getfield	GetResourceDateArray	

OPAssignments Collection

Description The **OPAssignments** collection contains the list of all **OPAssignment** objects belonging to the parent **OPActivity** object.

Remarks The **OPAssignments** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Fields	Remove
AssignCurrentFieldSet	Item	SetCollectionGrowth

OPAvailabilities Collection

Description The **OPAvailabilities** collection contains the list of all **OPAvailability** objects belonging to the parent **OPResourceRecord** object, **OPActivityResource** object, or **OPProjectResource** object.

Remarks The **OPAvailabilities** collection object has the following properties and methods:

Properties
Count
Methods

Add	Fields	Remove
AssignCurrentFieldSet	Item	

OPAvailability Object

Description The **OPAvailability** object contains properties and methods for an availability record within the **OPAvailabilities** collection.

Remarks The **OPAvailability** object has the following properties and methods:

Properties		
Calendar	RefreshData	StartDate
Level	Resource	StopDate
Methods		
GetCurrentFields	Getfields	SetCurrentFields
Getfield	Remove	Setfield

OPBarcharts Collection

Description The **OPBarcharts** collection contains the list of all barchart icons stored in the parent **OPProject** object. The members of this collection are **OPView** objects.

Remarks The **OPBarcharts** collection object has the following properties and methods:

Properties		
Count		
Methods		
ActivateByFilename	AddView	Item

OPBaselines Collection

Description The **OPBaselines** collection contains the list of all **OPActivity** objects belonging to the specified baseline.

Remarks The **OPBaselines** object has the following properties and methods:

Properties		
Count	Name	Type
Description	Selected	
Filter	Sort	
Methods		
Item		

OPBaselinesList Collection

Description The **OPBaselinesList** collection contains the list of all **OPBaselines** objects belonging to the parent **OPProject** object.

Remarks The **OPBaselinesList** collection object has the following properties and methods:

Properties	
Count	
Methods	
Item	Select

OPBatchGlobalEdit Object

Description The **OPBatchGlobalEdit** object contains properties for a batch global edit definition within the **OPBatchGlobalEdit s** collection.

Remarks The **OPBatchGlobalEdit** object has the following properties and methods:

Properties	Methods
Name	Apply

OPBatchGlobalEdits Collection

Description The **OPBatchGlobalEdits** collection contains the list of all **OPBatchGlobalEdit** objects belonging to the parent object.

Remarks The **OPBatchGlobalEdits** collection object has the following properties and methods:

Properties	Methods
Count	Item

OPCalculatedField Object

Description The **OPCalculatedField** object contains properties for a calculated field within the **OPCalculatedFields** collection.

Remarks The **OPCalculatedField** object has the following properties:

Properties		
CalcAcross	Expression	ResultType
Decimals	Name	References
TableName		

OPCalculatedFields Collection

Description The **OPCalculatedFields** collection contains the list of all **OPCalculatedField** objects belonging to the parent **OPProject**, **OPResource**, or **OPProjectResources** objects.

Remarks The **OPCalculatedFields** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPCalendar Collection

Description The **OPCalendar** collection represents a calendar file and contains a list of **OPCalendarRecord** objects.

Remarks The **OPCalendar** collection object has the following properties and methods:

Properties		
Count	Height	XPosition
Filename	Width	YPosition
Methods		
Add	GetFields	SaveAs
Conceal	Item	SetCollectionGrowth

Copy	Maximize	SetField
Disable	Minimize	Shut
Enable	Remove	ShutWOSave
Fields	Restore	
GetField	Save	

OPCalendarRecord Object

Description The **OPCalendarRecord** object contains properties and methods for a calendar record within an OP**Calendar** collection. It represents a named record within a calendar file.

Remarks The **OPCalendarRecord** object has the following methods:

Methods		
Date	GetStandardDays	SetStandardDays
ExtraWorkDays	Holidays	StandardDay
GetField	Name	
GetFields	SetField	

OPCalendars Collection

Description The **OPCalendars** collection contains the list of all open calendar files.

Remarks The **OPCalendars** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPCategories Collection

Description The **OPCategories** collection contains the list of all **OPCategory** objects belonging to the parent object. The following objects contain a Categories collection: **OPProject**, **OPResource**, **OPCode**, **OPProjectResources** and **OPProjectCode**.

Remarks The **OPCategories** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPCategory Object

Description The **OP Category** object contains properties for a category within the **OP Category** collection.

Remarks The **OPCategory** object has the following properties:

Properties
Name

OPCode Collection

Description The **OPCode** collection represents an open code file and contains a list of **OPCodeRecord** objects.

Remarks The **OPCode** collection object has the following properties and methods:

Properties		
Count	Height	XPosition
Filename	Width	YPosition
Methods		
Add	GetField	Rollups
AssignCurrentFieldSet	GetFields	Save
Categories	Item	SaveAs
Conceal	Maximize	SetCollectionGrowth
Disable	Minimize	SetField
Enable	Remove	Shut
Fields	Restore	ShutWOSave

To access only the code files belonging to a specific object, use the **OPProjectCodes collection.**

OPCodeRecord Object

Description The **OPCodeRecord** object contains properties and methods for a code record within an **OPCode** collection. It represents a named record within a code file.

Remarks The **OPCodeRecord** object has the following properties and methods:

Properties		
Code	Description	RefreshData
Methods		
GetCurrentFields	Notes	Setfield
GetField	Remove	
Getfields	SetCurrentFields	

OPCodes Collection

Description The **OPCodes** collection contains the list of all open code files.

Remarks The **OPCodes** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPCost Object

Description The **OPCost** object contains properties and methods for a cost record within the **OPCosts** collection.

Remarks The **OPCost** object has the following properties and methods:

Properties

ActivityID	EndDate	StartDate
ActualCost	RefreshData	
ActualQty	ResourceID	
Methods		
GetCurrentFields	Getfields	SetCurrentFields
GetField	Remove	Setfield

OPCosts Collection

Description The **OPCosts** collection contains the list of all **OPCost** objects belonging to the parent **OPProject** object.

Remarks The **OPCosts** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	SetCollectionGrowth
AssignCurrentFieldSet	Remove	

OPCreateApplication33 Object

Description The **OPCreateApplication33** object contains properties and methods for accessing the Open Plan application.

Remarks The **OPCreateApplication33** object has the following properties and methods:

Properties		
EnableUndo	SilentMode	Xposition
Height	Width	Yposition
Methods		
ActiveProject	FilePrintPreview	Restore
ActiveView	FilePrintSetup	SetResourceSelection
ApplyFilter	FileRename	SetPrinter
Calendars	GeneralExport	Show
Codes	GeneralImport	SysDir
Conceal	GetLastSecurityValidation	User
CreateBrowserView	Login	Version
Disable	Maximize	WebWindows
Enable	Minimize	WindowMinimizeAll
FileCabinet	Projects	WindowTile
FileNew	PageSetup	WindowTileVertical
FileOpen	ReportingCalendars	WorkDir
FileOpenEx	Resources	
FilePrint		

OPDate Object

Description The **OPDate** object contains properties and methods for a date record within a specified **OPCalendarRecord** object or within the **OPExtraWorkdays** or **OPHolidays** collections.

Remarks The **OPDate** object has the following properties and methods:

Properties		
Date	Work	
Methods		
Shifts		

OPExtraWorkDays Collection

Description The **OPExtraWorkDays** collection contains a set of **OPDate** objects that represent the extra work days that were defined for a specified **OPCalendarRecord** object.

Remarks The **OPExtraWorkDays** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPFCBarcharts Collection

Description The **OPFCBarcharts** collection contains the list of all barchart icons stored in the Open Plan Explorer (the **OPFileCabinet** object). The members of this collection are **OPFCView** objects.

Remarks The **OPFCBarcharts** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPFCCalendars Collection

Description The **OPFCCalendars** collection contains the list of all calendar file icons stored in the Open Plan Explorer. The members of this collection are **OPIcon** objects.

Remarks The **OPFCCalendars** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPFCCodes Collection

Description The **OPFCCodes** collection contains the list of all code file icons stored in the Open Plan Explorer. The members of this collection are **OPIcon** objects.

Remarks The **OPFCCodes** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPFCGraphs Collection

Description The **OPFCGraphs** collection contains the list of all histogram icons stored in the Open Plan Explorer (the **OPFileCabinet** object). The members of this collection are **OPFCView** objects.

Remarks The **OPFCGraphs** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPFCNetworks Collection

Description The **OPFCNetworks** collection contains the list of all network icons stored in the Open Plan Explorer (the **OPFileCabinet** object). The members of this collection are **OPFCView** objects.

Remarks The **OPFCNetworks** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPFCProjects Collection

Description The **OPFCProjects** collection contains the list of all project file icons stored in the Open Plan Explorer. The members of this collection are **OPIcon** objects.

Remarks The **OPFCProjects** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPFCResources Collection

Description The **OPFCResources** collection contains the list of all resource file icons stored in the Open Plan Explorer. The members of this collection are **OPIcon** objects.

Remarks The **OPFCResources** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPFCSpreadsheets Collection

Description The **OPFCSpreadsheets** collection contains the list of all spreadsheet icons stored in the Open Plan Explorer (the **OPFileCabinet** object). The members of this collection are **OPFCView** objects.

Remarks The **OPFCSpreadsheets** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPFCView Object

Description The **OPFCView** object represents a view icon in the Open Plan Explorer. The **OPFCView** object belongs to one of the following collections: **OPFCBarcharts**, **OPFCGraphs**, **OPFCNetworks**, **OPFCProcesses**, **OPFCSpreadsheets**, **OPFCStructures**, or **OPFCViews**.

Remarks The **OPFCView** object has the methods:

Methods	
Description	Name

OPFCViews Collection

Description The **OPFCViews** collection contains the list of all view icons stored in the Open Plan Explorer (the **OPFileCabinet** object). The members of this collection are **OPFCView** objects.

Remarks The **OPFCViews** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPField Object

Description The **OPField** object represents a column in an Open Plan data collection.

Remarks The **OPField** object has the following properties and methods:

Properties		
DBType	IsEditable	TableName
FieldName	Length	Type
FieldType	Scale	UserName

OPFields Collection

Description The **OPFields** collection contains the list of all fields available via the **OPField** object. The OPFields collection is accessible from the following collections: **OPActivities**, **OPActivityResources**, **OPAssignments**, **OPAvailabilities**, **OPCalendar**, **OPCalendars**, **OPCode**, **OPCodes**, **OPFCCalendars**, **OPFCCodes**, **OPFCProjects**, **OPFCResources**, **OPPredecessors**, **OPProjectResources**, **OPProjects**, **OPResource,** and **OPResources**.

Remarks The **OPFields** collection object has the following properties and methods:

Properties
Count
Methods
Item

GetFieldInfoInXMLFormat

OPFileCabinet Object

Description The **OPFileCabinet** object contains properties and methods for accessing the Open Plan File Cabinet.

Remarks The **OPFileCabinet** object has the following properties and methods:

Properties		
Height	XPosition	
Width	YPosition	
Methods		
Barcharts	Enable	Projects
Calendars	Graphs	Resources
Codes	Maximize	Restore
Conceal	Minimize	Spreadsheets
Disable	Networks	Views

OPFilter Object

Description The **OPFilter** object contains properties for a filter within the **OPFilters** collection.

Remarks The **OPFilter** object has the following properties:

Properties		
Expression	Name	References
TableName		

OPFilters Collection

Description The **OPFilters** collection contains the list of all **OPFilter** objects belonging to the parent **OPProject**, **OPProjectResources**, or **OPResource** object.

Remarks The **OPFilters** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPGlobalEdit Object

Description The **OPGlobalEdit** object contains properties for a global edit definition within the **OPGlobalEdits** collection.

Remarks The **OPGlobalEdit** object has the following properties and methods:

Properties		
ApplyToField	Filter	References
Expression	Name	TableName

Methods
Apply

OPGlobalEdits Collection

Description The **OPGlobalEdits** collection contains the list of all **OPGlobalEdits** objects belonging to the parent **OProject**, **OPResource**, or **OPProjectResources** object.

Remarks The **OPGlobalEdits** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPGraphs Collection

Description The **OPGraphs** collection contains the list of all histogram icons stored in the parent **OPProject** object. The members of this collection are **OPView** objects.

Remarks The **OPGraphs** collection object has the following properties and methods:

Properties		
Count		
Methods		
ActivateByFilename	AddView	Item

OPHolidays Collection

Description The **OPHolidays** collection contains a set of **OPDate** objects that represent the holidays that were defined for a specified **OPCalendarRecord** object.

Remarks The **OPHolidays** collection object has the following properties and methods:

Properties		
Count	Remove	
Methods		
Add	GetAll	Remove
AddAll	Item	

OPIcon Object

Description The **OPIcon** object represents a file icon in the Open Plan Explorer. The **OPIcon** object belongs to one of the following collections: **OPFCCalendars**, **OPFCCodes**, **OPFCResources**, or **OPFCProjects**.

Remarks The **OPIcon** object has the following methods:

Methods		
Activate	GetField	Name
FileName	GetFields	SetField

OPNetworks Collection

Description The **OPNetworks** collection contains the list of all network icons stored in the parent **OPProject** object. The members of this collection are **OPView** objects.

Remarks The **OPNetworks** collection object has the following properties and methods:

Properties

Count		
Methods		
ActivateByFilename	AddView	Item

OPNote Object

Description The **OPNote** object contains properties for a note within the **OPNotes** collection.

Remarks The **OPNote** object has the following properties:

Properties	
Category	ModifiedDate
ModifiedBy	NoteText

OPNotes Collection

Description The **OPNotes** collection contains the list of all **OPNote** objects belonging to the parent object. The following objects contain an **OPNotes** collection: **OPProject**, **OPActivity**, **OPCodeRecord, OPResourceRecord**, **OPActivityResources**, **OPProjectResources**.

Remarks The **OPNotes** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPPredecessor Object

Description The **OPPredecessor** object contains properties and methods for a predecessor record within the **OPPredecessors** collection.

Remarks The **OPredecessor** object has the following properties and methods:

Properties		
Calendar	Lag	SuccessorID
FreeFloat	RefreshData	TotalFloat
ID	RelationshipType	Turns
Methods		
GetCurrentFields	Getfields	SetCurrentFields
Getfield	Remove	Setfield

OPPredecessors Collection

Description The **OPPredecessors** collection contains the list of all **OPPredecessor** objects belonging to the parent **OPActivity** object.

Remarks The **OPPredecessors** collection object has the following properties and methods:

Properties		
Count	Sort	
Methods		
Add	Item	SetSortFields
AssignCurrentFieldSet	Remove	SetSortTo
Fields	SetCollectionGrowth	

OPProject Object

Description The **OPProject** object contains properties and methods for an open project within the **OPProjects** collection.

Remarks The **OPProject** object has the following properties and methods:

Properties		
AccessMode	DateFormat	MultipleEnd
AutoAnalyze	DefaultActivityCalendar	Name
AutoProgActBasedOn	DefaultActivityType	OutOfSeqOpt
AutoProgActComplete	DefaultAuxiliaryAccessMode	Priority1Name
AutoProgActFilter	DefaultDurationChar	Priority2Name
AutoProgActInProgress	DefaultDurationUnit	Priority3Name
AutoProgActivity	DefaultProjectAccessMode	Resource
AutoProgActProgressType	DefaultRelationshipCalendar	ScheduleFinishDate
AutoProgActSetPPC	Description	ScheduleMethod
AutoProgResEndDate	EarlyFinishDate	ScheduleTimeUnit
AutoProgResource	FileName	ScheduleTimeUnitMinutes
AutoProgResStartDate	FiscalCalendar	Smoothing
CalcCostActual	HardZeroes	StartDate
CalcCostBasedOn	Height	StartView
CalcCostBudget	InProgressPriority	StatusDate
CalcCostEarnedValues	LateFinishDate	TargetCost
CalcCostEscalated	Manager	TargetFinishDate
CalcCostIncludeChildValues	MinimumCalcDurationUnit	TargetFinishType
CalcCostRemaining	MinutesPerDay	TargetStartDate
Calendar	MinutesPerDefaultUnit	Width
Client	MinutesPerMinDurUnit	XPosition
Company	MinutesPerMonth	YPosition
CurrentActivity	MinutesPerWeek	

Methods		
Activities	Filters	RiskAnalyze
AutoProgress	GeneralExport	Rollups
AutoProgressEx	GeneralImport	Save
Barcharts	GenerateCrosstabDates	SaveAs
BaselinesList	GetCrosstabDates	SetCrosstabDates
BatchGlobalEdits	GetCrosstabDatesInXML	SetCrosstabDatesFromXML
CalculateCost	GetField	SetEarnedValueCrosstabOptions
CalculateCostEx	GetFields	SetField
CalculatedFields	GetSelectedActivitiesArray	SetResourceCrosstabOptions
Categories	GetSharedModeLock	SetSharedmodeLock
CloseView	GlobalEdits	Shut
CodeFiles	Graphs	ShutWOSave
Conceal	Maximize	Sorts
Costs	Minimize	Spreadsheets
CreateBackup	Networks	TimeAnalyze
CreateBaseline	Notes	UpdateBaseline
DeleteBaseline	Resources	UpdateBaselineEx
Disable	ResourceSchedule	Views
Enable	Restore	

OPProjectCode Collection

Description The **OPProjectCode** collection represents a code file that is associated with an open project and contains a list of **OPCodeRecord** objects.

Remarks The **OPProjectCode** collection object has the following properties and methods:

Properties		
Count	Height	YPosition
FieldName	Width	
Filename	XPosition	
Methods		
Add	Maximize	Save
Categories	Minimize	SaveAs
Conceal	Remove	SetCollectionGrowth
Disable	Restore	Shut
Enable	Restore	ShutWOSave
Item	Rollups	

OPProjectCodes Collection

Description The **OPProjectCodes** collection contains a list of all the code files that are associated with a specified **OPProject** object.

Remarks The **OPProjectCodes** collection object has the following properties and methods:

Properties
Count
Methods

Add	Item	Remove

OPProjectResource Object

Description The **OPProjectResource** object represents a resource assigned to one or more activities in the current project.

Remarks The **OPProjectResource** object has the following methods:

Properties		
Class	RollUp	UnitCost
Description	Suppress	Units
ID	Threshold	
RefreshData	Type	
Methods		
Availabilities	GetFields	Remove
GetCurrentFields	GetResourceCrosstabData	SetCurrentFields
GetEarnedValueCrosstabData	GetResourceCrosstabDataInXML	SetField
GetEarnedValueCrosstabDataInXML	GetResourceDateArray	
GetField	Notes	

OPProjectResources Collection

Description The **OPProjectResources** collection contains the list of all **OPProjectResource** objects in use by the parent **OPProject** object.

Remarks The **OPProjectResources** collection has the following properties and methods:

Properties		
Count	FileName	
Methods		
Add	Fields	Rollups
AssignCurrentFieldSet	Filters	Sorts
CalculatedFields	GlobalEdits	
Categories	Item	

OPProjects Collection

Description The **OPProjects** collection contains the list of all open project files.

Remarks The **OPProjects** collection object has the following properties and methods:

Properties	
Count	
Methods	
Fields	Item

OPReportingCalendar Collection

Description The **OPReportingCalendar** collection contains the list of dates and labels that make up an individual reporting calendar.

Remarks The **OPReportingCalendar** collection object has the following properties and methods:

Properties		
Count	FileName	
Methods		
Add	Item	Remove

OPReportingCalendarRecord Object

Description The **OPReportingCalendarRecord** object represents a single reporting calendar period.

Remarks The **OPProjects** collection object has the following properties and methods:

Properties	
Date	Label

OPReportingCalendars Collection

Description The **OPReportingCalendars** collection contains the list of all reporting calendars available to the user.

Remarks The **OPProjects** collection object has the following properties and methods:

Properties
Count
Methods
Item

OPResource Collection

Description The **OPResource** collection represents a open resource file and contains a list of **OPResourceRecord** objects.

Remarks The **OPResource** collection object has the following properties and methods:

Properties		
Calendar	Filter	Width
Count	Height	XPosition
Filename	Sort	YPosition
Methods		
Add	GetCrosstabdatesInXML	Save
AssignCurrentFieldSet	GetCrosstabMinutesPerDefaultUnit	SaveAs
CalculatedFields	GetField	SetCollectionGrowth
Categories	GetFields	SetCrosstabDates
Conceal	GlobalEdits	SetCrosstabDatesFromXML
Disable	Item	SetCrosstabMinutesPerDefaultUnit
Enable	Maximize	SetField
Fields	Minimize	SetResourceCrosstabOptions
Filters	Remove	Shut
GenerateCrosstabDates	Restore	ShutWOSave

GetCrosstabDates	Rollups	Sorts

OPResourceRecord Object

Description The **OPResourceRecord** object contains properties and methods for a resource record within an **OPResource** collection. It represents a record within a resource file.

Remarks The **OPResourceRecord** object has the following properties and methods:

Properties		
Class	RollUp	UnitCost
Description	Suppress	Units
ID	Threshold	
RefreshData	Type	
Methods		
AssignSkill	Getfield	RemoveSkill
Availabilities	Getfields	SetCurrentFields
GetAvailabilityCrosstabData	GetSkills	Setfield
GetAvailabilityCrosstabDataInXML	Notes	
GetCurrentFields	Remove	

OPResourceUsage Object

Description The **OPResourceUsage** object represents a resource usage assigned to a specific activity/assignment in the current project.

Remarks The **OPResourceUsage** object has the following properties and Methods:

Properties		
ActivityID	EscalatedCost	ResourceID
AlternateResourceID	FinishDate	StartDate
Cost	RefreshData	Used

Methods		
GetCurrentFields	Remove	
GetField	SetCurrentField	
GetFields	SetField	

OPResourceUsages Collection

Description The **OPResourceUsages** collection contains the list of all OPResourceUsage objects belonging to the parent **OPActivity** object.

Remarks The **OPResourceUsages** collection object has the following properties and methods:

Properties	
Count	
Methods	
Add	Item

OPResources Collection

Description The **OPResources** collection contains the list of all open resource files.

Remarks The **OPResources** collection object has the following properties and methods:

Properties		
Count		
Methods		
Fields	Item	

OPRollup Object

Description The **OPRollup** object contains properties for a rollup definition within the **OPRollups** collection.

Remarks The **OPRollup** object has the following properties and methods:

Properties		
Name		
Methods		
Apply	Remove	

OPRollups Collection

Description The **OPRollups** collection contains the list of all **OPRollups** belonging to the parent object.

Remarks The **OPRollups** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	

OPShift Object

Description The **OPShift** object contains properties for a shift record within the **OPShifts** collection.

Remarks The **OPShift** object has the following properties:

Properties	
StartTime	StopTime

OPShifts Collection

Description The **OPShifts** collection contains a set of **OPShift** objects.

Remarks The **OPShifts** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	GetAll	Remove
AddAll	Item	RemoveAll

OPSkill Object

Description The **OPSkill** object contains methods for a skill record within the **OPSkills** collection.

Remarks The **OPSkill** object has the following properties:

Methods		
Description	Name	UnitCost

OPSkills Collection

Description The **OPSkills** collection contains a set of **OPSkill** objects.

Remarks The **OPSkills** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPSort Object

Description The **OPSort** object contains properties for a sort within the **OPSorts** collection.

Remarks The **OPSort** object has the following properties:

Properties		
Expression	Name	References
TableName		

OPSorts Collection

Description The **OPSorts** collection contains the list of all **OPSort** objects belonging to the parent **OPProject**, **OPProjectResources**, or **OPResource** object.

Remarks The **OPSorts** collection object has the following properties and methods:

Properties		
Count		
Methods		
Add	Item	Remove

OPSpreadsheets Collection

Description The **OPSpreadsheets** collection contains the list of all spreadsheet icons stored in the parent **OPProject** object. The members of this collection are **OPView** objects.

Remarks The **OPSpreadsheets** collection object has the following properties and methods:

Properties		
Count		
Methods		
ActivateByFieldname	AddView	Item

OPStandardDay Object

Description The **OPStandardDay** object contains properties and methods for a standard day defined in a calendar record.

Remarks The **OPStandardDay** object has the following properties and methods:

Properties
Work
Methods
Shifts

OPView Object

Description The **OPView** object represents a view icon stored in a project.

Remarks The **OPView** object has the following methods:

Methods		
Activate	Name	SetPrintExpandColumns
CollapseAll	RefreshFilterandSort	SetViewLegend
Conceal	Restore	SetViewSpreadsheet
Description	ViewCategory	SetViewHorizontalPage
Disable	ViewClass	SetViewResourceData
Enable	Snapshot	SetPrintDateOption
ExpandAll	SetPrintMargins	SetPrintDateRange
Filename	SetPrintNotes	SetDateScaleOption
Maximize	SetPrintLineNumbers	SetNumberOfAxes
Minimize	SetPrintHeadingColorsOff	SetDateAutoScale

OPViews Collection

Description The **OPViews** collection contains the list of all view icons stored in a project.

Remarks The **OPViews** collection object has the following properties and methods:

Properties		
Count		
Methods		
ActivateByFilename	AddView	Item
ActivateByFileNameEx		

OPWebWindow Object

Description The **OPWebWindow** object represents an open browser view.

Remarks The **OPWebWindow** object has the following properties and methods:

Properties		
QueryString	Title	URLAddress
Methods		
Close	GetPosition	Restore
Conceal	Maximize	SetFocus
Disable	Minimize	SetPosition

Enable	Refresh

OPWebWindows Collection

Description The **OPWebWindows** collection contains the list of all open browser views.

Remarks The **OPWebWindows** collection object has the following properties and methods:

Properties	
Count	
Methods	
Add	Item

Properties

AccessMode Property

Applies To **OPProject** object

Description Defines the default access mode in which the project was opened.

Access Type Get

Data Type String

Syntax *string* = *project*.**AccessMode**

Remarks The **AccessMode** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A variable of type String

ActivityID Property

Applies To The following objects: **OPActivityResource**, **OPAssignment**, **OPCost**, and **OPResourceUsage**

Description The unique identifier for the activity associated with the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object*.**ActivityID**
object.**ActivityID** = *string*

Remarks The **ActivityID** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

When applied to the **OPActivityResource** or **OPAssignment** objects, the only allowed access type is **Get**.

ActivityLogicFlag Property

Applies To **OPActivity** object

Description Indicates if the activity represented by the specified object is a start or finish activity.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ActivityLogicFlag**

Remarks The **ActivityLogicFlag** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The **ActivityLogicFlag** property returns one of the following values: End Activity, None, Start Activity, or Start and End Activity.

ActualCost Property

Applies To **OPCost** object

Description The actual cost associated with the cost record represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *cost*.**ActualCost**
cost.**ActualCost =** *double*

Remarks The **ActualCost** property syntax uses these parts:

Part	Description
Cost	An object variable of type OPCost
double	A double-precision floating-point number or variable of type Double

ActualFinishDate Property

Applies To **OPActivity** object

Description Gets or sets the actual completion date for a specific **OPActivity** object.

Access Type Get or set

Data Type Date

Syntax *date*= *activity*.**ActualFinishDate**
activity.**ActualFinishDate =** *date*

Remarks The **ActualFinishDate** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
Date	A date or variable of type Date

ActualQty Property

Applies To **OPCost** object

Description The actual quantity associated with the cost record represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *cost*.**ActualQty**
cost.**ActualQty =** *double*

Remarks The **ActualQty** property syntax uses these parts:

Part	Description
Cost	An object variable of type OPCost
double	A double-precision floating-point number or variable of type Double

ActualStartDate Property

Applies To **OPActivity** object

Description Gets or sets the actual start date for a specific **OPActivity** object.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**ActualStartDate**
activity.**ActualStartDate =** *date*

Remarks The **ActualStartDate** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A date or variable of type Date

AlternateResourceID Property

Applies To The following objects: **OPAssignment** and **OPResourceUsage**.

Description The unique identifier of a resource that resource scheduling may use to fulfill the resource requirement represented by the specified object if the originally requested resource is not available.

Access Type Get or set

Data Type String

Syntax *string* = *assignment*.**AlternateResourceID**
assignment.**AlternateResourceID =** *string*

Remarks The **AlternateResourceID** property syntax uses these parts:

Part	Description
assignment	An object variable of type OPAssignment
string	A string or variable of type String

ApplyToField

Applies To **OPGlobalEdit** object

Description The name of the field to which the global edit represented by the parent OPGlobalEdit object will be applied.

Access Type Get or set

Data Type String

Syntax *string* = *globaledit.***ApplyToField**

*globaledit.***ApplyToField** = *string*

Remarks The **ApplyToField** property syntax uses these parts:

Part	Description
string	A string or variable of type String
globaledit	An object variable of type OPGlobalEdit

AutoAnalyze Property

Applies To **OPProject** object

Description Gets or sets the auto-time analysis option flag for the project represented by the specified object.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project.***AutoAnalyze**
*project.***AutoAnalyze =** *boolean*

Remarks The **AutoAnalyze** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

AutoProgActBasedOn Property

Applies To **OPProject** object

Description Specifies the type of dates Open Plan should use when performing automatic progress calculations when the Update Activity Status and Actual Dates option on the Automatic Progress is on. The setting of this property is the equivalent of setting the Based On option on the Automatic Progress dialog box.

Access Type Get or set

Data Type String

Syntax *string* = *project.***AutoProgActBasedOn**
project. **AutoProgActBasedOn =** *string*

Remarks The **AutoProgActBasedOn** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String equal to one of the following values: EARLY (the default), LATE, SCHEDULE, or BASELINE.

AutoProgActComplete Property

Applies To **OPProject** object

Description Controls whether Open Plan should treat as complete any activity whose Based On finish date is before Time Now. If the activity is to be treated as complete, the appropriate finish date is written to the Actual Finish Date field. The setting of this property is the equivalent of selecting the Complete If Finished Before Time Now option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**AutoProgActComplete**
project. **AutoProgActComplete =** *boolean*

Remarks The **AutoProgActComplete** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Complete If Finished Before Time Now option is On.

AutoProgActFilter Property

Applies To **OPProject** object

Description Defines a filter to be used to limit the scope of automatic progress calculations. The setting of this property is the equivalent of setting the Matching Filter option on the Automatic Progress dialog box.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**AutoProgActFilter**
project. **AutoProgActFilter =** *string*

Remarks The **AutoProgActFilter** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String containing the name of a valid filter.

AutoProgActInProgress Property

Applies To **OPProject** object

Description Controls whether Open Plan should treat as in progress any activity whose Based On start date is before Time Now. If the activity is to be treated as in progress, the appropriate start date is written to the Actual Start Date field. The setting of this property is the equivalent of selecting the In-Progress if Started Before Time Now option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**AutoProgActInProgress**
project. **AutoProgActInProgress =** *boolean*

Remarks The **AutoProgActInProgress** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the In-Progress if Started Before Time Now option is On.

AutoProgActivity Property

Applies To **OPProject** object

Description Controls whether Open Plan should update the status and actual dates for the activities in the project. The setting of this property is the equivalent of selecting the Update Activity Status and Actual Dates option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**AutoProgActivity**
project. **AutoProgActivity =** *boolean*

Remarks The **AutoProgActivity** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Update Activity Status and Actual Dates option is On.

AutoProgActProgressType Property

Applies To **OPProject** object

Description Specifies the type of progress Open Plan should calculate when performing automatic progress calculations when the In-Progress if Started Before Time Now option is on. The setting of this property is the equivalent of setting the Calculate option on the Automatic Progress dialog box.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**AutoProgActProgressType**
project.**AutoProgActProgressType =** *string*

Remarks The **AutoProgActProgressType** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
String	A string or variable of type String equal to one of the following values: REMAINING (the default), ELAPSED, PERCENT, or EXPECTED.

AutoProgActSetPPC Property

Applies To **OPProject** object

Description Controls whether Open Plan should update the Physical Percent Complete field for any activity changed by Automatic Progress calculations. The setting of this property is the equivalent of selecting the Update Physical Percent Complete option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**AutoProgActSetPPC**
project.**AutoProgActSetPPC =** *boolean*

Remarks The **AutoProgActSetPPC** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

A value of True (the default) indicates that the Update Physical Percent Complete option is On.

AutoProgResEndDate Property

Applies To **OPProject** object

Description Defines the end of the period covered by the automatic resource progress calculation. The setting of this property is the equivalent of setting the End Date option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**AutoProgResourceEndDate**
project.**AutoProgResEndDate =** *date*

Remarks The **AutoProgResEndDate** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A date value or variable of type Date

AutoProgResource Property

Applies To **OPProject** object

Description Controls whether Open Plan should update resource progress for any resource whose Progress Based on Activity Progress option on the Resource Details dialog box is set to True. The setting of this property is the equivalent of selecting the Update Resource Progress option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**AutoProgResource**
project.**AutoProgResource =** *boolean*

Remarks The **AutoProgResource** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject

boolean	A boolean value or variable of type Boolean

A value of True (the default) indicates that the Update Resource Progress option is On.

AutoProgResStartDate Property

Applies To **OPProject** object

Description Defines the start of the period covered by the automatic resource progress calculation. The setting of this property is the equivalent of setting the Start Date option on the Automatic Progress dialog box.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**AutoProgResourceStartDate**
project.**AutoProgResStartDate =** *date*

Remarks The **AutoProgResStartDate** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A date value or variable of type Date

BaselineFinish Property

Applies To **OPActivity** object

Description Gets or sets the original planned finish date from the currently selected baseline for a specific **OPActivity** object.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**BaselineFinish**
activity.**BaselineFinish =** *date*

Remarks The **BaselineFinish** property syntax uses these parts:

Part	Description
Activity	An object variable of type OPActivity
Date	A date value or variable of type Date

Note: The value in this field is overwritten each time a baseline is created or selected.

BaselineStart Property

Applies To **OPActivity** object

Description Gets or sets the original planned finish date from the currently selected baseline for a specific **OPActivity** object.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**BaselineStart**
activity.**BaselineStart =** *date*

Remarks The **BaselineStart** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A date value or variable of type Date

> **Note:** The value in this field is overwritten each time a baseline is created or selected.

BoxHeight Property

Applies To **OPActivity** object

Description Gets or sets the height of the activity box for the activity represented by the specified object in Network views.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *activity*.**BoxHeight**
activity.**BoxHeight** = *integer*

Remarks The **BoxHeight** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
integer	An integer or variable of type Integer

BoxWidth Property

Applies To **OPActivity** object

Description Gets or sets the width of the activity box for the activity represented by the specified object in Network views.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *activity*.**BoxWidth**
activity.**BoxWidth** = *integer*

Remarks The **BoxWidth** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
integer	An integer or variable of type Integer

BudgetCost Property

Applies To **OPActivity** object

Description Gets or sets the activity budget cost of the specified **OPActivity** object.

Access Type Get or set

Data Type Double

Syntax *double* = *activity*.**BudgetCost**
activity.**BudgetCost =** *double*

Remarks The **BudgetCost** property syntax uses these parts:

Part	Description
Activity	An object variable of type OPActivity
Double	A double-precision floating-point number or variable of type Double

> **Note:** This value does not include any resource budget cost.

CalcAcross Property

Applies To **OPCalculatedField** object

Description Specifies whether or not the value of the calculated field represented by the specified object is calculated across a grouping summary or summarized up.

Access Type Get or Set

Data Type Boolean

Syntax *boolean* = *calcfield*.**CalcAcross**
calcfield. **CalcAcross** = *boolean*

Remarks The **CalcAcross** property syntax uses these parts:

Part	Description
calcfield	An object variable of type OPCalculatedField
boolean	A boolean value or variable of type Boolean

CalcCostActual Property

Applies To **OPProject** object

Description Controls whether Open Plan should calculate the actual costs for the resources assigned to each progressed activity in the project. The setting of this property is the equivalent of selecting the Actual Cost option on the Cost Calculations dialog box.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostActual**
project.**CalcCostActual =** *boolean*

Remarks The **CalcCostActual** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

A value of True (the default) indicates that the Actual Cost option is On.

CalcCostBasedOn Property

Applies To **OPProject** object

Description Controls whether Open Plan should calculate the budgeted costs for the resources assigned to each activity in the project using the early dates, late dates or scheduled dates. The setting of this property is the equivalent of setting the Based On option on the Cost Calculations dialog box.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**CalcCostBasedOn**
project.**CalcCostBasedOn =** *string*

Remarks The **CalcCostBasedOn** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String equal to one of the following values: EARLY (the default), LATE, SCHEDULE, or BASELINE.

CalcCostBudget Property

Applies To **OPProject** object

Description Controls whether Open Plan should calculate the budgeted costs for the resources assigned to each activity in the project. The setting of this property is the equivalent of selecting the Budget Cost option on the Cost Calculations dialog box

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostBudget**
project.**CalcCostBudget =** *boolean*

Remarks The **CalcCostBudget** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Budget Cost option is On.

CalcCostEscalated Property

Applies To **OPProject** object

Description Controls whether Open Plan should use escalated or unescalated costs when calculating the budgeted costs for the resources assigned to each activity in the project. The setting of this property is the equivalent of selecting the Use Resource Cost Escalation option on the Cost Calculations dialog box

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostEscalated**
project.**CalcCostEscalated =** *boolean*

Remarks The **CalcCostEscalated** property syntax uses these parts:

Part	Description

Project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Use Resource Cost Escalation option is On.

CalcCostEarnedValue Property

Applies To **OPProject** object

Description Controls whether Open Plan should use earned value costs when calculating the budgeted costs for the resources assigned to each activity in the project. The setting of this property is the equivalent of selecting the Earned Values (BCWP and BCWS) option on the Cost Calculations dialog box

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostEarnedValue**
project.**CalcCostEarnedValue =** *boolean*

Remarks The **CalcCostEarnedValue** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Earned Values (BCWP and BCWS) option is On.

CalcCostRemaining Property

Applies To **OPProject** object

Description Controls whether Open Plan should use remaining costs when calculating the budgeted costs for the resources assigned to each activity in the project. The setting of this property is the equivalent of selecting the Remaining Cost option on the Cost Calculations dialog box

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostRemaining**
project.**CalcCostRemaining =** *boolean*

Remarks The **CalcCostRemaining** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the Remaining Cost option is On.

CalcCostIncludeChildValues Property

Applies To **OPProject** object

Description Controls whether Open Plan should include child activity values in subproject costs when calculating the budgeted costs for the resources assigned to each activity in the project.

The setting of this property is the equivalent of selecting the Include Child Activity Values in Subproject Costs option on the Cost Calculations dialog box

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**CalcCostIncludeChildValues**
project.**CalcCostIncludeChildValues =** *boolean*

Remarks The **CalcCostIncludeChildValues** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

The default value of True indicates that the the Include Child Activity Values in Subproject Costs option is On.

Calendar Property

Applies To **OPActivity**, **OPAvailability**, **OPPredecessor**, **OPProject**, and **OPResource** objects

Description For the **OPActivity**, **OPAvailability**, and **OPPredecessor** objects, the property gets or sets the name of the calendar record used by the object. For the **OPProject** and **OPResource** object, the property gets or sets the calendar file used by the object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Calendar**
object.**Calendar =** *string*

Remarks The **Calendar** property syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above
String	A string or variable of type String

For the **OPProject** object, the string recognized by this property will be the complete path name of a calendar file. For the **OPActivity**, **OPAvailability**, and **OPPredecessor** object, this string will be the name of a calendar record within a calendar file.

Category Property

Applies To **OPNote** object

Description Gets the name of the category to which the note represented by the specified object applies.

Access Type Get

Data Type String

Syntax *string* = *note*.**Category**

Remarks The **Category** property syntax uses these parts:

Part	Description
Note	An object variable of type OPNote
String	A variable of type String

Class Property

Applies To **OPActivityResource**, **OPProjectResource**, and **OPResourceRecord** objects

Description Gets or sets the classification of the resource record represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *resourcerecord*.**Class**
resourcerecord.**Class =** *string*

Remarks The **Class** property syntax uses these parts:

Part	Description
resourcerecord	An object variable of one of the types listed above
string	A string or variable of type String

The string values recognized by the **Class** property are: Labor, Material, Other Direct Costs, Subcontract

Client Property

Applies To **OPProject** object

Description Gets or sets the client for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Client**
project.**Client =** *string*

Remarks The **Client** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Code Property

Applies To **OPCodeRecord** object

Description The unique identifier for the code record represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *coderecord*.**Code**
coderecord.**Code =** *string*

Remarks The **Code** property syntax uses these parts:

Part	Description
coderecord	An object variable of type OPCodeRecord
string	A string or variable of type String

Company Property

Applies To **OPProject** object

Description Gets or sets the company for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Company**
project.**Company =** *string*

Remarks The **Company** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Complete Property

Applies To **OPStep** object

Description A value indicating whether a step is complete or not. This value is also calculated using the PercentComplete property. A value of **True** sets the **PercentComplete** value to **100**, and a value of **False** sets **PercentComplete** to **0**.

Access Type Get or Set

Data Type Boolean

Syntax *boolean* = *step*.**Complete**

Remarks The **Complete** property syntax uses these parts:

Part	Description
step	An object variable of type OPStep
boolean	A boolean or variable of type Boolean

ComputedRemainingDuration Property

Applies To **OPActivity** object

Description Gets the remaining duration computed by time analysis for a specific **OPActivity** object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ComputedRemainingDuration**

Remarks The **ComputedRemainingDuration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Cost Property

Applies To **OPResourceUsage** object

Description The cost value associated with the resource usage record represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *resourceusage*.**Cost**
resourceusage.**Cost** = *double*

Remarks The **Cost** property syntax uses these parts:

Part	Description
resourceusage	An object variable of type OPResourceUsage
double	A double-precision floating-point number or variable of type Double

ComputedStatus Property

Applies To **OPActivity** object

Description Gets an indicator of the activity status for a specific **OPActivity** object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ComputedStatus**

Remarks The **ComputedStatus** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Values returned by the **ComputedStatus** property are: Planned, In Progress, and Complete.

CostToDate Property

Applies To **OPActivity** object

Description The activity actual cost expended to date for the activity represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *activity*.**CostToDate**
activity.**CostToDate =** *double*

Remarks The **CostToDate** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
double	A double-precision floating-point number or variable of type Double

> **Note:** This value does not include the resource actual cost to date.

Count Property

Applies To All collection objects in the Open Plan object hierarchy

Description Counts the number of individual objects in the specified collection.

Access Type Get

Data Type Long

Syntax *long* = *collection*.**Count**

Remarks The **Count** property syntax uses these parts:

Part	Description
collection	An object variable of one of the types specified above
long	A variable of type Long

Critical Property

Applies To **OPActivity** object

Description An indicator of the criticality of the activity represented by the specified object.

Access Type Get

Data Type Long

Syntax *long* = *activity*.**Critical**

Remarks The **Critical** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
long	A variable of type Long

CriticalIndex Property

Applies To **OPActivity** object

Description The percentage of risk analysis simulation trials that resulted in the activity represented by the specified object being placed on the critical path.

Access Type Get

Data Type Long

Syntax *long* = *activity*.**CriticalIndex**

Remarks The **CriticalIndex** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
long	A variable of type Long

CurrentActivity Property

Applies To **OPProject** object

Description Returns the activity ID of the activity in the project represented by the specified object that is currently selected.

Access Type Get

Data Type String

Syntax *string* = *project*.**CurrentActivity**

Remarks The **CurrentActivity** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A variable of type String

Date Property

Applies To **OPDate** object, **OPReportingCalendarRecord** object

Description The date value of an **OPDate** or **OPReportingCalendarRecord** object.

Access Type Get

Data Type Date

Syntax *date* = *dateobject*.**Date**

Remarks The **Date** property syntax uses these parts:

Part	Description
dateobject	An object variable of type OPDate
date	A variable of type Date

DateFormat Property

Applies To **OPProject** object

Description Gets or sets the default date format for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**DateFormat**
project.**DateFormat =** *string*

Remarks The **DateFormat** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Note: Refer to the Date Formats table in this document for the parameters used to define date formats.

DBType Property

Applies To **OPField object**

Description The data type defined in WST_DCT for database fields, or the Open Plan data type for calculated and user-defined fields.

Access Type Get

Data Type String

Syntax *string* = *field*.**DBType**

Remarks The **DBType** property syntax uses these parts:

Part	Description
field	An object variable of type OPField

string	A variable of type String

Decimals Property

Applies To **OPCalculatedField** object

Description The number of decimal places in the result of the calculated field represented by the specified object.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *calcfield.***Decimals**
calcfield. **Decimals** = *integer*

Remarks The **Decimals** property syntax uses these parts:

Part	Description
calcfield	An object variable of type OPCalculatedField
integer	An integer or variable of type Integer

DefaultActivityCalendar Property

Applies To **OPProject** Object

Description Gets or sets the name of the default activity calendar for the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object.***DefaultActivityCalendar**
o*bject.***DefaultActivityCalendar** = *string*

Remarks The **DefaultActivityCalendar** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

DefaultActivityType Property

Applies To **OPProject** Object

Description Gets or sets the default activity type for the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object.***DefaultActivityType**
o*bject.***DefaultActivityType** = *string*

Remarks The **DefaultActivityType** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

DefaultAuxiliaryAccessMode Property

Applies To **OPProject** Object

Description Gets or sets the default access mode for the ancillary files belonging to the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object*.**DefaultAuxiliaryAccessMode**
o*bject*.**DefaultAuxiliaryAccessMode** = *string*

Remarks The **DefaultAuxiliaryAccessMode** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

DefaultDurationChar Property

Applies To **OPProject** object

Description Gets the default duration unit identifier for the project represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *project*.**DefaultDurationChar**

Remarks The **DefaultDurationChar** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A variable of type String

Values returned by the **DefaultDurationChar** property are: M (months), W (weeks), D (days), H (hours), and T (minutes).

DefaultDurationUnit Property

Applies To **OPProject** object

Description Gets or sets the default duration unit for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**DefaultDurationUnit**
project.**DefaultDurationUnit =** *string*

Remarks The **DefaultDurationUnit** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Valid values for **DefaultDurationUnit** are: Months, Weeks, Days, Hours, or Minutes. The property is case-sensitive.

DefaultProjectAccessMode Property

Applies To **OPProject** Object

Description Gets or sets the default access mode for the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object*.**DefaultProjectAccessMode**
o*bject*.**DefaultProjectAccessMode** = *string*

Remarks The **DefaultProjectAccessMode** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

DefaultRelationshipCalendar Property

Applies To **OPProject** Object

Description Gets or sets the name of the default relationship calendar for the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object*.**DefaultRelationshipCalendar**
o*bject*.**DefaultRelationshipCalendar** = *string*

Remarks The **DefaultRelationshipCalendar** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

DelayingResource Property

Applies To **OPActivity** object

Description The ID of the first resource to cause the activity represented by the specified object to be delayed during resource scheduling.

Access Type Get

Data Type String

Syntax *string* = *activity*.**DelayingResource**

Remarks The **DelayingResource** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Description Property

Applies To The following objects: **OPActivity**, **OPBaselines**, **OPCodeRecord**, **OPProject**, **OPActivityResource**, **OPProjectResource**, **OPResourceRecord**, **OPSkill**, and **OPStep**.

Description The description of the object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Description**
object.**Description =** *string*

Remarks The **Description** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

Duration Property

Applies To **OPActivity** object

Description The original duration of the activity.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**Duration**
activity.**Duration =** *string*

Remarks The **Duration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The string set to the **Duration** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required.

DurationDistShape Property

Applies To **OPActivity** object

Description The probability distribution shape used by risk analysis when interpreting the value of the Original Duration field for the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**DurationDistShape**
activity. **DurationDistShape =** *string*

Remarks The **DurationDistShape** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Valid values for **DurationDistShape** are: Beta, None, Normal, Triangular, or Uniform.

EarliestFeasible Property

Applies To **OPActivity** object

Description The earliest date that resource scheduling could attempt to schedule the activity represented by the specified object.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**EarliestFeasible**

Remarks The **EarliestFeasible** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
Date	A variable of type Date

EarlyFinish Property

Applies To **OPActivity** object

Description The early finish date for the activity represented by the specified object.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**EarlyFinish**

Remarks The **EarlyFinish** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

EarlyFinishDate Property

Applies To **OPProject** object

Description The early finish date for the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**EarlyFinishDate**
project.**EarlyFinishDate =** *date*

Remarks The **EarlyFinishDate** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

EarlyFinishStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the early finish date of the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**EarlyFinishStdDev**

Remarks The **EarlyFinishStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **EarlyFinishStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

EarlyStart Property

Applies To **OPActivity** object

Description The early start date for the activity represented by the specified object.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**EarlyStart**

Remarks The **EarlyStart** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

EarlyStartStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the early start date of the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**EarlyStartStdDev**

Remarks The **EarlyStartStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **EarlyStartStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

EndDate Property

Applies To **OPCost** object

Description The end date for the cost record represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *cost*.**EndDate**
cost.**EndDate=** *date*

Remarks The **EndDate** property syntax uses these parts:

Part	Description
cost	An object variable of type OPCost
date	A date or variable of type Date

Escalated Cost Property

Applies To **OPResourceUsage** object

Description The escalated cost value associated with the resource usage record represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *resourceusage*.**EscalatedCost**
resourceusage.**EscalatedCost =** *double*

Remarks The **Escalated Cost** property syntax uses these parts:

Part	Description
resourceusage	An object variable of type OPResourceUsage
double	A double-precision floating-point number or variable of type Double

ExpectedFinish Property

Applies To **OPActivity** object

Description Gets or sets the expected completion date for a specific **OPActivity** object.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**ExpectedFinish**
activity.**ExpectedFinish =** *date*

Remarks The **ExpectedFinish** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A date or variable of type Date

Expression Property

Applies To **OPCalculatedField, OPFilter, OPGlobalEdit**, and **OPSort** objects

Description Gets or sets the expression of the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Expression**
object. **Expression =** *string*

Remarks The **Expression** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

Fieldname Property

Applies To **OPProjectCode** collection, **OPField** object

Description For the **OPProjectCode** collection, refers to the Activity table field name associated with this code file. For the **OPField** object, refers to the name of the Open Plan field represented by the object.

Access Type Get

Data Type String

Syntax *string* = *object.***Fieldname**

Remarks The **Fieldname** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
string	A variable of type String

FieldType Property

Applies To **OPField** object

Description Refers to the Open Plan data type of the field represented by the OPField object.

Access Type Get

Data Type String

Syntax *string* = *object.***FieldType**

Remarks The **FieldType** property syntax uses these parts:

Part	Description
object	An object variable of type OPField
string	A variable of type String

Filename Property

Applies To The **OPProject**object and the following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, **OPProjectResources**, **OPResource**, and **OPReportingCalendar**

Description Returns the name of the file represented by the collection or object.

Access Type Get

Data Type String

Syntax *string* = *object*.**Filename**

Remarks The **Filename** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type String

Filter Property

Applies To **OPGlobalEdit** object and the following collections: **OPActivities**, **OPBaselines**, **OPResource**.

Description For the **OPGlobalEdit** object, gets or sets the name of the filter that is part of the global edit's definition. For the collections listed, gets or sets the name of the filter that is applied to the collection.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Filter**
object.**Filter** = *string*

Remarks The **Filter** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

When using the **Filter** property to apply a filter to a collection, make sure that the filter is applicable to the specified collection. The collection will ignore the setting if an invalid filter is specified. To stop applying any filter to a collection, set this property to an empty string ("").

FinishDate Property

Applies To The following objects: **OPAvailability**, **OPCost**, **OPProject**, and **OPResourceUsage**.

Description For the **OPAvailability** object, this is the date that the resource becomes available. For the **OPCost** object, it is the finish date for the cost record represented by the specified object. For the **OPProject** object, it is the finish date for the project represented by the specified object. For the **OPResourceUsage** object, it is the finish date for the resource usage represented by the specified activity/assignment in the current project.

Access Type Get or set

Data Type Date

Syntax *date* = *object*.**FinishDate**
object.**FinishDate**= *date*

Remarks The **FinishDate** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
date	A date or variable of type Date

FinishFreeFloat Property

Applies To **OPActivity** object

Description The amount of time the finish of the activity represented by the specified object can be delayed without impacting the early dates of any successor.

Access Type Get

Data Type String

Syntax *string* = *activity*.**FinishFreeFloat**

Remarks The **FinishFreeFloat** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **FinishFreeFloat** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

FinishTotalFloat Property

Applies To **OPActivity** object

Description The amount of time the finish of the activity represented by the specified object can be delayed without delaying the project finish or any target finish dates.

Access Type Get

Data Type String

Syntax *string* = *activity*.**FinishTotalFloat**

Remarks The **FinishTotalFloat** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **FinishTotalFloat** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

FirstUsage Property

Applies To **OPActivity** object

Description The first date any resource assigned to the activity represented by the specified object was used during resource scheduling.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**FirstUsage**

Remarks The **FirstUsage** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

FiscalCalendar Property

Applies To **OPProject** Object

Description Gets or sets the name of the fiscal (reporting) calendar for the project represented by the specified object.

Access Type Get or Set

Data Type String

Syntax *string* = *object*.**FiscalCalendar**
object.**FiscalCalendar** = *string*

Remarks The **FiscalCalendar** property syntax uses these parts:

Part	Description
object	An object variable of type OPProject
string	A string or variable of type String

FreeFloat Property

Applies To **OPActivity** object, **OPPredecessor** object

Description The amount of time the activity represented by the specified object can be delayed without impacting the early dates of any successor.

Access Type Get

Data Type String

Syntax *string* = *object*.**FreeFloat**

Remarks The **FreeFloat** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type String

The string returned by the **FreeFloat** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

FreeFloatStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the free float of the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**FreeFloatStdDev**

Remarks The **FreeFloatStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **FreeFloatStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

HardZeroes Property

Applies To **OPProject** object

Description The current setting of the resource scheduling Hard Zeroes option for the project represented by the specified object.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**HardZeroes**
project.**HardZeroes =** *boolean*

Remarks The **HardZeroes** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

A value of True indicates that the Hard Zeroes option is on.

Height Property

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, and **OPProject**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Gets or sets the height of the object's window in pixels.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *object*.**Height**
object.**Height =** *integer*

Remarks The **Height** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
integer	An integer or variable of type Integer

ID Property

Applies To **OPActivity**, **OPActivityResource**, **OPPredecessor**, **OPProjectResource**, and **OPResourceRecord** objects

Description For the **OPActivity** and **OPResourceRecord** objects, the **ID** property refers to the unique identifier of the object. For the **OPPredecessor** object, the **ID** property refers to the activity ID of the predecessor activity. For the **OPActivityResource** and **OPProjectResource** objects, refers to the resource ID of the object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**ID**
object.**ID =** *string*

Remarks The **ID** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

For the **OPPredecessor** object, the only access type allowed is **Get.**

InProgressPriority Property

Applies To **OPProject** object

Description The current setting of the resource scheduling option for in-progress priority for the project represented by the specified object.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**InProgressPriority**
project.**InProgressPriority =** *boolean*

Remarks The **InProgressPriority** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

A value of True indicates that the In-Progress priority option is on.

IsEditable Property

Applies To **OPField** object

Description Indicates whether the data in the field represented by the specified object is editable.

Access Type Get

Data Type Boolean

Syntax *boolean* = *field*.**IsEditable**

Remarks The **IsEditable** property syntax uses these parts:

Part	Description
field	An object variable of type OPField
boolean	A variable of type Boolean

KeyActivityStatus Property

Applies To **OPActivity** object

Description Indicates whether the activity represented by the specified object is designated as a key activity for the purpose of risk analysis.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *activity*.**KeyActivityStatus**
activity. **KeyActivityStatus =** *boolean*

Remarks The **KeyActivityStatus** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
boolean	A boolean value or variable of type Boolean

Label Property

Applies To **OPReportingCalendarRecord** object

Description Refers to the label assigned to the reporting calendar period represented by the specified object.

Access Type Get or Set

Data Type

Syntax *string* = *reportingcalendarrecord*.**Label**
reportingcalendarrecord.**Label =** *string*

Remarks The **Label** property syntax uses these parts:

Part	Description
reportingcalendarrecord	An object variable of type OPReportingCalendarRecord
string	A string or variable of type String

Lag Property

Applies To **OPPredecessor** object

Description The duration of the lag of the relationship represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *predecessor*.**Lag**
predecessor.**Lag =** *string*

Remarks The **Lag** property syntax uses these parts:

Part	Description
predecessor	An object variable of type OPPredecessor
string	A string or variable of type String

The string set to the **Lag** property must be in a format recognized by Open Plan as a valid duration or percentage, consisting of a number and a duration unit identifier or percent symbol. If the duration is zero, no duration unit identifier or percent symbol is required.

LateFinish Property

Applies To **OPActivity** object

Description The latest date the activity represented by the specified object could finish based on time analysis calculations.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**LateFinish**

Remarks The **LateFinish** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

LateFinishDate Property

Applies To **OPProject** object

Description The late finish date for the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**LateFinishDate**
project.**LateFinishDate** = *date*

Remarks The **LateFinishDate** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
date	A variable of type Date

LateFinishStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the late finish date of the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**LateFinishStdDev**

Remarks The **LateFinishStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **LateFinishStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

LateStart Property

Applies To **OPActivity** object

Description The latest date the activity represented by the specified object could start based on time analysis calculations.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**LateStart**

Remarks The **LateStart** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

LateStartStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the late start date of the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*. **LateStartStdDev**

Remarks The **LateStartStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **LateStartStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

Length Property

Applies To **OPField** object

Description The length of the Open Plan field represented by the specified object. For database fields, returns the length as defined in WST_DCT, otherwise returns the maximum length of the data that can be stored in the field.

Access Type Get

Data Type Integer

Syntax *integer* = *field*.**Length**

Remarks The **Length** property syntax uses these parts:

Part	Description
field	An object variable of type OPField
integer	A variable of type Integer

Level Property

Applies To **OPAssignment** object, **OPAvailability** object

Description For the **OPAssignment** object, the amount of resource required for the assignment represented by the specified object. For the **OPAvailability** object, the amount of resource available during the availability period is represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *object*.**Level**
object.**Level =** *double*

Remarks The **Level** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
double	A double-precision floating-point number or variable of type Double

LevelType Property

Applies To **OPAssignment** object

Description A flag indicating if the resource requirement represented by the specified object is a total or a rate-per-time unit.

Access Type Get or set

Data Type String

Syntax *string* = *assignment*.**LevelType**
assignment.**LevelType =** *string*

Remarks The **LevelType** property syntax uses these parts:

Part	Description
assignment	An object variable of type OPAssignment
string	A string or variable of type String

The string values recognized by the **LevelType** property are: B (back load), D (double peak), E (early peak), F (front load), L (late peak), N (normal), T (linear), blank, and any spread curves added by the user.

Manager Property

Applies To **OPProject** object

Description The manager name stored in the **Project Properties** dialog box of the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Manager**
project.**Manager =** *string*

Remarks The **Manager** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

MaxDuration Property

Applies To **OPActivity** object

Description The maximum duration the activity represented by the specified object can take during splitting or re-profiling.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**MaxDuration**
activity.**MaxDuration =** *string*

Remarks The **MaxDuration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The string set to the **MaxDuration** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required.

MaxNoSplits Property

Applies To **OPActivity** object

Description The maximum number of segments into which the activity represented by the specified object can be split during resource scheduling.

Access Type Get or set

Data Type Integer

Syntax *integer* = *activity*.**MaxNoSplits**
activity.**MaxNoSplits =** *integer*

Remarks The **MaxNoSplits** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
integer	A integer or variable of type Integer

MinimumCalcDurationUnit Property

Applies To **OPProject** object

Description The smallest time unit used for calculating durations in the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**MinimumCalcDurationUnit**
project.**MinimumCalcDurationUnit =** *string*

Remarks The **MinimumCalcDurationUnit** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Allowed values are Minutes, Hours, Days, Weeks, and Months. The property is case-insensitive.

MinSplitLength Property

Applies To **OPActivity** object

Description The minimum duration of any segment of the activity represented by the specified object during resource scheduling, when Splittable is chosen for **ResourceScheduleType**.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**MinSplitLength**
activity.**MinSplitLength =** *string*

Remarks The **MinSplitLength** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The string set to the **MinSplitLength** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required.

MinutesPerDay Property

Applies To **OPProject** object

Description The duration conversion factor for days to hours for the project represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *integer* = *project*.**MinutesPerDay**
project.**MinutesPerDay** = *integer*

Remarks The **MinutesPerDay** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
integer	An integer or variable of type Integer

MinutesPerDefaultUnit Property

Applies To **OPProject** object

Description The number of minutes in the default duration unit for the project represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *project*.**MinutesPerDefaultUnit**
project.**MinutesPerDefaultUnit** = *double*

Remarks The **MinutesPerDefaultUnit** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
double	A double-precision floating-point number or variable of type Double

MinutesPerMinDurUnit Property

Applies To **OPProject** object

Description The number of minutes in the minimum duration unit in the project represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *project*.**MinutesPerMinDurUnit**
project.**MinutesPerMinDurUnit** = *double*

Remarks The **MinutesPerMinDurUnit** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
double	A double-precision floating-point number or variable of type Double

MinutesPerMonth Property

Applies To **OPProject** object

Description The duration conversion factor for months to hours for the project represented by the specified object.

Access Type Get or set

Data Type Long

Syntax *long* = *project*.**MinutesPerMonth**
project.**MinutesPerMonth** = *Long*

Remarks The **MinutesPerMonth** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
long	A long integer or variable of type Long

MinutesPerWeek Property

Applies To **OPProject** object

Description The duration conversion factor for weeks to hours for the project represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *integer* = *project*.**MinutesPerWeek**
project.**MinutesPerWeek** = *integer*

Remarks The **MinutesPerWeek** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
integer	An integer or variable of type Integer

ModifiedBy Property

Applies To **OPNote** object

Description Gets the name of the last user to modify the note represented by the specified object. This property returns an empty value.

Access Type Get

Data Type String

Syntax *string* = *note*.**ModifiedBy**
note. **ModifiedBy** = *string*

Remarks The **ModifiedBy** property syntax uses these parts:

Part	Description
note	An object variable of type OPNote
string	A string or variable of type String

ModifiedDate Property

Applies To **OPNote** object

Description Gets the date the note represented by the specified object was last modified. This property returns an empty value.

Access Type Get

Data Type Date

Syntax *date* = *note*.**ModifiedDate**
note. **ModifiedDate** = *date*

Remarks The **ModifiedDate** property syntax uses these parts

Part	Description
note	An object variable of type OPNote
date	A date or variable of type Date

MultipleEnd Property

Applies To **OPProject** object

Description The current setting for the time analysis multiple-ends option for the project represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *boolean* = *project*.**MultipleEnd**
project.**MultipleEnd** = *boolean*

Remarks The **MultipleEnd** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A boolean value or variable of type Boolean

Name Property

Applies To The following objects: **OPBaselines**, **OPBatchGlobalEdit, OPCalculatedField**, **OPCategory**, **OPFilter**, **OPGlobalEdit**, **OPProject**, **OPRollup**, **OPSkill**, and **OPSort**.

Description For the **OPBaselines**, **OPCalculatedField**, **OPCategory**, **OPFilter**, **OPGlobalEdit**, **OPRollup**, **OPSkill**, or **OPSort** objects, the **Name** property refers to the name assigned to the object when the object was saved in Open Plan. For the **OPProject** the **Name** property refers to the name of the file associated with the collection or object.

Access Type Get

Data Type String

Syntax *string* = *object*.**Name**

Remarks The **Name** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type String

NoteText Property

Applies To **OPNote** object

Description Gets or sets the text stored in the note represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *note*.**NoteText**
note. **NoteText** = *string*

Remarks The **NoteText** property syntax uses these parts:

Part	Description
note	An object variable of type OPNote
string	A string or variable of type String

OptimisticDuration Property

Applies To **OPActivity** object

Description The minimum estimated duration for the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**OptimisticDuration**
activity. **OptimisticDuration =** *string*

Remarks The **OptimisticDuration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The string set to the **OptimisticDuration** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required.

OutofSeqOpt Property

Applies To **OPProject** object

Description The current setting for the time analysis out-of-sequence option for the project represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *integer* = *project*.**OutOfSeqOpt**
project.**OutOfSeqOpt =** *integer*

Remarks The **OutOfSeqOpt** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
integer	An integer or variable of type Integer

The following values are recognized by the **OutOfSeqOpt** property: 0 (Ignore positive lag of predecessor), 1 (Observe positive lag of predecessor), or 2 (Ignore predecessor relationship).

PercentComplete Property

Applies To The following objects: **OPActivity** and **OPStep**.

Description The physical percent complete of an object.

Access Type Get or set

Data Type Double

Syntax *double* = *object*.**PercentComplete**
object.**PercentComplete** = *double*

Remarks The **PercentComplete** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
double	A double-precision floating-point number or variable of type Double

PessimisticDuration Property

Applies To **OPActivity** object

Description The maximum estimated duration for the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**PessimisticDuration**
activity.**PessimisticDuration =** *string*

Remarks The **PessimisticDuration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

The string set to the **PessimisticDuration** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required.

Priority1Name Property

Applies To **OPProject** object

Description The name of the first field used as a tiebreaker for resource scheduling for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Priority1Name**
project.**Priority1Name** = *string*

Remarks The **Priority1Name** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Priority2Name Property

Applies To **OPProject** object

Description The name of the second field used as a tie-breaker for resource scheduling for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Priority2Name**
project.**Priority2Name** = *string*

Remarks The **Priority2Name** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

Priority3Name Property

Applies To **OPProject** object

Description The name of the third field used as a tiebreaker for resource scheduling for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**Priority3Name**
project.**Priority3Name** = *string*

Remarks The **Priority3Name** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject

string	A string or variable of type String

ProgressFlag Property

Applies To **OPActivity** object

Description An indicator of the status of the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**ProgressFlag**
activity.**ProgressFlag** = *string*

Remarks The **ProgressFlag** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Valid values for this property are: Planned, Remaining Duration, Percent Complete, Elapsed Duration, and Complete.

ProgressValue Property

Applies To **OPActivity** object

Description The progress value of the activity, interpreted by Open Plan according to the **ProgressFlag** property.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**ProgressValue**
activity.**ProgressValue** = *string*

Remarks The **ProgressValue** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

If the **ProgressFlag** property is equal to Remaining Duration or Elapsed Duration, the **ProgressValue** will be a numeric value and a single trailing character to denote the unit of the duration. If the **ProgressFlag** property is equal to Planned, the **ProgressValue** property will be 0. If the **ProgressFlag** property is equal to Complete, the **ProgressValue** property will be 100%. If the **ProgressFlag** property is equal to Percent Complete, the **ProgressValue** property will be a numeric value followed by %. Zero values do not require the trailing character.

QueryString Property

Applies To **OPWebWindow** object

Description Returns the HTTP query string for the WebWindow represented by the specified object.

Access Type **Get**

Data Type String

Syntax *string* = *webwindow*.**QueryString**

Remarks The **QueryString** property syntax uses these parts:

Part	Description
webwindow	An object variable of type OPWebWindow
string	A variable of type String

References Property

Applies to The following objects: **OPCalculatedField**, **OPFilter**, **OPSort**, **OPGlobalEdit**

Description Gets information about the selected object including any references to it from other sources.

Access Type Get

Data Type XML string

Syntax string = my**CalculatedField.References**

Remarks The References property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
string	A variable of type string

RefreshData Property

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAssignment**, **OPAvailability**, **OPCodeRecord**, **OPCost**, **OPPredecessor**, **OPProjectResource**, **OPResourceRecord**, and **OPResourceUsage**.

Description Gets or sets a flag indicating if subsequent retrieval of properties results in data being refreshed from the database.

Access Type Get or Set

Data Type Boolean

Syntax *boolean* = *object*.**RefreshData**
object.**RefreshData** = *boolean*

Remarks The **RefreshData** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
boolean	A boolean value or variable of type Boolean

The **RefreshData** property should be set to **True** when working in shared mode.

RelationshipType Property

Applies To **OPPredecessor** object

Description The type of relationship between the predecessor activity and the successor activity for the relationship represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *predecessor*.**RelationshipType**

Remarks The **RelationshipType** property syntax uses these parts:

Part	Description
predecessor	An object variable of type OPPredecessor
string	A variable of type String

The **RelationshipType** property will return one of the following values: Finish to Finish, Finish to Start, Start to Finish, or Start to Start.

Remaining Property

Applies To **OPAssignment** object

Description The remaining amount of the assignment represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *assignment*.**Remaining**
assignment.**Remaining** = *double*

Remarks The **Remaining** property syntax uses these parts:

Part	Description
assignment	An object variable of type OPAssignment
double	A double-precision floating-point number or variable of type Double

Resource Property

Applies To **OPAvailability** object, **OPProject** object

Description For the **OPAvailability** object, the **Resource** property refers to the unique identifier of the resource associated with the availability record represented by the specified object. For the **OPProject** object, the **Resource** property refers to the resource file used by the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Resource**
object.**Resource** = *string*

Remarks The **Resource** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

When applied to the **OPAvailability** object, the only access type allowed is **Get**.

ResourceID Property

Applies To The following objects: **OPAssignment**, **OPCost**, and **OPResourceUsage**

Description For the **OPAssignment** object, it is the unique identifier of the resource being assigned. For the **OPCost** object, it is the unique identifier of the resource associated with the cost record represented by the specified object. For the **OPResourceUsage** object, it is the

unique identifier of the resource associated with a specific activity/assignment in the current project.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**ResourceID**
object.**ResourceID** = *string*

Remarks The **ResourceID** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

ResourceOffset Property

Applies To **OPAssignment** object

Description The amount of time between the start of the activity and the start of the assignment represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *assignment*.**ResourceOffset**
assignment.**ResourceOffset** = *string*

Remarks The **ResourceOffset** property syntax uses these parts:

Part	Description
assignment	An object variable of type OPAssignment
string	A string or variable of type String

The string set to the **ResourceOffset** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is required. Note that the sum of **ResourceOffset** and **ResourcePeriod** should not be greater than the activity duration.

ResourcePeriod Property

Applies To **OPAssignment** object

Description The amount of time for which a resource is required for this assignment.

Access Type Get or set

Data Type String

Syntax *string* = *assignment*.**ResourcePeriod**
assignment.**ResourcePeriod =** *string*

Remarks The **ResourcePeriod** property syntax uses these parts:

Part	Description
assignment	An object variable of type OPAssignment
string	A string or variable of type String

The string set to the **ResourcePeriod** property must be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the

duration is zero, no duration unit identifier is required. Note that the sum of **ResourceOffset** and **ResourcePeriod** should not be greater than the activity duration.

ResourceScheduleType Property

Applies To **OPActivity** object

Description Defines the resource scheduling type attribute of the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**ResourceScheduleType**
activity.**ResourceScheduleType =** *string*

Remarks The **ResourceScheduleType** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

Valid values for **ResourceScheduleType** are as follows: Normal, Splittable, Stretchable, Reprofilable, and Immediate. These values are not case-sensitive.

ResultType Property

Applies To **OPCalculatedField** object

Description Gets or sets the type of result returned by the calculated field represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *calcfield*.**ResultType**
calcfield. **ResultType** = *string*

Remarks The **ResultType** property syntax uses these parts:

Part	Description
calcfield	An object variable of type OPCalculatedField
string	A string or variable of type String

Valid values for this property are: Character, Date, Decimal, Duration, Finish Date, Integer, and Logical.

RollUp Property

Applies To **OPActivityResource** object, **OPProjectResource** object, **OPResourceRecord** object

Description Gets or sets the roll-up information for the resource record represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *resourcerecord*.**RollUp**
resourcerecord.**RollUp** = *string*

Remarks The **RollUp** property syntax uses these parts:

Part	Description
resourcerecord	An object variable of type of one of the types listed above
string	A string or variable of type String

Scale Property

Applies To **OPField** object

Description For numeric data types, returns the number of decimals places for the field represented by the specified **OPField** object. For all other data types, returns 0.

Access Type Get

Data Type Integer

Syntax *integer* = *field*.**Scale**

Remarks The **Scale** property syntax uses these parts:

Part	Description
field	An object variable of type OPField
integer	A variable of type Integer

ScheduleActions Property

Applies To **OPActivity** object

Description Indicates if resource scheduling actually split, stretched, or re-profiled the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ScheduleActions**

Remarks The **ScheduleActions** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
String	A variable of type String

ScheduledFinish Property

Applies To **OPActivity** object

Description The finish date of the activity represented by the specified object, taking into account any resource constraints.

Access Type Get

Data Type Date

Syntax *date* = *activity*.**ScheduledFinish**

Remarks The **ScheduledFinish** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

ScheduledStart Property

Applies To **OPActivity** object

Description The start date of the activity represented by the specified object, taking into account any resource constraints.

Access Type Get

Data Type Date

Syntax *date*= *activity*.**ScheduledStart**

Remarks The **ScheduledStart** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A variable of type Date

ScheduleDuration Property

Applies To **OPActivity** object

Description The duration of the activity represented by the specified object, taking into account any splitting, stretching, or re-profiling.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ScheduleDuration**

Remarks The **ScheduleDuration** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **ScheduleDuration** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

ScheduleFinishDate Property

Applies To **OPProject** object

Description The finish date calculated by resource scheduling for the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**ScheduleFinishDate**
project.**ScheduleFinishDate** = *date*

Remarks The **ScheduleFinishDate** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
date	A date or variable of type Date

ScheduleFloat Property

Applies To **OPActivity** object

Description The difference between the scheduled dates and the late dates for the activity represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *activity*.**ScheduleFloat**

Remarks The **ScheduleFloat** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A variable of type String

The string returned by the **ScheduleFloat** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

ScheduleMethod Property

Applies To **OPProject** object

Description The currently selected resource scheduling method (Time Limited or Resource Limited) for the project represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *integer* = *project*.**ScheduleMethod**
project.**ScheduleMethod** = *integer*

Remarks The **ScheduleMethod** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
integer	An integer or variable of type Integer

Time Limited resource scheduling is represented by the integer value 0; Resource Limited resource scheduling is represented by the integer value 1.

ScheduleTimeUnit Property

Applies To **OPProject** object

Description The resource scheduling time unit interval for the project represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**ScheduleTimeUnit**
project.**ScheduleTimeUnit** = *string*

Remarks The **ScheduleTimeUnit** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject

string	A string or variable of type String

Valid values for **ScheduleTimeUnit** are: M (months), W (weeks) , D (days), 12H (12 hours), 8H (8 hours), 6H (6 hours), 4H (4 hours), 2H (2 hours), or H (1 hour). The property is case-sensitive.

ScheduleTimeUnitMinutes Property

Applies To **OPProject** object

Description The number of minutes in the resource scheduling time unit interval for the project represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *project*.**ScheduleTimeUnitMinutes**
project.**ScheduleTimeUnitMinutes** = *double*

Remarks The **ScheduleTimeUnitMinutes** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
double	A double-precision floating-point number or variable of type Double

Selected Property

Applies To **OPBaselines** object

Description A flag indicating if the baseline represented by the **OPBaselines** object is currently selected.

Access Type Get

Data Type Boolean

Syntax *boolean* = *baseline*.**Selected**

Remarks The **Selected** property syntax uses these parts:

Part	Description
baseline	An object variable type OPBaselines
boolean	A variable of type Boolean

SilentMode Property

Applies To **OPCreateApplication33** object

Description When set equal to 1, the SilentMode property reduces the number of informational messages displayed to the user. The property remains on until it is set to 0 or Open Plan is restarted.

Access Type Get or Set

Data Type Boolean

Syntax *boolean* = *application*.**SilentMode**
application.**SilentMode** = *boolean*

Remarks The **SilentMode** property syntax uses these parts:

Part	Description

application	An object variable of type OPCreateApplication33
boolean	A variable of type Boolean

Smoothing Property

Applies To **OPProject** object

Description The current setting of the resource scheduling smoothing option for the activity represented by the specified object.

Access Type Get or set

Data Type Boolean

Syntax *boolean* = *project*.**Smoothing**
project.**Smoothing =** *boolean*

Remarks The **Smoothing** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
boolean	A variable of type Boolean

Sort Property

Applies To **OPActivities**, **OPBaselines**, **OPPredecessors**, and **OPResource** collections

Description Gets or sets the name of the sort that is being applied to the collection.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Sort**
object.**Sort** = *string*

Remarks The **Sort** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

Make sure that the sort is applicable to the specified collection. The collection will ignore the setting if an invalid sort is specified. To stop applying any sort from the collection, set this property to an empty string (“”).

StartDate Property

Applies To The following objects: **OPAvailability**,**OPCost**, **OPProject**, and **OPResourceUsage**.

Description For the **OPAvailability** object, this is the date that the resource becomes available. For the **OPCost** object, it is the start date for the cost record represented by the specified object. For the **OPProject** object, it is the start date for the project represented by the specified object. For the **OPResourceUsage** object, it is the start date for the resource usage represented by the specified activity/assignment in the current project.

Access Type Get or set

Data Type Date

Syntax *date* = *object*.**StartDate**
object.**StartDate**= *date*

Remarks The **StartDate** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
date	A date or variable of type Date

StartTime Property

Applies To **OPShift** object

Description The starting time of the shift represented by the specified object, measured in number of minutes since midnight.

Access Type Get or set

Data Type Integer

Syntax *integer* = *shift*.**StartTime**
shift.**StartTime** = *integer*

Remarks The **StartTime** property syntax uses these parts:

Part	Description
project	An object variable of type OPShift
integer	An integer or variable of type Integer

StartView Property

Applies To **OPProject** object

Description The name of the view to be opened when the project represented by the specified object is loaded.

Access Type Get or set

Data Type String

Syntax *string* = *project*.**StartView**
project.**StartView** = *string*

Remarks The **StartView** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
string	A string or variable of type String

StatusDate Property

Applies To **OPProject** object

Description The status date (Time Now) for the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**StatusDate**
project.**StatusDate** = *date*

Remarks The **StatusDate** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject

date	A variable of type Date

StepID Property

Applies To **OPStep** object

Description A string indicating a unique ID among steps assigned to a particular activity.

Access Type Get or set

Data Type String

Syntax *string* = *step*.**StepID**
step.**StepID** = *string*

Remarks The **StepID** property syntax uses these parts:

Part	Description
step	An object variable of type OPStep
string	A string or variable of type String

StepNumber Property

Applies To **OPStep** object

Description Unique long integer indicating the position (order) of the step among the steps.

Access Type Get or set

Data Type Integer

Syntax *integer* = *step*.**StepNumber**
step.**StepNumber** = *integer*

Remarks The **StepNumber** property syntax uses these parts:

Part	Description
step	An object variable of type OPStep
integer	An integer or variable of type Ineger

StopDate Property

Applies To **OPAvailability** object

Description The end of the availability period represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *availability*.**StopDate**
availability.**StopDate** = *date*

Remarks The **StopDate** property syntax uses these parts:

Part	Description
availability	An object variable of type OPAvailability
date	A date or variable of type Date

StopTime Property

Applies To **OPShift** object

Description The ending time of the shift represented by the specified object, measured in number of minutes since midnight.

Access Type Get or set

Data Type Integer

Syntax *integer* = *shift*.**StopTime**
shift.**StopTime** = *integer*

Remarks The **StopTime** property syntax uses these parts:

Part	Description
project	An object variable of type OPShift
integer	An integer or variable of type Integer

SubprojectFileName Property

Applies To **OPActivity** object

Description The name of the project file assigned to the activity represented by the specified object, when that activity is of the type External Subproject.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**SubprojectFileName**
object.**SubprojectFileName** = *string*

Remarks The **SubprojectFileName** property syntax uses these parts:

Part	Description
object	An object variable of type OPActivity
string	A string or variable of type String

SuccessorID Property

Applies To **OPPredecessor** object

Description The unique identifier for the successor activity associated with relationship represented by the specified object.

Access Type Get

Data Type String

Syntax *string* = *predecessor*.**SuccessorID**

Remarks The **SuccessorID** property syntax uses these parts:

Part	Description
predecessor	An object variable of type OPPredecessor
string	A variable of type String

Suppress Property

Applies To **OPAssignment** object, **OPResourceRecord** object, **OPActivityResource** object, **OPProjectResource** object

Description Gets or sets the suppress-for-scheduling flag for the object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Suppress**
object.**Suppress** = *string*

Remarks The **Suppress** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

SuppressRequirements Property

Applies To **OPActivity** object

Description A flag to have resource scheduling ignore the requirements for the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**SuppressRequirements**
activity.**SuppressRequirements =** *string*

Remarks The **SuppressRequirements** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
string	A string or variable of type String

TableName Property

Applies To **OPCalculatedField**, **OPField, OPFilter**, **OPGlobalEdit**, and **OPSort** objects

Description The name of the table to which this object applies.

Access Type Get

Data Type String

Syntax *string* = *object*.**TableName**

Remarks The **TableName** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type String
Table Names: Project Related	**Description**
ACT	Activity
ASG	Resource Assignment
BSA	Baseline Activity

BSU	Baseline Usage
CST	Resource Cost
PRJ	Project Directory
REL	Relationship
RSK	Risk Detail
SUB	Subproject
USE	Resource usage

Table Names: Resource Related	Description
AVL	Resource Availability
PSU	Project Summary
RES	Resource
RSL	Resource Escalation

Table Names: Code Related	Description
CDR	Code Data Table

TargetCost Property

Applies To **OPProject** object

Description The target cost of the project represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *project*.**TargetCost**
project. **TargetCost** = *double*

Remarks The **TargetCost** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
double	A double-precision floating-point number or variable of type Double

TargetFinish Property

Applies To **OPActivity** object

Description An external constraint applied to the finish of the activity represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**TargetFinish**
activity.**TargetFinish** = *date*

Remarks The **TargetFinish** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A date or variable of type Date

The TargetFinish property cannot be set to any value that would cause an invalid condition. For example, if the TargetFinishType property of an activity is currently "None", then the TargetFinish (date) property cannot be set. The AssignCurrentFieldSet and

SetCurrentFields methods can be used to change the fields represented by these properties in cases where both the TargetFinish and TargetFinishType must be set at the same time in order to avoid causing an invalid condition. Refer to the example "Using the AssignCurrentFieldSet Method" later in this document for more information.

TargetFinishDate Property

Applies To **OPProject** object

Description An external constraint applied to the finish of the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**TargetFinishDate**
project.**TargetFinishDate** = *date*

Remarks The **TargetFinishDate** property syntax uses these parts:

Part	Description
project	An object variable of type OPProject
date	A date or variable of type Date

TargetFinishType Property

Applies To **OPActivity** object, **OPProject** object

Description The type of constraint applied to the finish of the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**TargetFinishType**
object.**TargetFinishType** = *string*

Remarks The **TargetFinishType** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

Possible values are None, Not Earlier Than, Not Later Than, Fixed Target, or On Target.

The TargetFinishType property cannot be set to any value that would cause an invalid condition. For example, if the TargetFinish (date) property of an activity is blank, then its TargetFinishType cannot be changed from "None". The AssignCurrentFieldSet and SetCurrentFields methods can be used to change the fields represented by these properties in cases where both the TargetFinish and TargetFinishType must be set at the same time in order to avoid causing an invalid condition. Refer to the example "Using the AssignCurrentFieldSet Method" later in this document for more information.

TargetStart Property

Applies To **OPActivity** object

Description An external constraint applied to the activity start.

Access Type Get or set

Data Type Date

Syntax *date* = *activity*.**TargetStart**
activity.**TargetStart** = *date*

Remarks The **TargetStart** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
date	A date or variable of type Date

The TargetStart property cannot be set to any value that would cause an invalid condition. For example, if the TargetStartType property of an activity is currently "None", then the TargetStart (date) property cannot be set. The AssignCurrentFieldSet and SetCurrentFields methods can be used to change the fields represented by these properties in cases where both the TargetStart and TargetStartType must be set at the same time in order to avoid causing an invalid condition. Refer to the example "Using the AssignCurrentFieldSet Method" later in this document for more information.

TargetStartDate Property

Applies To **OPProject** object

Description An external constraint applied to the start of the project represented by the specified object.

Access Type Get or set

Data Type Date

Syntax *date* = *project*.**TargetStartDate**
project. **TargetStartDate** = *date*

Remarks The **TargetStartDate** property syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
date	A date or variable of type Date

TargetStartType Property

Applies To **OPActivity** object

Description The type of constraint applied to the start of the activity represented by the specified object.

Access Type Get or set

Data Type String

Syntax *string* = *activity*.**TargetStartType**
activity.**TargetStartType** = *string*

Remarks The **TargetStartType** property syntax uses these parts:

Part	Description
Activity	An object variable of type OPActivity
string	A string or variable of type String

Possible values are None, Not Earlier Than, Not Later Than, Fixed Target, or On Target.

The TargetStartType property cannot be set to any value that would cause an invalid condition. For example, if the TargetStart (date) property of an activity is blank, then its TargetStartType property cannot be changed from "None". The AssignCurrentFieldSet and SetCurrentFields methods can be used to change the fields represented by these properties in cases where both the TargetStart and TargetStartType must be set at the same time in order to avoid causing an invalid condition.

> **Attention:** Refer to the example "*Using the AssignCurrentFieldSet Method*" later in this document for more information.

Threshold Property

Applies To **OPActivityResource** object, **OPProjectResource** object, **OPResourceRecord** object

Description Gets or sets the threshold value for the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *resourcerecord*.**Threshold**
resourcerecord.**Threshold** = *double*

Remarks The **Threshold** property syntax uses these parts:

Part	Description
resourcerecord	An object variable of one of the types specified above.
double	A double-precision floating-point number or variable of type Double

Title Property

Applies To **OPWebWindow** object

Description Returns the title bar caption of the WebWindow represented by the specified **OPWebWindow** object.

Access Type Get

Data Type String

Syntax *string* = *webwindow*.**TotalFloat**

Remarks The **Title** property syntax uses these parts:

Part	Description
webwindow	An object variable of type OPWebWindow
string	A variable of type String

TotalFloat Property

Applies To **OPActivity** object, **OPPredecessor** object

Description The amount of time the activity or relationship represented by the specified object can be delayed without delaying the project or any target finish dates.

Access Type Get

Data Type String

Syntax *string* = *object*.**TotalFloat**

Remarks The **TotalFloat** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above.
string	A variable of type String

The string returned by the **TotalFloat** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

TotalFloatStdDev Property

Applies To **OPActivity** object

Description The standard deviation for the total float of the activity represented by the specified object.

Access Type Get

Data Type Long

Syntax *long* = *activity*.**TotalFloatStdDev**

Remarks The **TotalFloatStdDev** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
long	A variable of type Long

The string returned by the **TotalFloatStdDev** property will be in a format recognized by Open Plan as a valid duration, consisting of a number and a duration unit identifier. If the duration is zero, no duration unit identifier is appended.

TotalResourceCost Property

Applies To **OPActivity** object

Description The total cost of the resource usage for the activity represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *activity*.**TotalResourceCost**
activity.**TotalResourceCost** = *double*

Remarks The **TotalResourceCost** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
double	A double-precision floating-point number or variable of type Double

Turns Property

Description This property is no longer supported. It always returns a value of “0”.

Type Property

Applies To **OPActivity** object, **OPActivityResource** object, **OPBaselines** object, **OPField** object, **OPProjectResource** object, **OPResourceRecord** object

Description When applied to the **OPActivity** object, refers to the activity type: Subproject, External Subproject, Start Milestone, Finish Milestone, ASAP, ALAP, Discontinuous, Effort-Driven, Foreign Activity, Foreign Project, or Foreign Subproject. When applied to the **OPActivityResource**, **OPProjectResource**, or **OPResourceRecord** objects, refers to the resource type: Normal, Consumable, Perishable, Resource Pool, or Skill. When applied to the **OPBaselines** object, refers to the string indicating the set of dates used in the baseline: Early Dates, Late Dates, or Scheduled Dates. When applied to the **OPField** object,

Access Type Get or set

Data Type String

Syntax *string* **=** *object*.**Type**
object.**Type** = *string*

Remarks The **Type** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

When applied to the **OPBaselines** or **OPField** object, the only access type allowed is **Get**.

UnitCost Property

Applies To **OPActivityResource** object, **OPProjectResource** object, **OPResourceRecord** object, **OPSkill** object

Description Gets or sets the unit cost of the resource record represented by the specified object.

Access Type Get or set

Data Type Double

Syntax *double* = *resourcerecord*.**UnitCost**
resourcerecord.**UnitCost** = *double*

Remarks The **UnitCost** property syntax uses these parts:

Part	Description
resourcerecord	An object variable of one of the types listed above
double	A double-precision floating-point number or variable of type Double

Units Property

Applies To **OPActivityResource** object, **OPProjectResource** object, **OPResourceRecord** object

Description Gets or sets the unit in which the resource record represented by the specified object is measured.

Access Type Get or set

Data Type String

Syntax *string* = *resourcerecord*.**Units**
resourcerecord.**Units** = *string*

Remarks The **Units** property syntax uses these parts:

Part	Description
resourcerecord	An object variable of one of the types listed above
string	A string or variable of type String

URLAddress Property

Applies To **OPWebWindow** object

Description Returns the URL of the WebWindow represented by the specified **OPWebWindow** object.

Access Type Get

Data Type String

Syntax *string* = *webwindow*.**URLAddress**

Remarks The **URLAddress** property syntax uses these parts:

Part	Description
webwindow	An object variable of type OPWebWindow
string	A variable of type String

Used Property

Applies To **OPResourceUsage** object

Description The used quantity associated with the resource usage record represented by the specified object.

Access Type Get or Set

Data Type Double

Syntax *double* = *resourceusage*.**Used**
resourceusage.**Used** = *double*

Remarks The **Used** property syntax uses these parts:

Part	Description
resourceusage	An object variable of type OPResourceUsage
double	A double-precision floating-point number or variable of type Double

UserName Property

Applies To **OPField** object

Description For database fields, returns the display name of the field represented by the specified **OPField** object. For all other fields, returns the name of the field.

Access Type Get

Data Type String

Syntax *string* = *field*.**UserName**

Remarks The **UserName** property syntax uses these parts:

Part	Description
field	An object variable of type OPField
string	A variable of type String

Width Property

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, and **OPProject**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource.**

Description Gets or sets the width of the object window in pixels.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *object*.**Width**
object.**Width** = *integer*

Remarks The **Width** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
integer	An integer or variable of type Integer

Weight Property

Applies To **OPStep** object

Description A decimal field indicating the relative weight of a step among a group of steps.

Access Type Get or set

Data Type Decimal

Syntax *decimal* = *step*.**Weight**
step.**Weight** = *decimal*

Remarks The **Weight** property syntax uses these parts:

Part	Description
object	An object variable of type OPStep
decimal	A decimal or variable of type Decimal

Work Property

Applies To The following objects: **OPDate** and **OPStandardDay**

Description Gets or sets the flag that indicates whether or not the day represented by the specified object is a working day.

Access Type Get or set

Data Type String

Syntax *string* = *object*.**Work**
object.**Work** = *string*

Remarks The **Work** property syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String

X Property

Applies To **OPActivity** object

Description Gets or sets the activity box X coordinate for the activity represented by the specified object.

Access Type Get or set

Data Type Integer

Syntax *integer* = *activity*.**X**
activity.**X** = *integer*

Remarks The **X** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
integer	An integer or variable of type Integer

XPosition Property

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, and **OPProject**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource.**

Description Gets or sets the horizontal position of the object window.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *object*.**XPosition**
object.**XPosition** = *integer*

Remarks The **XPosition** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
integer	An integer or variable of type Integer

Y Property

Applies To **OPActivity** object

Description Gets or sets the activity box Y coordinate for the activity represented by the specified object.

Access Type Get or set

Data Type Long

Syntax *integer* = *activity*.**Y**
activity.**Y** = *integer*

Remarks The **Y** property syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
integer	An integer or variable of type Integer

YPosition Property

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, and **OPProject**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource.**

Description Gets or sets the vertical position of the object window.

Access Type Get or Set

Data Type Integer

Syntax *integer* = *object*.**YPosition**
object.**YPosition** = *integer*

Remarks The **YPosition** property syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
integer	An integer or variable of type Integer

Methods

Attention: Refer to the Visual Basic Language Reference for more information about the **Set** statement.

For a list of table names, see the **TableName** property in Properties.

Activate Method

Applies To **OPIcon** object, **OPView** object

Description Activates the specified object. Basically performs the same action that double-clicking the object's icon would perform.

Syntax *object.***Activate**

Remarks The **Activate** method syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above

ActivateByFilename Method

Applies To The following collections: **OPBarcharts**, **OPGraphs**, **OPNetworks**, **OPSpreadsheets**, and **OPViews**.

Description Activates the view represented by the specified object that matches the specified name.

Syntax *boolean* = *object*.**ActivateByFilename**(*name*)

Remarks The **ActivateByFilename** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
boolean	A variable of type Boolean
name	A string or variable of type String representing the name of the view to be activated

The **ActivateByFilename** method returns 0 if unsuccessful.

ActivateByFileNameEx Method

Applies To The following collections: **OPBarcharts**, **OPGraphs**, **OPNetworks**, **OPSpreadsheets**, and **OPViews**.

Description If the named view is not opened, it is opened by this method; otherwise, the view is made the active window in Open Plan.

Syntax *Set oViewObject = OPViews.***ActivateByFileNameEx***(sViewName)*

Remarks The **ActivateByFileNameEx** method syntax uses these parts:

Part	Description
sViewName	The name of the project view to be opened or activated (made the active window in OP)
oViewObject	An OPP30.OPView object (the result of the method). The caller should check for Nothing before using the object as it will not be set if the view does not exist.

ActiveProject Method

Applies To **OPCreateApplication33** object

Description Returns the name of the current open project. Returns null if no project is opened, or an open project does not have the focus.

Syntax *projectname* = *application*.**ActiveProject**

Remarks The **ActiveProject** method syntax uses these parts:

Part	Description
application	An object variable of type OPCreateApplication33
projectname	A variable of type String

ActiveView Method

Applies To **OPCreateApplication33** object

Description Returns the name of the current open view. Returns null if no view is opened, or an open view does not have the focus.

Syntax *viewname* = *application*.**ActiveView**

Remarks The **ActiveView** method syntax uses these parts:

Part	Description
application	An object variable of type OPCreateApplication33
viewname	A variable of type String

Activities Method

Applies To **OPProject** object

Description Retrieves the **OPActivities** collection object. When *index* is used as an argument to the **Activities** method, retrieves the **OPActivity** object mapped into that index. When *activityid* is used as an argument to the **Activities** method, returns the **OPActivity** object whose ID matches the argument.

Syntax
Set *activities* = *project*.**Activities**
Set *activity* = *project*.**Activities.Item**(*index*)
Set *activity* = *project*.**Activities.Item**(*activityid*)

Remarks The **Activities** method syntax uses these parts:

Part	Description
activities	An object variable of type OPActivities
activity	An object variable of type OPActivity
project	An object variable of type OPProject
index	An integer or variable of type that uniquely identifies an OPActivity object within the OPActivities collection.
activityid	A string or variable of type String representing the unique identifier for the requested OPActivity object.

ActivityResources Method

Applies To **OPActivity** object

Description Retrieves the **OPActivityResources** collection object. When *index* is used as an argument to the ActivityResources method, retrieves the **OPActivityResource** object mapped into that object.

Syntax **Set** *activityresources* = *object.***ActivityResources**
Set *activityresource* = *object.***ActivityResources.Item***(index)*

Remarks The **ActivityResources** method uses these parts:

Part	Description
activityresources	An object variable of type OPAssignments
activityresource	An object variable of type OPAssignment
object	An object variable of type OPActivity
index	An integer or variable of type Integer that uniquely identifies an OPActivityResource object within the OPActivityResources collection

Add Method

Applies To The following collections: **OPActivities**, **OPAssignments**, **OPAvailabilities**, **OPCalculatedFields**, **OPCalendar**, **OPCategories**, **OPCode**, **OPCosts**, **OPExtraWorkDays**, **OPFilters**, **OPGlobalEdits**, **OPHolidays**, **OPNotes**, **OPPredecessors**, **OPProjectCode**, **OPProjectCodes**, **OPProjectResources**, **OPReportingCalendar**, **OPResource**, **OPResourceUsages**, **OPRollups**, **OPShifts**, **OPSkills**, **OPSorts**, **OPSteps**, and **OPWebWindows**

Description Adds an object to the specified collection. The object to be added is described by the parameters passed to the **Add** method, which vary depending on the object to which the method has been applied.

Syntax The syntax of the **Add** method varies depending on the collection to which the method has been applied. The syntax for each collection is documented separately below.

Remarks The **Add** method can be combined with the **Set** statement to return the object being added to the collection, that is, **Set** *object* = *collection*.**Add**(*parameter1*,...). Using the **Add** method in this way makes it possible to detect the error condition generated when the method fails to add an object to the collection. If an attempt to get or set a property of the newly added object returns Error 91 (Object Variable Not Set), the object has not been added to the collection.

OPActivities collection

Syntax *activities*.**Add** *activityid*

Set *activity* = *activities*.**Add**(*activityid)*

Remarks When applied to the **OPActivities** collection, the **Add** method syntax uses these parts:

Part	Description
activity	An object variable of type OPActivity
activities	An object variable of type OPActivities
activityid	The unique identifier of an activity object

OPAssignments collection

Syntax *assignments*.**Add** *resourceid*

Set *assignment* = *assignments*.**Add**(*resourceid)*

Remarks When applied to the **OPAssignments** collection, the **Add** method syntax uses these parts:

Part	Description
assignments	An object variable of type OPAssignments
assignment	An object variable of type OPAssignment
resourceid	The unique identifier of a resource record object

OPAvailabilities collection

Syntax *availabilities*.**Add** *startdate*, *stopdate*

Set *availability* = *availabilities*.**Add**(*startdate*, *stopdate*)

Remarks When applied to the **OPAvailabilities** collection, the **Add** method syntax uses these parts:

Part	Description
availabilities	An object variable of type OPAvailabilities
availability	An object variable of type OPAvailability
startdate	The date that begins a period of resource availability
stopdate	The date that ends a period of resource availability

OPCalculatedFields collection

Syntax *calculatedfields*.**Add** *calcfieldname, expression, returntype, appliestotable*

Set *calculatedfield* = *calculatedfields*.**Add**(*calcfieldname, expression, returntype, appliestotable)*

Remarks When applied to the **OPCalculatedFields** collection, the **Add** method syntax uses these parts:

Part	Description
calculatedfields	An object variable of type OPCalculatedFields
calculatedfield	An object variable of type OPCalculatedField
calcfieldname	The name of the calculatedfield to be added.
expression	The expression that defines the object to be added.
returntype	The type of value returned by the new calculated field.
appliestotable	The table to which the added object applies.

OPCalendar collection

Syntax *calendar*.**Add** *calendarrecordname*

Set *calendarrecord* = *calendar*.**Add**(*calendarrecordname)*

Remarks When applied to the **OPCalendar** collection, the **Add** method syntax uses these parts:

Part	Description
calendar	An object variable of type OPCalendar
calendarrecord	An object variable of type OPCalendarRecord
calendarrecordname	The name of a calendar within a calendar file

OPCategories collection

Syntax *categories*.**Add** *categoryname*

Set *category* = *categories*.**Add**(*categoryname)*

Remarks When applied to the **OPCategories** collection, the **Add** method syntax uses these parts:

Part	Description
categories	An object variable of type OPCategories
category	An object variable of type OPCategory
categoryname	The name of a category

OPCode collection

Syntax *code*.**Add** *coderecordcode*

Set *coderecord* = *code*.**Add**(*coderecordcode)*

Remarks When applied to the **OPCode** collection, the **Add** method syntax uses these parts:

Part	Description
code	An object variable of type OPCode
coderecord	An object variable of type OPCodeRecord
coderecordcode	The unique identifier of a code record object

OPCosts collection

Syntax *costs*.**Add** *activityid*, *resourceid*

Set *cost* = *costs*.**Add**(*activityid*, *resourceid)*

Remarks When applied to the **OPCosts** collection, the **Add** method syntax uses these parts:

Part	Description
costs	An object variable of type OPCosts
cost	An object variable of type OPCost
activityid	The unique identifier of an activity object
resourceid	The unique identifier of a resource record object

OPExtraWorkDays collection

Syntax *extraworkdays*.**Add** *date*
Set *dateobject* = *extraworkdays*.**Add**(*date)*

Remarks When applied to the **OPExtraWorkDays** collection, the **Add** method syntax uses these parts:

Part	Description
extraworkdays	An object variable of type OPExtraWorkDays
dateobject	An object variable of type OPDate
date	A date or variable of type Date

OPFilters collection

Syntax *filters*.**Add** *filtername, expression, appliestotable*
Set *filter* = *filters*.**Add**(*filtername, expression, appliestotable)*

Remarks When applied to the **OPFilters** collection, the **Add** method syntax uses these parts:

Part	Description
filters	An object variable of type OPFilters
filter	An object variable of type OPFilter
filtername	The name of the filter to be added
expression	The expression that defines the object to be added.

appliestotable	The table to which the added object applies

OPGlobalEdits collection

Syntax

globaledits.**Add** *globaleditname, expression, appliestotable, globaleditfield*
Set *globaledit* = *globaledits*.**Add**(*globaleditname, expression, appliestotable, globaleditfield)*

Remarks

When applied to the **OPGlobalEdits** collection, the **Add** method syntax uses these parts:

Part	Description
globaledits	An object variable of type OPGlobalEdits
globaledit	An object variable of type OPGlobalEdit
globaleditname	The name of the globaledit to be added.
expression	The expression that defines the object to be added.
appliestotable	The table to which the added object applies.
globaleditfield	The name of the field which is to be modified by this global edit.

OPHolidays collection

Syntax

holidays.**Add** *date*
Set *dateobject* = *holidays*.**Add**(*date)*

Remarks

When applied to the **OPHolidays** collection, the **Add** method syntax uses these parts:

Part	Description
holidays	An object variable of type OPHolidays
dateobject	An object variable of type OPDate
date	A date or variable of type Date

OPNotes collection

Syntax

notes.**Add** *categoryname*
Set *note* = *notes*.**Add**(*categoryname, notetext)*

Remarks

When applied to the **OPNotes** collection, the **Add** method syntax uses these parts:

Part	Description
notes	An object variable of type OPNotes
note	An object variable of type OPNote
categoryname	The name of a category
notetext	The text of the note

OPPredecessors collection

Syntax

predecessors.**Add** *activityid*, *reltype*

Set *predecessor* = *predecessors*.**Add**(*activityid*, *reltype)*

Remarks

When applied to the collection, the **Add** method syntax uses these parts:

Part	Description
predecessors	An object variable of type OPPredecessors
predecessor	An object variable of type OPPredecessor
activityid	The unique identifier of an activity object
reltype	A string representing one of the four relationship types: FS (Finish to Start), FF (Finish to Finish), SS (Start to Start) or SF (Start to Finish)

OPProjectCode collection

Syntax

projectcode.**Add** *coderecordcode*

Set *coderecord* = *projectcode*.**Add**(*coderecordcode)*

Remarks When applied to the collection, the **Add** method syntax uses these parts:

Part	Description
projectcode	An object variable of type OPProjectCode
coderecord	An object variable of type OPCodeRecord
coderecordcode	The unique identifier of a code record object

OPProjectCodes collection

Syntax *projectcodes*.**Add** *codefilename*, *fieldname*

Set *projectcode* = *projectcodes*.**Add**(*codefilename*, *fieldname)*

Remarks When applied to the collection, the **Add** method syntax uses these parts:

Part	Description
projectcodes	An object variable of type OPProjectCodes
projectcode	An object variable of type OPProjectCode
codefilename	The fully qualified path name of a code file
fieldname	The name of the field in the activity table that will store the code values from a code file

OPProjectResources collection

Syntax *projectresources*.**Add** *resourceid*

Set *resourcerecord* = *projectresource*.**Add**(*resourceid)*

Remarks When applied to the **OPProjectResource** collection, the **Add** method syntax uses these parts:

Part	Description
projectresources	An object variable of type OPProjectResources
projectresource	An object variable of type OPResourceRecord
resourceid	The unique identifier of a resource record object

OPReportingCalendar collection

Syntax *reportingcalendar*.**Add**(*date, label)*

Set *reportingcalendarrecord* = *reportingcalendar*.**Add**(*date, label*)

Remarks When applied to the **OPReportingCalendar** collection, the **Add** method syntax uses these parts:

Part	Description
reportingcalendar	An object variable of type OPReportingCalendar
reportingcalendarrecord	An object variable of type OPReportingCalendarRecord
date	A date or variable of type Date
label	A string or variable of type String

OPResource collection

Syntax *resource*.**Add** (*resourceid)*

Set *resourcerecord* = *resource*.**Add**(*resourceid*)

Remarks When applied to the **OPResource** collection, the **Add** method syntax uses these parts:

Part	Description
resource	An object variable of type OPResource

resourcerecord	An object variable of type OPResourceRecord
resourceid	The unique identifier of a resource record object.

OPRollups collection

Syntax *rollups*.**Add** *name, definition, shareable*

Set *rollup* = *rollups*.**Add**(*name, definition, shareable*)

Remarks When applied to the **OPShifts** collection, the **Add** method syntax uses these parts:

Part	Description
rollups	An object variable of type OPRollups
rollup	An object variable of type OPRollup
name	A string or variable of type String
definition	A string or variable of type String
shareable	A long integer or variable of type Long

OPShifts collection

Syntax *shifts*.**Add** *starttime, stoptime*

Set *shift* = *shifts*.**Add**(*starttime, stoptime*)

Remarks When applied to the **OPShifts** collection, the **Add** method syntax uses these parts:

Part	Description
shifts	An object variable of type OPShifts
shift	An object variable of type OPShifts
starttime	The number of minutes past midnight that a shift begins
stoptime	The number of minutes past midnight that a shift ends

OPSkills collection

Syntax *skills*.**Add** *skillid, description, unitcost*

Set *skill* = *skills*.**Add**(*skillid, description, unitcost*)

Remarks When applied to the **OPSkills** collection, the **Add** method syntax uses these parts:

Part	Description
skills	An object variable of type OPSkills
skill	An object variable of type OPSkill
skillid	A string or variable of type String
description	A string or variable of type String
unitcost	A double-precision floating-point number or variable of type Double

OPSorts collection

Syntax *sorts*.**Add** *sortname, expression, appliestotable*

Set *sort* = *sorts*.**Add**(*sortname, expression, appliestotable)*

Remarks When applied to the **OPSorts** collection, the **Add** method syntax uses these parts:

Part	Description
sorts	An object variable of type OPSorts
sort	An object variable of type OPSort
sortname	The name of the sort to be added

expression	The expression that defines the object to be added.

OPSteps collection

Syntax *steps*.**Add** *stepID, description, weight*

Set *step* = *steps*.**Add**(*stepID, description, weight)*

Remarks When applied to the **OPSteps** collection, the **Add** method syntax uses these parts:

Part	Description
steps	An object variable of type OPSteps
step	An object variable of type OPStep
stepID	The unique name of the step among steps assigned to a certain activity.
description	The description for the step to be added.
weight	A decimal value indicating the relative weight of the step among the steps assigned to the activity.

OPWebWindows collection

Syntax *webwindows*.**Add** *title, URL, querystring*

Set *webwindow* = *webwindows*.**Add**(*title, URL, querystring)*

Remarks When applied to the **OPWebWindows** collection, the **Add** method syntax uses these parts:

Part	Description
Webwindows	An object variable of type OPWebWindows
webwindow	An object variable of type OPWebWindow
title	A string or variable of type String
URL	A string or variable of type String
querystring	A string or variable of type String

AddAll Method

Applies To **OPHolidays** collection, **OPShifts** collection

Description When applied to the **OPHolidays** collection, adds a list of dates to the collection. When applied to the **OPShifts** collection, adds a list of up to ten pairs of times to the collection.

Syntax *boolean* = *object*.**AddAll**(*string*)

Remarks The **AddAll** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String containing a comma-delimited list of dates or times, depending on the collection to which the method is applied
boolean	A variable of type Boolean

The **AddAll** method returns 0 if unsuccessful.

AddExtended Method

Applies To **OPSteps** collection

Description Adds an object to the **OPSteps** collection. The object to be added is described by the parameters passed to the **AddExtended** method.

Syntax The syntax of the **AddExtended** method uses the syntax of the **OPSteps** collection, which is documented below.

Remarks The **AddExtended** method can be combined with the **Set** statement to return the object being added to the collection, that is, **Set** *object* = *collection*.**AddExtended**(*parameter1*,). Using the **AddExtended** method in this way makes it possible to detect the error condition generated when the method fails to add an object to the collection. If an attempt to get or set a property of the newly added object returns Error 91 (Object Variable Not Set), the object has not been added to the collection.

OPSteps collection

Syntax *steps*.**AddExtended** *stepID, description, weight*

Set *step* = *steps*.**AddExtended**(*stepID, description, weight)*

Remarks When applied to the **OPSteps** collection, the **AddExtended** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A string or variable of type String containing a comma-delimited list of dates or times, depending on the collection to which the method is applied
Boolean	A variable of type Boolean
percentComplete	A decimal value indicating the percent complete of the step (0 to 100). Note that this value will be ignored if the **Allow % Complete on Steps** project preference is not selected.
stepNumber	An integer value indicating the order of the step. Step number 100, for example, would precede step number 200. Step numbers must be non-negative integers.

AddView Method

Applies To **OPBarcharts**, **OPGraphs**, **OPNetworks**, **OPSpreadsheets**, **OPViews** collections

Description Adds a view to the specified collection.

Syntax *boolean* **=** *collection***.AddView(***name***)**

Remarks The **AddView** method syntax uses these parts:

Part	Description
boolean	A variable of type Boolean indicating a success or failure of the operation.
collection	An object variable of one of the types listed above.
name	The name of the view to be added to the collection.

The method will return false if the specified view name does not exist.

Apply Method

Applies To **OPGlobalEdit, OPBatchGlobalEdit, OPRollup** objects

Description Applies the global edit definition to the table for which it was defined.

Syntax *boolean* = *globaledit*.**Apply**

Remarks The **Apply** method syntax uses these parts:

Part	Description
globaledit	An object variable of type OPGlobalEdit
boolean	A variable of type Boolean

The **Apply** method returns 0 if unsuccessful.

ApplyFilter Method

Applies To **OPCreateApplication33** object

Description Applies a specified filter to the active view.

Syntax *Boolean* = *application*.**ApplyFilter***(filter)*

Remarks The **ApplyFilter** method syntax uses these parts:

Part	Description
boolean	A variable of type Boolean indicating the success or failure of the method. The method will fail if there is no view active, if the filter specified is inappropriate for the active view, or if the filter name or expression specified is invalid.
application	An object variable of type OPCreateApplication33
filter	A string or variable of type String representing the name of a defined filter or filter expression.

AssignCurrentFieldSet Method

Applies To The following collections: **OPActivities**, **OPActivityResources**, **OPAssignments**, **OPAvailabilities**, **OPCode, OPCosts**, **OPPredecessors**, **OPProjectResources**, **OPResource**, and **OPSteps**.

Description Defines the list of fields to be used with the GetCurrentFields or SetCurrentFields methods.

Syntax *boolean* = *collection*.**AssignCurrentFieldSet**(*string*)

Remarks The **AssignCurrentFieldSet** method syntax uses these parts:

Part	Description
collection	An object variable of one of the types listed above
boolean	A variable of type Boolean
string	A string or variable of type String containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **AssignCurrentFieldSet** method returns 0 if unsuccessful for any field in *string*, but the method will not fail if an invalid field name is included in *string*. Field names for date and enumerated fields in *string* can be optionally followed by a formatting string enclosed in square brackets. This formatting string will be used to format the data when it is retrieved using the GetCurrentFields method, and has no effect on the behavior of the

SetCurrentFields method. Formatting strings will be ignored if applied inappropriately; for example, a date format will be ignored if specified for a decimal field.

For date fields, the format string must be a valid date format recognized by Open Plan. For enumerated fields, the format string [s] will return the short, language-independent form of the enumerated value. The following example shows how these format strings might appear:

```
ESDATE[%D%A%Y]|ACT_TYPE[s]
```

> **Attention:** Refer to the Date Formats table in this document for the parameters used to define date formats.

Assignments Method

Applies To **OPActivity** object

Description Retrieves the **OPAssignments** collection object. When *index* is used as an argument to the **Assignments** method, retrieves the **OPAssignment** object mapped into that index.

Syntax **Set** *assignments* = *activity*.**Assignments**

Set *assignment* = *activity*.**Assignments.Item**(*index)*

Remarks The **Assignments** method uses these parts:

Part	Description
assignments	An object variable of type OPAssignments
assignment	An object variable of type OPAssignment
activity	An object variable of type OPActivity
index	An integer or variable of type Integer that uniquely identifies an OPAssignment object within the OPAssignments collection

AssignSkill Method

Applies To **OPResourceRecord** object

Description Assigns an existing skill to the resource represented by the specified **OPResourceRecord** object.

Syntax *boolean* **=** *resourcerecord.***AssignSkill**(*skillid*)

Remarks The **AssignSkill** method syntax uses these parts:

Part	Description
Resourcerecord	An object variable of type OPResourceRecord
Boolean	A variable of type Boolean
skillid	A string or variable of type String

AutoProgress Method

Applies To **OPProject** object

Description Displays the Automatic Progress dialog box.

Syntax *project.***AutoProgress**

Remarks The **AutoProgress** method syntax uses these parts:

Part	Description
Project	An object variable of type OPProject

AutoProgressEx Method

Applies To **OPProject** object

Description Performs Automatic Progress calculations, using either parameters specified via the **OPProject** object's AutoProgress properties or through the Automatic Progress dialog box. **This method supersedes the existing AutoProgress method.**

Syntax *success* **=** *project.***AutoProgressEx**(*showdialog*)

Remarks The **AutoProgressEx** method syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
Success	A variable of type Boolean indicating if the method was successful.
Showdialog	A Boolean value or variable of type Boolean indicating if the Automatic Progress dialog box is to be displayed (a value of False will suppress the display of the Automatic Progress dialog box).

The **AutoProgressEx** method returns False under the following conditions:

- The Automatic Progress dialog box was displayed and the user selected Cancel.
- An invalid filter name has been specified through the AutoProgFilter property.
- The project Status date is empty or invalid.
- The resource progress start date (AutoProgStartDate property) is empty or invalid when the resource progress option has been set (AutoProgResource property).
- The resource progress end date (AutoProgEndDate) is invalid or earlier than the resource progress start date when the resource progress option has been set. Note that the resource progress end date is allowed to be empty.

Availabilities Method

Applies To **OPActivityResource** object, **OPProjectResource** object, **OPResourceRecord** object

Description Retrieves the **OPAvailabilities** collection object. When *index* is used as an argument to the **Availabilities** method, retrieves the **OPAvailability** object mapped into that index.

Syntax **Set** *availabilities* = *resourcerecord*.**Availabilities**
Set *availability* = *resourcerecord*.**Availabilities.Item**(*index*)

Remarks The **Availabilities** method uses these parts:

Part	Description
Availabilities	An object variable of type OPAvailabilities
Assignment	An object variable of type OPAvailability
Resourcerecord	An object variable of one of the types listed above
Index	An integer or variable of type Integer that uniquely identifies an OPAvailability object within the OPAvailabilities collection

Barcharts Method

Applies To **OPFileCabinet** object, **OPProject** object

Description Retrieves the **OPBarcharts** or **OPFCBarcharts** collection object. When *index* is used as an argument to the **Barcharts** method, retrieves the **OPFCView** or **OPView** object mapped into that index. When *name* is used as an argument to the **Barcharts** method, retrieves the **OPFCView** or **OPView** object that matches the specified name.

Syntax

Set *barcharts* = *object*.**Barcharts**
Set *barchart* = *object*.**Barcharts.Item**(*index*)
Set *barchart* = *object*.**Barcharts.Item**(*name*)

Remarks The **Barcharts** method syntax uses these parts:

Part	Description
object	An object variable of type OPFileCabinet or OPProject
Barcharts	An object variable of type OPFCBarcharts or OPBarcharts
barchart	An object variable of type OPFCView or OPView
index	An integer or variable of type Integer that uniquely identifies an OPFCView or OPView object within the OPFCBarcharts or OPBarcharts collection
name	A string or variable of type String representing the identifier for the requested OPFCView or OPView object

BaselinesList Method

Applies To **OPProject** object

Description Retrieves the **OPBaselinesList** collection object. When *index* is used as an argument to the **OPBaselinesList** method, retrieves the **OPBaselines** object mapped into that index. When *baselinename* is used as an argument to the **OPBaselinesList** method, returns the **OPBaselines** object whose name matches the argument.

Syntax

Set *baselineslist* = *project*.**BaselinesList**
Set *baseline* = *project*.**BaselinesList.Item**(*index*)
Set *baseline* = *project*.**BaselinesList.Item**(*baselinename*)

Remarks The **BaselinesList** method syntax uses these parts:

Part	Description
baselineslist	An object variable of type OPBaselinesList
baseline	An object variable of type OPBaselines
project	An object variable of type OPProject
index	An integer or variable of type Integer that uniquely identifies an OPBaselines object within the OPBaselinesList collection
baselinename	A string or variable of type String representing the unique name of the requested OPBaselines object

BatchGlobalEdits Method

Applies To **OPProject** object

Description Retrieves the **OPBatchGlobalEdits** collection object. When *index* is used as an argument to the **BatchGlobalEdits** method, retrieves the **OPBatchGlobalEdit** object mapped into that index. When *name* is used as an argument to the **BatchGlobalEdits** method, retrieves the **OPBatchGlobalEdit** object that matches the specified name.

Syntax **Set** *batchglobaledits* = *object*. **BatchGlobalEdits**
Set *batchglobaledit* = *object*. **BatchGlobalEdits.Item**(*index*)
Set *batchglobaledit* = *object*. **BatchGlobalEdits.Item**(*name*)

Remarks The **BatchGlobalEdits** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above
batchglobaledits	An object variable of type OPBatchGlobalEdits
batchglobaledit	An object variable of type OPBatchGlobalEdit
index	An integer or variable of type Integer that uniquely identifies an OPBatchGlobalEdit object to be retrieved
name	A string or variable of type String representing the unique name of the requested OPBatchGlobalEdit object

CalculateCost Method

Applies To **OPProject** object

Description Displays the Calculate Costs dialog box.

Syntax *project.***CalculateCost**

Remarks The **CalculateCost** method syntax uses these parts:

Part	Description
Project	An object variable of type OPProject

CalculateCostEx Method

Applies To **OPProject** object

Description Performs Cost Calculations, using either parameters specified via the **OPProject** object's CalcCost properties or through the Cost Calculations dialog box. **This method supersedes the existing CalculateCost method.**

Syntax *success* **=** *project.* **CalculateCostEx**(*showdialog*)

Remarks The **CalculateCostEx** method syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
success	A variable of type Boolean indicating if the method was successful.
showdialog	A Boolean value or variable of type Boolean indicating if the Cost Calculations dialog box is to be displayed (a value of False suppresses the display of the Cost Calculations dialog box).

The **CalculateCostEx** method returns False if unsuccessful.

CalculatedFields Method

Applies To **OPProject** object, **OPResource** and **OPProjectResources** collections

Description Retrieves the **OPCalculatedFields** collection object. When *index* is used as an argument to the **CalculatedFields** method, retrieves the **OPCalculatedField** object mapped into that index. When *name* is used as an argument to the **CalculatedFields** method, retrieves the **OPCalculatedField** object that matches the specified name.

Syntax

Set *calculatedfields* = *object.***CalculatedFields**
Set *calculatedfield* = *object.***CalculatedFields.Item**(*index*)
Set *calculatedfield* = *object.***CalculatedFields.Item**(*name*)

Remarks

The **CalculatedFields** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
calculatedfields	An object variable of type OPCalculatedFields
calculatedfield	An object variable of type OPCalculatedField
index	An integer or variable of type Integer that uniquely identifies an OPCalculatedField object within the OPCalculatedFields collection
name	A string or variable of type String representing the name of the OPCalculatedField object to be retrieved

Calendars Method

Applies To

OPFileCabinet object, **OPCreateApplication33** object

Description

Retrieves the **OPFCCalendars** or **OPCalendars** collection object. When *index* is used as an argument to the **Calendars** method, retrieves the **OPIcon** or **OPCalendar** object mapped into that index. When *name* is used as an argument to the **Calendars** method, retrieves the **OPIcon** or **OPCalendar** object that matches the specified name.

Syntax

Set *calendars* = *object.***Calendars**
Set *calendar* = *object.***Calendars.Item(***index***)**
Set *calendar* = *object.***Calendars.Item(***name***)**

Remarks

The **Calendars** method syntax uses these parts:

Part	Description
object	An object variable of type OPFileCabinet or OPCreateApplication33
calendars	An object variable of type OPFCCalendars or OPCalendars
calendar	An object variable of type OPIcon or OPCalendar
index	An integer or variable of type Integer that uniquely identifies an OPIcon object within the OPFCCalendars collection or an OPCalendar object within the OPCalendars collection
name	A string or variable of type String representing the name of the OPIcon or OPCalendar object to be retrieved

Categories Method

Applies To

The following objects: **OPCode**, **OPProject**, **OPResource**, **OPProjectCode**, or **OPProjectResources**. The following collection: **OPActivities**.

Description

When applied to an object, retrieves the **OPCategories** collection object. When *index* is used as an argument to the **Categories** method, retrieves the **OPCategory** object mapped into that index. When *name* is used as an argument to the **Categories** method, retrieves the **OPCategory** object that matches the specified name. When applied to the collection, returns the Notes Categories associated with the **OPActivities** collection.

Syntax

Set *categories* = *object.***Categories**
Set *category* = *object.* **Categories.Item(***index***)**
Set *category* = *object.* **Categories.Item(***name***)**

Remarks The **Categories** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above
categories	An object variable of type OPCategories
category	An object variable of type OPCategory
index	An integer or variable of type Integer that uniquely identifies an OPCategory object within the OPCategories collection
name	A string or variable of type String representing the name of the OPCategory object to be retrieved

Close Method

Applies To **OPWebWindow** object

Description Closes the WebWindow represented by the specified **OPWebWindow** object.

Syntax *webwindow.***Close**

Remarks The **Close** method syntax uses these parts:

Part	Description
webwindow	An object variable of type OPWebWindow

CloseView Method

Applies To **OPProject** object

Description Closes the view that matches the specified name.

Syntax *boolean* = *project.***CloseView**(*saveview*, *name*)

Remarks The **CloseView** method syntax uses these parts:

Part	Description
Project	An object variable of type OPProject
boolean	A variable of type Boolean
Saveview	A boolean value or variable of type Boolean, where True indicates the view should be saved before closing
name	A string or variable of type String representing the name of the view to be closed

The **CloseView** method returns 0 if unsuccessful.

CodeFiles Method

Applies To **OPProject** object

Description Retrieves the **OPProjectCodes** collection object. When *index* is used as an argument to the **CodeFiles** method, retrieves the **OPProjectCode** object mapped into that index. When *name* is used as an argument to the **CodeFiles** method, retrieves the **OPProjectCode** object that matches the specified name.

Syntax
Set *projectcodes* = *project.***CodeFiles**
Set *projectcode* = *project.***CodeFiles.Item(***index***)**
Set *projectcode* = *project.***CodeFiles.Item(***name***)**

Remarks The **CodeFiles** method syntax uses these parts:

Part	Description

project	An object variable of type OPProject
projectcodes	An object variable of type OPProjectCodes
projectcode	An object variable of type OPProjectCode
index	An integer or variable of type Integer that uniquely identifies an OPProjectCode object within the OPProjectCodes collection
name	A string or variable of type String representing the name of the OPProjectCode object to be retrieved

Codes Method

Applies To **OPFileCabinet** object, **OPCreateApplication33** object

Description Retrieves the **OPFCCodes** or **OPCodes** collection object. When *index* is used as an argument to the **Codes** method, retrieves the **OPIcon** or **OPCode** object mapped into that index. When *name* is used as an argument to the **Codes** method, retrieves the **OPIcon** or **OPCode** object that matches the specified name.

Syntax

Set *codes* = *object.***Codes**
Set *code* = *object.***Codes.Item(***index***)**
Set *code* = *object.***Codes.Item(***filename***)**

Remarks The **Codes** method syntax uses these parts:

Part	Description
object	An object variable of type OPFileCabinet or OPCreateApplication33
codes	An object variable of type OPFCCodes or OPCodes
code	An object variable of type OPFCode or OPCode
index	An integer or variable of type Integer that uniquely identifies an OPIcon object within the OPFCCodes collection or an OPCode object within the OPCodes collection
name	A string or variable of type String representing the name of the OPIcon or OPCode object to be retrieved

CollapseAll Method

Applies To **OPView** Object

Description Allows the user to initiate a “collapse all” for the corresponding Open Plan view.

Syntax *bResult* = *OPView.***CollapseAll**

Remarks The method syntax uses these parts:

Part	Description
bResult	The Boolean result of the method. The method will return True if the view is open; False if closed.

Conceal Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Hides the window represented by the specified object and passes activation to another window.

Syntax *object.***Conceal**

Remarks The **Conceal** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above.

Use the **Restore** method to reverse the effects of **Conceal**.

Copy Method

Applies To **OPCalendar** collection

Description Adds an **OPCalendarRecord** object to the collection using a specified **OPCalendarRecord** object as a template. The new **OPCalendarRecord** object will have the same properties as the source **OPCalendarRecord** object.

Syntax *calendar*.**Copy** *source*, *target*
Set *calendarrecord* = *calendar*.**Copy**(*source, target)*

Remarks The **Copy** method syntax uses these parts:

Part	Description
Calendar	An object variable of type OPCalendar
calendarrecord	An object variable of type OPCalendarRecord
source	A string or variable of type String representing the name of the OPCalendarRecord object to be copied
target	A string or variable of type String representing the name of the OPCalendarRecord object to be created

Costs Method

Applies To **OPProject** object

Description Retrieves the **OPCosts** collection object. When *index* is used as an argument to the **OPCosts** method, retrieves the **OPCost** object mapped into that index.

Syntax **Set** *costs* = *project*.**Costs**
Set *cost* = *project*.**Costs**(*index*)

Remarks The **Costs** method syntax uses these parts:

Part	Description
Costs	An object variable of type OPCosts
cost	An object variable of type OPCost
project	An object variable of type OPProject
index	An integer or variable of type Integer that uniquely identifies an OPCost object within the OPCosts collection

CreateBackup Method

Applies To **OPProject** object

Description Create a backup for the project represented by the specified **OPProject** object.

Syntax *boolean* = *object*.**CreateBackup**(*filename, overwrite, baselines*)

Remarks The **CreateBackup** method syntax uses these parts:

Part	Description
boolean	A variable of type Boolean
object	An object variable of type OPProject

filename	A string or variable of type String that contains the full path name of the backup file. The extension may be omitted, and is changed to .BK3 if it is anything else.
overwrite	A Boolean value or variable of type Boolean that determines the behavior of the operation if the specified backup file already exists. If TRUE, the file is overwritten. If FALSE, a file dialog is displayed.
baselines	A Boolean value or variable of type Boolean that determines whether baselines in the specified project are to be included in the backup file. If TRUE, then baselines are included in the backup file. If FALSE, then baselines will not be included in the backup file. If the project does not contain baselines this parameter has no affect. The baselines parameter is used for Open Plan 2.x backward compatibility only. It is not used and has no effect in any Open Plan 8.x version.

Any other problems, for example, the specified path does not exist, or there is not enough disk space will result in the normal dialogs being displayed as if the backup was initiated from the menu.

CreateBaseline Method

Applies To **OPProject** object

Description Creates a new baseline in the project represented by the specified **OPProject** object.

Syntax *boolean= object*.**CreateBaseline**(*name, description, type, filter, progress, complete, useActuals*)

Remarks The **CreateBaseline** method syntax uses these parts:

Part	Description
boolean	A variable of type Boolean
object	An object variable of type OPProject
name	A string or variable of type String that specifies the unique name ofnew baseline. If more than eight characters are specified, only first eight characters are used to create the baseline name If the specified name is the name of a baseline that already exists in the project, the method will fail.
description	A string or variable of type String that specifies the description of the new baseline.
type	An integer or variable of type Integer that specifies the set of dates that will be used in the new baseline. Possible values for type are: 0 (Early Dates), 1 (Late Dates), and 2 (Scheduled Dates, only when the project has resource scheduled dates).
filter	A string or variable of type String that specifies the name of a filter used to limit the new baseline to a specific group of activities. If the name of the specified filter does not exist in the project, the method will fail.
progress	A Boolean value or variable of type Boolean that indicates whether in-progress activities are to be included in the new baseline. A value of FALSE will exclude in-progress activities from the new baseline.
complete	A Boolean value or variable of type Boolean that indicates whether completed activities are to be included in the new baseline. A value of FALSE will exclude completed activities from the new baseline.
useActuals	A Boolean value or variable of type Boolean that indicates whether progressed dates should be stored in the baseline

Part	Description
	for completed activities. A value of TRUE indicates that progress dates will be stored in the baseline for completed activities. A value of FALSE indicates that existing baseline dates will be stored in the baseline for completed activities. This parameter has no effect if completed is equal to FALSE.

The **CreateBaseline** method returns False if unsuccessful.

CreateBaselineWithOptionFlags Method

Applies To **OPProject** object

Description Creates a new baseline in the project represented by the specified **OPProject** object.

Syntax *boolean= object*.**CreateBaselineWithOptionFlags**(*name, description, type, filter, flags*)

Remarks The **CreateBaselineWithOptionFlags** method syntax uses these parts:

Part	Description
boolean	A variable of type Boolean
object	An object variable of type OPProject
name	A string or variable of type String that specifies the unique name of new baseline. If more than eight characters are specified, only first eight characters are used to create the baseline name If the specified name is the name of a baseline that already exists in the project, the method will fail.
description	A string or variable of type String that specifies the description of the new baseline.
Type	An integer or variable of type Integer that specifies the set of dates that will be used in the new baseline. Possible values for type are: 0 (Early Dates), 1 (Late Dates), and 2 (Scheduled Dates, only when the project has resource scheduled dates).
filter	A string or variable of type String that specifies the name of a filter used to limit the new baseline to a specific group of activities. If the name of the specified filter does not exist in the project, the method will fail.
Flags	An integer or variable of type Integer that specifies which of the OpenPlanBaselineOptionFlags will be used in creating the baseline.

The **CreateBaselineWithOptionFlags** method returns **False** if unsuccessful.

> **Attention:** For a list of flags supported by Open Plan for the CreateBaselineWithOptionFlags method, see Appendix A: Create/Update Baseline Option Flags.

CreateBrowserView Method

Applies To **OPCreateApplication33** object

Description Activates a browser window and displays the specified document.

Syntax *boolean = application*.**CreateBrowserView***(document, embedded, title, display)*

Remarks The **CreateBrowserView** method syntax uses these parts:

Part	Description

boolean	A variable of type Boolean indicating a success or failure of the operation.
application	An object variable of type OPCreateApplication33
document	A string or variable of type String representing the URL or pathname of the document to be displayed.
embedded	A variable of type Boolean indicating whether the document should be displayed in Open Plan Explorer's internal browser window. *Title* is ignored if *embedded* is False.
display	A variable of type Integer used to specify the browser window's state. A value of 0 is for a normal state. A value of 1 is used to maximize the current browser window. A value of 2 is used to tile all internal windows if the variable *embedded* is true.

Note: Your browser's security settings can affect the way scripts or ActiveX controls are handled in the Open Plan Explorer's internal browser window. These security settings can be controlled via the use of Open Plan's Advanced System Options **AllowActiveXInBrowserView** and **AllowScriptInBrowserView**. Refer to Open Plan's Help system for more information on these options.

Date Method

Applies To **OPCalendarRecord** object

Description Retrieves the specified **Date** object.

Syntax **Set** *dateobject* = *object*.**Date**(*date*)

Remarks The **Date** method syntax uses these parts:

Part	Description
dateobject	An object variable of type OPDate
date	A date or variable of type Date
calendar	An object variable of type OPCalendarRecord.

DeleteBaseline Method

Applies To **OPProject** object

Description When applied to **OPProject** object, deletes the specified baseline.

Syntax *boolean* = *object*.**DeleteBaseline**(*name, showdialog)*

Remarks The **DeleteBaseline method** uses these parts:

Part	Description
boolean	A variable of the type Boolean.
name	A string or variable of type String that specifies the name of the baseline name to be deleted. If the specified baseline name doesn't exist, the method fails.
showdialog	A Boolean value or variable of type Boolean indicating whether a confirmation dialog should be displayed.

The **DeleteBaseline** method returns False if unsuccessful.

Description Method

Applies To **OPView** object, **OPFCView** object

Description Returns the description of the view represented by the specified object.

Syntax *string* = *object*.**Description**

Remarks The **Description** method uses these parts:

Part	Description
object	An object variable of one of the types specified above
string	A string or variable of type String

DCMADriverAnalysis Method

Applies To The **OPProject** object

Description Performs DCMA Driver analysis on activities in the project.

Syntax *long* = *object*.**DCMADriverAnalysis** (*ActivityID, Number Of Months, Calendar Option, Target Date Options, Number of Paths, Field to Store Results*)

Remarks The **DCMADriverAnalysis** method syntax uses these parts:

Part	Description
ActivitID	Id of the target activity
Numbe of Months	Enter the duration in months (m). Range 6 to 36 months.
Calendar Option	Enter 1 for Activities use activity calendars or 2 for activities use project default calendar.
Target Date Option	Enter 1 for Ignore all target dates, 2 for Consider Start and Finish Not Earlier Than targets.
Number of Paths	1 to N paths to analyze
Field to Store Results	Typically, a User Character Field used to store the path number

If successful, the **DCMADriverAnalysis** method returns a value of **0**.

Disable Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Disables mouse and keyboard input to the window represented by the specified object.

Syntax *object*.**Disable**

Remarks The **Disable** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above.

Use the **Enable** method to reverse the effects of **Disable**. Do not apply the **Disable** method to the **OPCreateApplication33** object when the application window is maximized—you will be unable to close the application or switch to another window.

EnableUndo Method

Applies To **OPCreateApplication33** object

Description Turns the multiple level Undo functionality on or off.

Syntax lResult = EnableUndo (bEnable)

Remarks The **EnableUndo** method syntax uses these parts:

Part	Description
bEnable	A Boolean argument. Set to True by default. When set to False, disables the multiple-level Undo functionality.
lResult	The error result of type Long. The result is zero if the method succeeds.

Enable Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Enables mouse and keyboard input to the window represented by the specified object.

Syntax *object*.**Enable**

Remarks The **Enable** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above.

Use the **Enable** method to reverse the effects of **Disable**.

ExpandAll Method

Applies To **OPView** Object

Description Allows the user to initiate an “expand all” for the corresponding Open Plan view.

Syntax *bResult* = *OPView*.**ExpandAll***(bOpenExternals)*

Remarks The **ExpandAll** method syntax uses these parts:

Part	Description
bOpenExternals	A Boolean argument. When set to True, any externals that are not currently opened will be opened as a result of expanding the view (this setting has no effect on non-activity related views). The failure to open one or more of the externals during the expansion process does not constitute an error
bResult	The Boolean result of the method. The method will return True if the view is open; False if closed.

ExtraWorkDays Method

Applies To **OPCalendarRecord** object

Description Retrieves the **OPExtraWorkDays** collection.

Syntax **Set** *extraworkdays* = *calendarrecord*.**ExtraWorkDays**

Remarks The **ExtraWorkDays** method syntax uses these parts:

Part	Description
extraworkdays	An object variable of type OPExtraWorkDays
calendarrecord	An object variable of type OPCalendarRecord

Fields Method

Applies To The following objects and collections: **OPActivities**, **OPActivityResources**, **OPAssignments**, **OPAvailabilities, OPCalendar, OPCalendars, OPCode, OPCodes**, **OPResources**, **OPPredecessors**, **OPFCCalendars**, **OPFCCodes**, **OPFCProjects**, **OPFCResources**, **OPProjectResources**, **OPProjects**, **OPResource**, **OPStep**, and **OPSteps**.

Description Retrieves the **OPFields** collection from the specified object.

Syntax **Set** *fields* = *object.***Fields***(calcfields, UDFs)*

Remarks The **Fields** method syntax uses these parts:

Part	Description
fields	An object variable of type **OPFields**
object	An object variable of one of the types specified above
calcfields	A boolean value or variable of type **Boolean**, specifying whether the collection should included calculated fields
UDFs	A boolean value or variable of type **Boolean**, specifying whether the collection should included user-defined fields

FieldsEx Method

Applies To The following objects and collections: **OPActivities**, **OPActivityResources**, **OPAssignments**, **OPAvailabilities, OPCalendar, OPCalendars**, **OPCode**, **OPCodes**, **OPFCalendar**, **OPFCCalendars, OPFCCodes**, **OPFCProjects**, **OPFCResources**, **OPProjectResources**, **OPProjects, OPResource, OPResources**, **OPPredecessors**, **OPStep**, and **OPSteps**.

Description Retrieves the **OPFields** collection from the specified object.

Syntax **Set** *fields* = *object.***FieldsEx***(flags)*

Remarks The **FieldsEx** method syntax uses these parts:

Part	Description
fields	An object variable of type **OPFields**
object	An object variable of one of the types specified above
flags	An integer or variable of type integer that represents the field types of interest. The following field types are available: All_Fields = 32767, // (May not be combined with any other flag) User_Fields = 1, Foreign_Key_Fields = 2, Calculated_Fields = 4, First_Baseline_Fields = 8, Second_Baseline_Fields = 16, Third_Baseline_Fields = 32, Code_Fields = 64, Filter_Fields = 128, Global_Edit_Fields = 256 You may use the constant All_Fields or combine other constants to get the field types of interest (for example, FieldsEx(User_Fields + Calculated_Fields)).

FileCabinet Method

Applies To **OPCreateApplication33** object

Description Retrieves the **OPFileCabinet** object.

Syntax **Set** *filecabinet* = *application*.**FileCabinet**

Remarks The **FileCabinet** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
filecabinet	An object variable of type **OPFileCabinet**

FileDelete Method

Applies To **OPCreateApplication33** object

Description Deletes a "file" (database object) of the specified type with the specified name. Returns True if the operation is successful.

Access Type Get

Data Type Boolean

Syntax boolean = application.**FileDelete** filetype filename

Remarks The **FileDelete** method syntax uses these parts:

Part	Description
filetype	A string or variable of type **String** specifying the type of file to be deleted: Project, Resource, Calendar, Code, or View.
filename	A string or variable of type **String** specifying the name of the file to be deleted.

Filename Method

Applies To **OPIcon** object, **OPView** object

Description Returns the name of the file associated with the collection or object.

Access Type Get

Data Type String

Syntax *string* = *object*.**Filename**

Remarks The **Filename** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type String

FileNew Method

Applies To **OPCreateApplication33** object

Description Creates a new file of the specified type.

Syntax *application*.**FileNew** *filetype*

Remarks The **FileNew** method syntax uses these parts:

Part	Description
application	An object variable of one of the types listed above
filetype	A string or variable of type **String** specifying the type of file to be created: Project, Resource, Calendar, Code, or CODEP.

FileNewEx Method

Applies To **OPCreateApplication33** object

Description Creates a new "file" (database object) of the specified type. Returns a non-blank file name if the operation is successful.

Access Type Get

Data Type String

Syntax *string = application.**FileNewEx** filetype filename filedescription*

Remarks The **FileNewEx** method syntax uses these parts:

Part	Description
filetype	A string or variable of type **String** specifying the type of file to be created: Project, Resource, Calendar, Code, or CODEP..
filename	A string or variable of type **String** specifying the name of the file to be created. If the string is blank, a file name will be generated by Open Plan. If it is not blank, the name must be unique for objects of that type (i.e., you cannot overwrite an existing file).
filedescription	A string or variable of type **String** specifying the description for the file to be created.

FileOpen Method

Applies To **OPCreateApplication33** object

Description Opens the specified file.

Syntax *boolean = application.***FileOpen(***filename, filetype, filemode)*

Remarks The **FileOpen** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
filename	A string or variable of type **String** specifying the full path name of the file to be opened
filetype	A string or variable of type **String** specifying one of the following types of file to be opened: PROJECT, RESOURCE, CALENDAR, CODE
filemode	A string or variable of type **String** specifying one of the following access modes in which to open the specified file: Exclusive, Shared, or Read Only. If an empty string is specified, the file will be opened in its default mode.
boolean	A variable of type **Boolean**

The **FileOpen** method returns False if unsuccessful.

FileOpenEx Method

Applies To **OPCreateApplication33** object

Description Opens the specified file.

Syntax *boolean = application.***FileOpenEx(***filename, filetype, filemode,openallexternalsubprojects)*

Remarks The **FileOpen** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
filename	A string or variable of type **String** specifying the full path name of the file to be opened
filetype	A string or variable of type **String** specifying one of the following types of file to be opened: PROJECT, RESOURCE, CALENDAR, CODE
filemode	A string or variable of type **String** specifying one of the following access modes in which to open the specified file: Exclusive, Shared, or Read Only. If an empty string is specified, the file will be opened in its default mode.
openallexternalsub projects	A variable of type **Boolean**. If the variable is set to FALSE, FileOpenEx will not open any external subprojects. If the variable is set to TRUE, FileOpenEx will open all external subprojects and their external subprojects.

FilePrint Method

Applies To **OPCreateApplication33** object

Description Prints the current view.

Syntax *application.***FilePrint** *showdialog*

Remarks The **FilePrint** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
showdialog	A boolean value or variable of type **Boolean**, with ‘True” indicating that the Print dialog box is to be displayed

FilePrintPreview Method

Applies To **OPCreateApplication33** object

Description Displays the print preview of the current view.

Syntax *application.***FilePrintPreview**

Remarks The **FilePrintPreview** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**

FilePrintSetup Method

Applies To **OPCreateApplication33** object

Description Displays the Print Setup dialog box.

Syntax *application.***FilePrintSetup**

Remarks The **FilePrintSetup** method syntax uses these parts:

Part	Description

application	An object variable of type **OPCreateApplication33**

FileRename Method

Applies To **OPCreateApplication33** object

Description Renames the specified file.

Syntax *boolean* = *application*.**FileRename***(filetype, oldname, newname)*

Remarks The **FileRename** method syntax uses these parts:

Part	Description
boolean	A variable of type **Boolean** indicating a success or failure of the operation.
application	An object variable of type **OPCreateApplication33**
filetype	A string or variable of type **String** representing the type of file that is to be renamed: **Project**, **Code**, **Calendar**, **Resource**.
oldname	A string or variable of type **String** representing the name of the file that is to be renamed.
newname	A string or variable of type **String** representing the new name of the file.

Filters Method

Applies To **OPProject** object, **OPResource** and **OPProjectResources** collections

Description Retrieves the **OPFilters** collection object. When *index* is used as an argument to the **Filters** method, retrieves the **OPFilter** object mapped into that index. When *name* is used as an argument to the **Filters** method, retrieves the **OPFilter** object that matches the specified name.

Syntax
Set *filters* = *object*.**Filters**
Set *filter* = *object.* **Filters.Item(***index***)**
Set *filter* = *object.* **Filters.Item(***name***)**

Remarks The **Filters** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
filters	An object variable of type **OPFilters**
filter	An object variable of type **OPFilter**
index	An integer or variable of type **Integer** that uniquely identifies an OPFilter object within the OPFilters collection
name	A string or variable of type **String** representing the name of the OPFilter object to be retrieved

GeneralExport Method

Applies To **OPCreateApplication33** object, **OPProject** object

Description When applied to the **OPCreateApplication33** object, this method exports the information in the current project if there is one. When applied to the **OPProject** object, the method exports the information in the project referenced by the **OPProject** object.

Syntax *boolean* = *object*.**GeneralExport**(*specname, filename*)

Remarks The **GeneralExport** method syntax uses these parts:

Part	Description
object	An object variable of type **OPCreateApplication33** or **OPProject**
specname	A string or variable of type **String** representing the export specification
filename	A string or variable of type **String** specifying the filename (with complete path) of the file to contain the exported data
boolean	A variable of type **Boolean**

GeneralImport Method

Applies To **OPCreateApplication33** object, **OPProject** object

Description When applied to the **OPCreateApplication33** object, this method updates the current project if there is one and *bactive* is equal to TRUE. If there is no current project and *bactive* is equal to TRUE, the method fails and returns FALSE. If *bactive* is equal to FALSE, the method creates an untitled project containing the imported information. Note, however, that a parameter set in the import specification determines whether or not the imported data overwrites existing records.

When applied to the **OPProject** object, the method updates the project referenced by the **OPProject** object, and *bactive* has no effect on the operation.

Syntax *boolean* = object.**GeneralImport**(*specname, filename, bactive*)

Remarks The **GeneralImport** method syntax uses these parts:

Part	Description
object	An object variable of type **OPCreateApplication33** or **OPProject**
specname	A string or variable of type **String** representing the import specification
filename	A string or variable of type **String** specifying the filename (with complete path) of the file to be imported
bactive	A Boolean value or variable of type **Boolean** specifying whether the data being imported should be applied to the current open project
boolean	A variable of type **Boolean**

The **GeneralImport** method fails and returns FALSE if the specified import specification given by *specname* doesn't exist or an invalid file name is specified by *filename.* But it allows *specname* and/*or filename* to be blank and opens the appropriate dialog box. If there is no given file extension, it appends the extension specified in the import specification to the filename.

GenerateCrosstabDates Method

Applies To **OPProject** object, **OPResource** collection

Description This method generates an array of dates to use when requesting crosstab data.

Syntax *project*.**GenerateCrosstabDates** *granularity, frequency, startdate, enddate*

Remarks The **GenerateCrosstabDates** method syntax uses these parts:

Part	Description
project	An object variable of type **OPProject**

granularity	A string or variable of type **String** with on one of the following case-sensitive values: Hours, Days, Weeks, Months, Quarters, or Years.
frequency	An integer or variable of type **Integer** with a value between 1 and 255 (inclusive), which is a multiplier of the time unit specified by *granularity*. For example a frequency of 2 and a granularity of Weeks generates a date every two weeks.
startdate	A date or variable of type **Date** representing the first date in the array
enddate	A date or variable of type **Date** representing the last date in the array

Care must be taken when specifying *granularity* in Hours. If the range of dates between *startdate* and *enddate* is too great, it is possible that the amount of data generated will be too large to store in memory.

GetAll Method

Applies To **OPHolidays** collection, **OPShifts** collection

Description When applied to the **OPHolidays** collection, returns a comma-delimited string containing all of the holidays for calendar record referenced by the collection. When applied to the **OPShifts** collection, returns a comma-delimited string containing all of the shifts for the date referenced by the **OPShifts** collection.

Syntax *string* = *holidays*.**GetAll**(*formatstring*)
string = *shifts*.**GetAll**

Remarks The **GetAll** method syntax uses these parts:

Part	Description
holidays	An object variable of type **OPHolidays**
shifts	An object variable of type **OPShifts**
string	A variable of type **String**
formatstring	A string or variable of type **String** containing the definition of the date format in which the holidays are to be returned.

> **Attention:** Refer to the Date Formats table in this document for the parameters used to define date formats.

GetAvailabilityCrosstabData Method

Applies To **OPResourceRecord** object

Description This method fills a dynamically allocated floating point array with time-phased resource availability data. The definition of the data is controlled by the GetCrosstabDates or SetCrosstab dates methods.

Syntax *resourcerecord*.**GetAvailabilityCrosstab** *floatarray*

Remarks The **GetAvailabilityCrosstab** method syntax uses these parts:

Part	Description
resourcerecord	An object variable of **OPResourceRecord**
floatarray	A dynamically dimensioned floating point or string array

GetAvailabilityCrosstabDataInXML Method

Applies To **OPResourceRecord** object

Description This method gets an XML string with the time-phased availability data. The definition of the data is controlled by the **GetCrosstabDates** or **SetCrosstabDates** methods.

Syntax *string* = *resourcerecord*.**GetAvailabilityCrosstabInXML** *(calcValues, calcOption)*

Remarks The **GetAvailabilityCrosstabInXML** method syntax uses these parts:

Part	Description
resourcerecord	An object variable of type **OPResourceRecord**
string	A variable of type **String**
calcValues	A string or variable of type **String** representing the case-sensitive values: QTY (quantity), COST, or ESCCOST (escalated cost). The values can be concatenated by a character "|", e.g., QTY|COST.
calcOption	A string or variable of type **String** representing one of the following case-sensitive values: PERIOD, CUMULATIVE, or AVERAGE

GetCalculatedFieldString Method

Applies To **OPActivities** collection

Description Gets the expression of the specified calculated field belonging to the Activity table.

Syntax *expression* = *activities*.**GetCalculatedFieldString**(*calcfieldname*)

Remarks The **GetCalculatedFieldString** method uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
expression	A variable of type **String**
calcfieldname	A string or variable of type **String** representing the name of the calculatedfield expression to be retrieved.

This method is documented solely for OPP v1.2b compatibility. It has been superseded by the properties and methods belonging to the **OPCalculatedField** object.

GetCrosstabDates Method

Applies To **OPProject** object, **OPResource** Collection

Description This method fills a dynamically allocated variant array with the dates generated by the **GenerateCrosstabDates** method.

Syntax *project*.**GetCrosstabDates** *datearray*

Remarks The **GetCrosstabDates** method syntax uses these parts:

Part	Description
project	An object variable of type **OPProject**
datearray	A dynamically dimensioned variant array

When using this method, the array must first be declared as a variant variable, not an array. The ReDim statement is then used to declare the array's type, as in the following example:

```
Dim CrosstabDates As Variant
ReDim CrosstabDates(0) As Date
```

The methods that will use this date array, **GetEarnedValueCrosstabData** and **GetResourceCrosstabData**, do not retrieve any data for the final element in the date array.

GetCrosstabDatesInXML Method

Applies To **OPProject** object, **OPResource** object

Description This method gets a XML string with the dates generated by the **GenerateCrosstabDates** method.

Syntax *string* = *project*.**GetCrosstabDates**

Remarks The **GetCrosstabDatesInXML** method syntax uses these parts:

Part	Description
string	A variable of type **String** that contains XML data. For example, <Dates> <Date index="1">1/1/99</Date> <Date index="2">2/1/99</Date> <Date index="3">3/1/99</Date> <Date index="4">4/1/99</Date> </Dates>

GetCrosstabMinutesPerDefaultUnit Method

Applies To **OPResource** object

Description Returns the number of minutes in the default duration unit defined for the project to which the resource file represented by the specified object is assigned.

Syntax *integer* = *resource*.**GetCrosstabMinutesPerDefaultUnit**

Remarks The **GetCrosstabMinutesPerDefaultUnit** method syntax uses these parts:

Part	Description
resource	An object variable of type **OPResource**
integer	A variable of type **Integer**

GetCurrentFields Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAssignment**, **OPAvailability, OPCodeRecord**, **OPCost**, **OPPredecessor**, **OPProjectResource**, **OPResourceRecord**, **OPResourceUsage**, and **OPStep**.

Description Retrieves the values of a list of fields, which can include user-defined fields and calculated fields, from the data file represented by one of the object types listed above. The list of fields and their optional formats is defined by the **AssignCurrentFieldSet** method.

Syntax *string* = *object*.**GetCurrentFields**

Remarks The **GetCurrentFields** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above

string	A variable of type **String**, representing the queried data, with each value separated by the pipe symbol (|)

The **GetCurrentFields** method returns a zero-length string if unsuccessful.

GetEarnedValueCrosstabData Method

Applies To **OPAssignment** object, **OPActivityResource** object, **OPProjectResource** object

Description This method fills a dynamically allocated floating point array with the time-phased earned value data. The definition of the data is controlled by the **SetEarnedValueCrosstabOptions** and **GetCrosstabDates** or **SetCrosstabDates** methods.

Syntax *object*.**GetEarnedValueCrosstabData** *result, floatarray*

Remarks The **GetEarnedValueCrosstabData** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
result	A string or variable of type **String** that contains one of the following case-sensitive values representing the type of earned value data that is to be returned: BCWS, ACWP, BCWP or FORECAST. The forecast type (EARLY, LATE, or SCHEDULE) is defined by the SetEarnedValueCrosstabOptions method.
floatarray	A dynamically dimensioned floating point or string array

When using this method, the array must first be declared as a variant variable, not an array. The ReDim statement is then used to declare the array's type, as in the following example:

```
Dim EVCrosstabData As Variant
ReDim EVCrosstabData(0) As Single
```

The last element in this array will always be zero and should be discarded.

GetEarnedValueCrosstabDataInXML Method

Applies To **OPAssignment** object, **OPActivityResource** object, **OPProjectResource** object

Description This method gets a XML string with the time-phased earned value data. The definition of the data is controlled by the **GetCrosstabDates** or **SetCrosstabDates** methods.

Syntax *string = object.***GetEarnedValueCrosstabDataInXML***(result, calcValues, calcOption, calcForecast)*

Remarks The **GetEarnedValueCrosstabDataInXML** method syntax uses these parts:

Part	Description
string	A variable of type **String** that contains XML data. For example, <ResourceAssignment ID="R1"> <EarnedValue CalculationValue="Quantity" ValueType="BCWS" Index="1" DateValue="1/1/99">10</EarnedValue> </ResourceAssignment>
object	An object variable of one of the types specified above.
result	A string or variable of type **String** that contains the case-sensitive values representing the type of earned value data that is to be returned: BCWS, ACWP, BCWP, EARLY, LATE, SCHEDULE

	The values can be concatenated by a character "|", e.g., BCWS|BCWP.
calcValues	A string or variable of type **String** representing the case-sensitive values: QTY (quantity, COST, or ESCCOST (escalated cost). The values can be concatenated by a character "|", e.g., QTY|COST.
calcOption	A string or variable of type **String** representing one of the following case-sensitive values: PERIOD, CUMULATIVE.
calcForecast	A string or variable of type **String**. If the result string contains the EARLY, LATE, or SCHEDULE value, it should be set as FORECAST. Otherwise it should be set as an empty string.

For example,

```
result = "EARLY"
calcValues = "QTY|COST"
calcOption = "PERIOD"
calcForecast = "FORECAST"
strXML = OPActivityResource.GetEarnedValueCrosstabDataInXML (result,
calcValues, calcOption, calcForecast)
```

GetField Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAvailability**, **OPAssignment**, **OPCalendar**, **OPCalendarRecord**, **OPCode**, **OPCodeRecord**, **OPCost**, **OPIcon**, **OPPredecessor**, **OPProject**, **OPProjectResource**, **OPResource**, **OPResourceRecord**, **OPResourceUsage**, and **OPStep**.

Description Retrieves the value of any field, including user-defined fields and calculated fields, from the data file represented by one of the object types listed above.

Syntax *fieldvalue* = *object.***GetField** *fieldname*

Remarks The **GetField** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
fieldname	A string or variable of type **String** containing the name of the field to be queried
fieldvalue	A variable of the appropriate data type for the field queried

GetFieldInfoInXMLFormat Method

Applies To **OPFields** collection.

Description Retrieves the properties in XML format for each **OPField** object contained in the collection.

Syntax sFieldProperties = *object.*GetFieldInfoInXMLFormat*(bFullProperties)*

Remarks The **GetFieldInfoInXMLFormat** method syntax uses these parts:

Part	Description
object	An object variable of type **OPFields**
bFullProperties	A variable of type Boolean indicating whether the full property set or a more limited property set should be returned (see below)
sFieldProperties	A String or variable of type String that contains the field properties in a standard XML format

Example, with *bFullProperties* set to True:

```
<Fields>
        <Field>
                <Name>ACT_ID</Name>
                <UserName>Activity ID</UserName>
            <Type>HIER</Type>
            <DBType>STR</DBType>
                <FieldType>DBT</FieldType>
            <IsEditable>Yes</IsEditable>
            <Length>59</Length>
            <Scale>0</Scale>
        </Field>
</Fields>
```

Example, with *bFullProperties* set to False:

```
<Fields>
        <Field>
                <Name>ACT_ID</Name>
                <UserName>Activity ID</UserName>
            <Type>HIER</Type>
                <FieldType>DBT</FieldType>
        </Field>
</Fields>
```

Assuming that you were interested in getting a collection of fields on the activity table, you would first get the OAActivities collection. The following sample VB6 code shows how to access the method. OAFields::GetFieldInfoInXMLFormat returns an XML string. The Boolean parameter for the method indicates whether the caller wants “full” (True) or “abbreviated” (False) output. An explanation of the output follows the code sample.

```
bResult = myApp.FileOpen(“CLEAN”, “PROJECT”, “Exclusive”)
If bResult Then
        Set myProject = myApp.Projects(1)
        If Not myProject Is Nothing Then
                Set myActivities = myProject.Activities
                If Not myActivities Is Nothing Then
                        myFields = myActivities.Fields
                        If Not myFields Is Nothing Then
```

```
                                    sFields = myFields.GetFieldInfoInXMLFormat(True)
                                Set myFields = Nothing
                            End If
                        Set MyActivities = Nothing
                    End If
                    myProject.ShutWOSave
                Set MyProject = Nothing
            End If
End If
```

"Abbreviated" output gives you the following:

- Name (e.g. "ACT_ID")
- UserName (e.g. "Activity Id")
- Type (e.g., "HIER"). The following is a list of types:
  - UNKN (unknown)
  - TEXT
  - MEMO (e.g., a note field)
  - CODE
  - INTE (integer)
  - DBLE (double)
  - DATE (start date)
  - FDAT (finish date)
  - DURA (duration)
  - HIER, HIEI (hierarchies). The second type indicates specifically that the hierarchy is defined as case-insensitive, although in practice, OP treats all hierarchical data as case-insensitive.
  - GUID (e.g., "ACT_UID")
  - DSPC (dayspec)
  - SHFT (calendar shift)
  - Additionally, there are a number of enumerated types. I can provide a list of these if desired. One example is the relationship type field, whose database type is "RELT".

"Full" output gives you the above, plus:

- DBTYPE (database type). Values are the following:
  - EMPTY (used internally for the **results** of calculated fields—calculated field results are not stored in the database, so they don't really have a DBTYPE)
  - STR (string)

- I4 (integer)
- R8 (double)
- DATE
- TIMESTAMP
- IUNKNOWN

- FieldType. Values are the following:
  - CLC (calculated field)
  - UDT (user field of type date or finish date)
  - UNM (user field of type double)
  - UTX (user field of type text)
  - NTX (a note or memo field)
  - CRA (a code field)
  - UDU (user field of type duration)
  - UDF (user field that is something other than one of the above)
  - DBT (neither a calculated field nor a user field)
- IsEditable ("Yes" or "No")
- Length (maximum length of field—For non-text items, the length field may refer to the **display** length when the data is transformed into a string)
- Scale (number of decimal places)

GetFields Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAvailability**, **OPAssignment**, **OPCalendar**, **OPCalendarRecord**, **OPCode**, **OPCodeRecord**, **OPCost**, **OPIcon**, **OPPredecessor**, **OPProject**, **OPProjectResource**, **OPResource**, **OPResourceRecord**, **OPResourceUsage**, and **OPStep**.

Description Retrieves the values of a list of fields, which can include user-defined fields and calculated fields, from the data file represented by one of the object types listed above.

Syntax *valuelist* = *object.***GetFields** *fieldlist*

Remarks The **GetFields** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
fieldlist	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)
valuelist	A variable of type **String**

The **GetFields** method returns string representation of the queried data, with each value separated by the pipe symbol (|)

GetFilterString Method

Applies To **OPActivities** collection

Description Returns the definition of the specified filter.

Syntax *definition* = *activities*.**GetFilterString**(*filtername*)

Remarks The **GetFilterString** method syntax uses these parts

Part	Description
activities	An object variable of one of type **OPActivities**
fieldlist	A string or variable of type **String** containing the name of the filter definition to be retrieved
definition	A string or variable of type **String**

GetLastSecurityValidation Method

Applies To **OPCreateApplication33** object

Description This method gets the information about the last security validation.

Syntax *Boolean* = *application*.**GetLastSecurityValidation***(result, class, action, rights, message)*

Remarks The **GetLastSecurityValidation** method syntax uses these parts:

Part	Description
boolean	A variable of type **Boolean** indicating a success or failure of the operation
application	An object variable of type **OPCreateApplication33**
result	A variable of type **Boolean** indicating a success or failure of the security validation
class	A variable of type **String** indicating which automation class is involved
action	A variable of type **String** indicating which automation method is called
rights	A variable of type **String** containing which access rights are required for the automation method
message	A variable of type **String** containing the error message

GetPosition Method

Applies To **OPWebWindow**

Description Returns the size and position of the window represented by the specified **OPWebWindow** object.

Syntax *webwindow*.**GetPosition** *top, left, height, width*

Remarks The **GetPosition** method syntax uses these parts:

Part	Description
webwindow	An object variable of type **OPWebWindow**
top	A variable of type **Long**
left	A variable of type **Long**
height	A variable of type **Long**
width	A variable of type **Long**

The top position, left position, height, and width of the WebWindow is returned to the specified variables.

GetResourceCrosstabData Method

Applies To **OPAssignment** object, **OPActivityResource** object, **OPProjectResource** object

Description This method fills a dynamically allocated floating point array with the time-phased resource data. The definition of the data is controlled by the **SetResourceCrosstabOptions** method and **GetCrosstabDates** or **SetCrosstabDates** methods.

Syntax *object*.**GetResourceCrosstabData** *result, floatarray*

Remarks The **GetResourceCrosstabData** method syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
result	A string or variable of type **String** that contains one of the following case-sensitive values representing the type of earned value data that is to be returned: EARLY, LATE, SCHEDULE, BASELINE, ACTUAL, or AVAILABILITY.
floatarray	A dynamically dimensioned floating point or string array

When using this method, the array must first be declared as a variant variable, not an array. The ReDim statement is then used to declare the array's type, as in the following example:

```
Dim ResCrosstabData As Variant
ReDim ResCrosstabData(0) As Single
```

The last element in this array will always be zero and should be discarded.

GetResourceCrosstabDataInXML Method

Applies To **OPAssignment** object, **OPActivityResource** object, **OPProjectResource** object

Description This method gets a XML string with the time-phased resource data. The definition of the data is controlled by the **GetCrosstabDates** or **SetCrosstabDates** methods.

Syntax *string* = *object*.**GetResourceCrosstabDataInXML***(result, calcValues, calcOption)*

Remarks The GetResourceCrosstabDataInXML method syntax uses these parts:

Part	Description
string	A variable of type **String** that contains XML data. For example, <ResourceAssignment ID=”R1”> <ResourceValue CalculationValue=”Quantity” ValueType=”Early” Index=”1” DateValue=”1/1/99”>10</ResourceValue> </ResourceAssignment>
object	An object variable of one of the types specified above
result	A string or variable of type **String** that contains the case-sensitive values representing the type of earned value data that is to be returned: EARLY, LATE, SCHEDULE, BASELINE, ACTUAL, AVAILABILITY. The values can be concatenated by a character “|”, e.g., EARLY|LATE.
calcValues	A string or variable of type **String** representing the case-sensitive values: QTY (quantity), COST, or ESCCOST (escalated cost). The values can be concatenated by a character “|”, e.g., QTY|COST.

calcOption	A string or variable of type **String** representing one of the following case-sensitive values: PERIOD, CUMULATIVE, or AVERAGE

For example,

```
result = "EARLY|LATE"
calcValues = "QTY|COST"
calcOption = "PERIOD"
strXML = OPActivityResource.GetResourceCrosstabDataInXML(result, calcValues,
calcOption)
```

GetResourceDateArray Method

Applies To **OPAssignment** object**, OPActivityResource** object, **OPProjectResource** object

Description This method fills a dynamically allocated variant array with the dates corresponding to the periods of usage (early, late, schedule, baseline, or actual) or availability for a resource represented by the specified object.

Syntax *object*.**GetResourceDateArray** *result*, *datearray*

Remarks The **GetResourceDateArray** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
result	A string or variable of type **String** that contains one of the following case-sensitive values representing the type of dates that are to be returned: EARLY, LATE, SCHEDULE, BASELINE, ACTUAL, or AVAILABILITY.
datearray	A dynamically dimensioned variant array

When using this method, the array must first be declared as a variant variable, not an array. The ReDim statement is then used to declare the array's type, as in the following example:

```
Dim ResourceDates As Variant
ReDim ResourceDates(0) As Date
```

GetRights Method

Applies To **OPAccessControl** object

Description This method returns an integer error value.

Syntax *integer* = *object*.**GetRights** *(GroupID, UserID, RoleID, ReadOnly)*

Remarks The **GetRights** method syntax uses these parts:

Part	Description
object	An object variable of type **OPAccessControl**.
integer	A negative value indicates an error. Non-zero values are warnings or additional information. For example, the value OP_S_COLUMN_VALUE_IS_SAME (324032) indicates that one or more of the fields being set was not changed from the original value(s).
GroupID	A string or variable of type **String** indicating the Group ID to be assigned to the associated access control record.
UserID	A string or variable of type **String** indicating the User ID to be assigned to the associated access control record.
RoleID	A string or variable of type **String** indicating the Role ID to be assigned to the associated access control record.

ReadOnly	A string or variable of type **String** is returned indicating whether the specified Group or User is to be assigned read-only access to the owner object (the owner object may be a project, calculated field, sort, etc.). The value will be either "Yes" or "No".

GetSelectedActivitiesArray Method

Applies To **OPProject** object

Description This method fills a dynamically allocated string array with the activity ID's of all activities selected in the topmost view if that view belongs to the project represented by the specified object. The method returns the number of items in the string array.

Syntax *integer* = *project*.**GetSelectedActivitiesArray**(*stringarray*)

Remarks The **GetSelectedActivitiesArray** method syntax uses these parts:

Part	Description
project	An object variable of type **OPProject**
integer	An integer or variable of type **Integer**
stringarray	A dynamically dimensioned string array

When using this method, the array must first be declared as a variant variable, not an array. The ReDim statement is then used to declare the array's type, as in the following example:

```
Dim SelectedActivities As Variant
ReDim SelectedActivities(0) As String
```

GetSharedModeLock Method

Applies To **OPProject** object

Description This method gets the shared mode lock state for the specified **OPProject** object.

Syntax *long*= *object*.**GetSharedModeLock** (*islockset*)

Remarks The **GetSharedModeLock** method syntax uses these parts:

Part	Description
long	Error code where a zero value indicates success. A negative value indicates an error. Non-zero positive values are warnings or additional information.
islockset	A pointer to a Boolean value indicating the current state of the shared mode lock for the project. A value of TRUE indicates that the project currently holds an active shared mode lock, while FALSE indicates that it does not.

GetSkills Method

Applies To **OPResourceRecord** object

Description This method generates a string array containing the names (IDs) of the skills assigned to the resource represented by the specified **OPResourceRecord** object.

Syntax *resourcerecord*.**GetSkills** *skills*

Remarks The **GetSkills** method syntax uses these parts:

Part	Description
resourcerecord	An object variable of type **OPResourceRecord**

skills	A dynamically dimensioned string array

GetStandardDays Method

Applies To **OPCalendarRecord** collection

Description Returns a seven-character string representing the state of the Work flag for each day of the week in the specified calendar, where the first character in the string represents Sunday. A non-working day is represented by 0 and a working day is represented by 1.

Syntax *string* = *calendarrecord*.**GetStandardDays**

Remarks The **GetStandardDays** method syntax uses these parts:

Part	Description
calendarrecord	An object variable of type **OPCalendarRecord**
string	A variable of type **String**

GlobalEdits Method

Applies To **OPProject** object, **OPResource** and **OPProjectResources** collections

Description Retrieves the **OPGlobalEdits** collection object. When *index* is used as an argument to the **GlobalEdits** method, retrieves the **OPGlobalEdit** object mapped into that index. When *name* is used as an argument to the **GlobalEdits** method, retrieves the **OPGlobalEdit** object that matches the specified name.

Syntax
Set *globaledits* = *object*.**GlobalEdits**
Set *globaledit* = *object.* **GlobalEdits.Item(***index***)**
Set *globaledit* = *object.* **GlobalEdits.Item(***name***)**

Remarks The **GlobalEdits** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above
globaledits	An object variable of type **OPGlobalEdits**
globaledit	An object variable of type **OPGlobalEdit**
index	An integer or variable of type Integer that uniquely identifies an OPGlobalEdit object within the OPGlobalEdits collection
name	A string or variable of type **String** representing the name of the OPGlobalEdit object to be retrieved

Graphs Method

Applies To **OPFileCabinet** object, **OPProject** object

Description Retrieves the **OPGraphs** or **OPFCGraphs** collection object. When *index* is used as an argument to the **Graphs** method, retrieves the **OPFCView** or **OPView** object mapped into that index. When *name* is used as an argument to the **Graphs** method, retrieves the **OPFCView** or **OPView** object that matches the specified name.

Syntax
Set *graphs* = *object*.**Graphs**
Set *graph* = *object*.**Graphs.Item(***index***)**
Set *graph* = *object*.**Graphs.Item(***name***)**

Remarks The **Graphs** method syntax uses these parts:

Part	Description
Object	An object variable of type **OPFileCabinet** or **OPProject**
graphs	An object variable of type **OPFCGraphs** or **OPGraphs**
graph	An object variable of type **OPFCView** or **OPView**
index	An integer or variable of type Integer that uniquely identifies an OPFCView or OPView object within the OPFCGraphs or OPGraphs collection
name	A string or variable of type **String** representing the identifier for the requested OPFCView or OPView object

Holidays Method

Applies To **OPCalendarRecord** object

Description Retrieves the **OPHolidays** collection.

Syntax **Set** *holidays* = *calendarrecord*.**Holidays**

Remarks The **Holidays** method syntax uses these parts:

Part	Description
Holidays	An object variable of type **OPHolidays**
calendarrecord	An object variable of type **OPCalendarRecord**

Item Method

Applies To All collection objects in the Open Plan object hierarchy

Description Retrieves an object from the specified collection.

Syntax The general syntax for the **Item** method is as follows:

Set *object* = *collection*.**Item**(*index)*

Set *object* = *collection*.**Item**(*uniqueidentifier)*

Remarks When *index* is used as an argument to the **Item** method, the object mapped into that index is retrieved from the collection. All of the Open Plan collection objects listed above support the *index* argument. Some collection objects may also use a unique identifier as an argument. When this identifier is used as an argument to the **Item** method, the object whose unique identifier matches the argument is retrieved from the collection.

The syntax of the **Item** method using the unique identifier argument varies depending on the collection to which the method has been applied. The syntax for each of these collections is documented separately below.

OPActivities collection

Syntax Set *activity* = *activities*.**Item**(*activityid)*

Remarks When applied to the **OPActivities** collection, the **Item** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
activity	An object variable of type **OPActivity**
activityid	A string or variable of type **String** representing the unique identifier of an activity

OPBaselines collection

Syntax Set *activity* = *baselines*.**Item**(*activityid)*

Remarks When applied to the **OPBaselines** collection, the **Item** method syntax uses these parts:

Part	Description
baselines	An object variable of type **OPBaselines**
activity	An object variable of type **OPActivity**
activityid	A string or variable of type **String** representing the unique identifier of an activity

OPBaselinesList collection

Syntax Set *baselines* = *baselineslist*.**Item**(*baselinename*)

Remarks When applied to the **OPBaselinesList** collection, the **Item** method syntax uses these parts:

Part	Description
baselineslist	An object variable of type **OPBaselinesList**
baselines	An object variable of type **OPBaselines**
baselinename	A string or variable of type **String** representing the unique identifier of a baseline

OPCalculatedFields collection

Syntax **Set** *calcfield* = *calculatedfields*.**Item(***calcfieldname*)

Remarks When applied to the **OPCalculatedFields** collection, the **Item** method syntax uses these parts:

Part	Description
calcfield	An object variable of type **OPCalculatedField**
calculatedfields	An object variable of type **OPCalculatedFields**
calcfieldname	A string or variable of type **String** representing the name of a calculated field

OPCalendar collection

Syntax **Set** *calendarrecord* = *calendar*.**Item(***calendarrecordname*)

Remarks When applied to the **OPCalendar** collection, the **Item** method syntax uses these parts:

Part	Description
calendar	An object variable of type **OPCalendar**
calendarrecord	An object variable of type **OPCalendarRecord**
calendarrecordname	A string or variable of type **String** representing the name of a calendar within a calendar file

OPCalendars collection

Syntax **Set** *calendar* = *calendars*.**Item(***calendarfilename*)

Remarks When applied to the **OPCalendars** collection, the **Item** method syntax uses these parts:

Part	Description
calendars	An object variable of type **OPCalendars**
calendar	An object variable of type **OPCalendar**
calendarfilename	A string or variable of type **String** representing the fully qualified path name of a calendar file

OPCategories collection

Syntax **Set** *category* = *categories*.**Item(***categoryname*)

Remarks When applied to the **OPCategories** collection, the **Item** method syntax uses these parts:

Part	Description
category	An object variable of type **OPCategory**
categories	An object variable of type **OPCategories**
categoryname	A string or variable of type **String** representing the name of a category

OPCode collection

Syntax **Set** *coderecord* = *code*.**Item**(*coderecordcode*)

Remarks When applied to the **OPCode** collection, the **Item** method syntax uses these parts:

Part	Description
code	An object variable of type **OPCode**
coderecord	An object variable of type **OPCodeRecord**
coderecordcode	A string or variable of type **String** representing the unique identifier of a code record object

OPCodes collection

Syntax **Set** *code* = *codes*.**Item(***codefilename*)

Remarks When applied to the **OPCodes** collection, the **Item** method syntax uses these parts:

Part	Description
codes	An object variable of type **OPCodes**
code	An object variable of type **OPCode**
codefilename	A string or variable of type **String** representing the fully qualified path name of a code file

OPExtraWorkDays collection

Syntax **Set** *date* = *extraworkdays*.**Item**(*date*)

Remarks When applied to the **OPExtraWorkDays** collection, the **Item** method syntax uses these parts:

Part	Description
extraworkdays	An object variable of type **OPExtraWorkDays**
dateobject	An object variable of type **OPDate**
date	A date or variable of type **Date**

OPFCBarcharts collection

Syntax **Set** *view* = *barcharts*.**Item(***viewdescription*)

Remarks When applied to the **OPFCBarcharts** collection, the **Item** method syntax uses these parts:

Part	Description
barcharts	An object variable of type **OPFCBarcharts**
view	An object variable of type **OPFCView**
viewdescription	A string or variable of type **String** representing the descriptive text appearing under the icon represented by the specified object

OPFCCalendars collection

Syntax **Set** *calendar* = *calendars*.**Item(***calendarfilename*)

Remarks When applied to the **OPFCCalendars** collection, the **Item** method syntax uses these parts:

Part	Description
calendars	An object variable of type **OPFCCalendars**
icon	An object variable of type **OPIcon**
calendarfilename	A string or variable of type **String** representing the fully qualified path name of a calendar file

OPFCCodes collection

Syntax **Set** *code* = *codes*.**Item(***codefilename*)

Remarks When applied to the **OPFCCodes** collection, the **Item** method syntax uses these parts:

Part	Description
codes	An object variable of type **OPFCCodes**
icon	An object variable of type **OPIcon**
codefilename	A string or variable of type **String** representing the fully qualified path name of a code file

OPFCGraphs collection

Syntax **Set** *view* = *graphs*.**Item(***viewdescription*)

Remarks When applied to the **OPFCGraphs** collection, the **Item** method syntax uses these parts:

Part	Description
graphs	An object variable of type **OPFCGraphs**
view	An object variable of type **OPFCView**
viewdescription	A string or variable of type **String** representing the descriptive text appearing under the icon represented by the specified object

OPFCNetworks collection

Syntax **Set** *view* = *networks*.**Item(***viewdescription*)

Remarks When applied to the **OPFCNetworks** collection, the **Item** method syntax uses these parts:

Part	Description
networks	An object variable of type **OPFCNetworks**
view	An object variable of type **OPFCView**
viewdescription	A string or variable of type **String** representing the descriptive text appearing under the icon represented by the specified object

OPFCProjects collection

Syntax **Set** *project* = *projects*.**Item(***projectfilename*)

Remarks When applied to the **OPFCProjects** collection, the **Item** method syntax uses these parts:

Part	Description
projects	An object variable of type **OPFCProjects**
icon	An object variable of type **OPIcon**
projectfilename	A string or variable of type **String** representing the fully qualified path name of a project file

OPFCResources collection

Syntax **Set** *resource* = *resources*.**Item**(*resourcefilename*)

Remarks When applied to the **OPFCResources** collection, the **Item** method syntax uses these parts:

Part	Description
resources	An object variable of type **OPFCResources**
icon	An object variable of type **OPIcon**
resourcefilename	A string or variable of type **String** representing the fully qualified path name of a resource file

OPResourceUsages collection

Syntax **Set** *resourceusage* = *resourceusages*.**Item**(*resourceusageid*)

Remarks When applied to the **OPResourcesUsages** collection, the **Item** method syntax uses these parts:

Part	Description
resourceusages	An object variable of type **OPResourceUsages**
resourceusage	An object variable of type **OPResourceUsage**
resourceusageid	A string or variable of type **String** representing the name (ID) of a defined resource usage or an integer or variable of type **Integer** representing the index of the resource usage in the parent array

OPFCSpreadsheets collection

Syntax **Set** *view* = *spreadsheets*.**Item(***viewdescription*)

Remarks When applied to the **OPFCSpreadsheets** collection, the **Item** method syntax uses these parts:

Part	Description
spreadsheets	An object variable of type **OPFCSpreadsheets**
view	An object variable of type **OPFCView**
viewdescription	A string or variable of type **String** representing the descriptive text appearing under the icon represented by the specified object

OPFCViews collection

Syntax **Set** *view* = *views*.**Item(***viewdescription*)

Remarks When applied to the **OPFCViews** collection, the **Item** method syntax uses these parts:

Part	Description
views	An object variable of type **OPFCViews**
view	An object variable of type **OPFCView**
viewdescription	A string or variable of type **String** representing the descriptive text appearing under the icon represented by the specified object

OPFields collection

Syntax Set *field* = *fields*.**Item**(*fieldname*)

Remarks When applied to the **OPFields** collection, the **Item** method syntax uses these parts:

Part	Description
fields	An object variable of type **OPFields**

field	An object variable of type **OPField**
fieldname	A string or variable of type **String** representing the name of the field

OPFilters collection

Syntax **Set** *filter* = *filters*.**Item**(*filtername*)

Remarks When applied to the **OPFilters** collection, the **Item** method syntax uses these parts:

Part	Description
filters	An object variable of type **OPFilters**
filter	An object variable of type **OPFilter**
filtername	A string or variable of type **String** representing the name of a filter object

OPGlobalEditscollection

Syntax **Set** *globaledit* = *globaledits*.**Item**(*globaleditname*)

Remarks When applied to the **OPGlobalEdits** collection, the **Item** method syntax uses these parts:

Part	Description
globaledits	An object variable of type **OPGlobalEdits**
globaledit	An object variable of type **OPGlobalEdit**
globaleditname	A string or variable of type **String** representing the name of a global edit object

OPHolidays collection

Syntax **Set** *dateobject* = *holidays*.**Item**(*date*)

Remarks When applied to the **OPHolidays** collection, the **Item** method syntax uses these parts:

Part	Description
holidays	An object variable of type **OPHolidays**
dateobject	An object variable of type **Date**
date	An date or variable of type **Date**

OPNotes collection

Syntax **Set** *note* = *notes*.**Item**(*category*)

Remarks When applied to the **OPNotes** collection, the **Item** method syntax uses these parts:

Part	Description
notes	An object variable of type **OPNotes**
note	An object variable of type **OPNote**
category	A string or variable of type **String** representing the category to which the note object belongs

OPPredecessors collection

Syntax **Set** *predecessor* = *predecessors*.**Item**(*activityid)*

Remarks When applied to the **OPPredecessors** collection, the **Item** method syntax uses these parts:

Part	Description
predecessors	An object variable of type **OPPredecessors**
predecessor	An object variable of type **OPPredecessor**

activityid	A string or variable of type **String** representing the unique identifier of an activity

OPProjectCode collection

Syntax **Set** *coderecord* = *projectcode*.**Item**(*coderecordcode*)

Remarks When applied to the **OPProjectCode** collection, the **Item** method syntax uses these parts:

Part	Description
projectcode	An object variable of type **OPProjectCode**
coderecord	An object variable of type **OPCodeRecord**
coderecordcode	A string or variable of type **String** representing the unique identifier of a code record object

OPProjectCodes collection

Syntax **Set** *projectcode* = *projectcodes*.**Item**(*codefilename*)

Remarks When applied to the **OPProjectCodes** collection, the **Item** method syntax uses these parts:

Part	Description
projectcodes	An object variable of type **OPProjectCodes**
projectcode	An object variable of type **OPProjectCode**
codefilename	A string or variable of type **String** representing the fully qualified path name of a code file

OPProjectResources collection

Syntax **Set** *projectresource* = *projectresources*.**Item**(*resourceid*)

Remarks When applied to the **OPProjectResources** collection, the **Item** method syntax uses these parts:

Part	Description
projectresources	An object variable of type **OPProjectResources**
projectresource	An object variable of type **OPProjectResource**
resourceid	A string or variable of type **String** representing the unique identifier of a resource record object

OPProjects collection

Syntax **Set** *project* = *projects*.**Item(***projectfilename*)

Remarks When applied to the **OPProjects** collection, the **Item** method syntax uses these parts:

Part	Description
projects	An object variable of type **OPProjects**
project	An object variable of type **OPProject**
projectfilename	A string or variable of type **String** representing the fully qualified path name of a project file

OPResource collection

Syntax **Set** *resourcerecord* = *resource*.**Item**(*resourceid*)

Remarks When applied to the **OPResource** collection, the **Item** method syntax uses these parts:

Part	Description
resource	An object variable of type **OPResource**

resourcerecord	An object variable of type **OPResourceRecord**
resourceid	A string or variable of type **String** representing the unique identifier of a resource record object

OPResources collection

Syntax **Set** *resource* = *resources*.**Item**(*resourcefilename*)

Remarks When applied to the **OPResources** collection, the **Item** method syntax uses these parts:

Part	Description
resources	An object variable of type **OPResources**
resource	An object variable of type **OPResource**
resourcefilename	A string or variable of type **String** representing the fully qualified path name of a resource file

OPSkills collection

Syntax **Set** *skill* = *skills*.**Item**(*skillid*)

Remarks When applied to the **OPSkills** collection, the **Item** method syntax uses these parts:

Part	Description
skills	An object variable of type **OPSkills**
skill	An object variable of type **OPSkill**
skillid	A string or variable of type **String** representing the name (ID) of a defined skill

OPSorts collection

Syntax **Set** *sort* = *sorts*.**Item**(*sortname*)

Remarks When applied to the **OPSorts** collection, the **Item** method syntax uses these parts:

Part	Description
sorts	An object variable of type **OPSorts**
sort	An object variable of type **OPSort**
sortname	A string or variable of type **String** representing the name of a sort object

OPSteps collection

Syntax **Set** *step* = *steps*.**Item**(*stepID* or *stepIndex)*

Remarks When applied to the **OPSteps** collection, the **Item** method syntax uses these parts:

Part	Description
step	An object variable of type OPStep
stepID or stepIndex	Either the unique name of the step among steps assigned to a certain activity or the order (an integer starting from 1) of the step in the list of steps for the given activity.

OPWebWindows collection

Syntax Set *webwindow* = *webwindows*.**Item**(*title*)

Remarks When applied to the **OPWebWindows** collection, the **Item** method syntax uses these parts:

Part	Description
webwindows	An object variable of type **OPWebWindows**
webwindow	An object variable of type **OPWebWindow**

title	A string or variable of type **String** representing the title bar caption of an open WebWindow

IsExternalOpen Method

Applies To **OPActivity** object

Description Returns **True** if a specified activity is associated with an open external subproject.

Syntax *external_is_open = activity.IsExternalOpen*

Remarks The **IsExternalOpen** method syntax uses these parts:

Part	Description
activity	An object variable of one of type OPActivity
Boolean	A variable of type Boolean indicating whether an external subproject associated with the activity (if applicable) is open (True) or closed (False)

Login Method

Applies To **OPCreateApplication33** object

Description Logs into Open Plan when security is enabled.

Syntax *success* = *application*.**Login**(*loginname*, *password*)

Remarks The **Login** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
loginname	A string or variable of type **String** specifying the user's login name
password	A string or variable of type **String** specifying the user's password
success	A variable of type **Boolean**, where "True" means the Login method was not successful.

Native Authentication:

When native authentication is enabled, you may omit the loginname parameter. When the loginname parameter is omitted, the Open Plan Login dialog will be displayed.

Delegated Authentication:

1. You must pass an empty loginname and password to login using delegated authentication.
2. If you pass a non-empty loginname and password, the login will be performed using native authentication.

> **Attention:** For more information about Native and Delegated authentication, refer to the help in the Deltek EPM Security Administrator application.

LoginWithErrorResult() Method

Applies To OPApplication object

Description Unlike the standard Login() method, which returns True if the login succeeds, and False if the login fails, this method returns a numeric error, which in some cases might provide better information about what went wrong. In general, an error condition is indicated by a return value of < 0.

Syntax error = application. LoginWithErrorResult(loginname, password)

Remarks The LoginWithErrorResult() method syntax uses these parts:

Part	Description
application	An object variable of type OPCreateApplication33
login name	A string or variable of type String specifying the user's login name
password	A string or variable of type String specifying the user's password
error	A variable of type Long , where 0 means the login was successful

The LoginWithErrorResult() method returns 0 if successful.

Maximize Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Activates and maximizes the window represented by the specified object.

Syntax *object*.**Maximize**

Remarks The **Maximize** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above

Do not apply the **Maximize** method to the **OPCreateApplication33** object when the keyboard and mouse input to the application window has been disabled—you will be unable to close the application or switch to another window.

Minimize Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Activates the window represented by the specified object and displays it as an icon.

Syntax *object*.**Minimize**

Remarks The **Minimize** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above

Use the **Maximize** or **Restore** methods to reverse the effects of the **Minimize** method.

Name Method

Applies To The following objects: **OPCalendarRecord**, **OPFCView**, **OPIcon**, and **OPView.**

Description For the **OPFCView**, **OPIcon**, and **OPView** objects. returns the name of the file represented by the specified object. For the **OPCalendarRecord** object, returns the name assigned to the calendar record represented by the specified object when it was created.

Syntax *string* = *object*.**Name**

Remarks The **Name** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
string	A variable of type **String**

Networks Method

Applies To **OPFileCabinet** object, **OPProject** object

Description Retrieves the **OPNetworks** or **OPFCNetworks** collection object. When *index* is used as an argument to the **Networks** method, retrieves the **OPFCView** or **OPView** object mapped into that index. When *name* is used as an argument to the **Networks** method, retrieves the **OPFCView** or **OPView** object that matches the specified name.

Syntax **Set** *networks* = *object*.**Networks**
Set *network* = *object*.**Networks.Item(***index***)**
Set *network* = *object*.**Networks.Item(***name***)**

Remarks The **Networks** method syntax uses these parts:

Part	Description
object	An object variable of type **OPFileCabinet** or **OPProject**
networks	An object variable of type **OPFCNetworks** or **OPNetworks**
network	An object variable of type **OPFCView** or **OPView**
index	An integer or variable of type Integer that uniquely identifies an OPFCView or OPView object within the OPFCNetworks or OPNetworks collection
name	A string or variable of type **String** representing the identifier for the requested OPFCView or OPView object

Notes Method

Applies To **OPActivity**, **OPActivityResource, OPCodeRecord**, **OPProject**, **OPResourceRecord**, or **OPProjectResource** objects

Description Retrieves the **OPNotes** collection object. When *index* is used as an argument to the **Notes** method, retrieves the **OPNote** object mapped into that index. When *category* is used as an argument to the **Notes** method, retrieves the **OPNote** object that matches the specified category.

Syntax **Set** *notes* = *object*.**Notes**
Set *note* = *object*.**Notes.Item(***index***)**
Set *note* = *object*.**Notes.Item(***category***)**

Remarks The **Notes** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above

notes	An object variable of type **OPNotes**
note	An object variable of type **OPNote**
index	An integer or variable of type **Integer** that uniquely identifies an OPNote object within the OPNotes collection
category	A string or variable of type **String** representing the category of the OPNote object to be retrieved

PageSetup Method

Applies To **OPApplication** object

Description Sets the page orientation and optionally the paper size used during printing for the current Open Plan session.

Syntax *bResult* = *OPApp*.**PageSetup***(sSize, bLandscape)*

Remarks The **PageSetup** method syntax uses these parts:

Part	Description
sSize	A string representing the paper size supported by your printer.
bLandscape	A variable of type Boolean. Set to True for Landscape; False for Portrait.
bResult	The Boolean result of the method.

The PageSetup method returns True id successful.

Predecessors Method

Applies To **OPActivity** object

Description Retrieves the **OPPredecessors** collection object. When *index* is used as an argument to the **Predecessors** method, retrieves the **OPPredecessor** object mapped into that index.

Syntax **Set** *predecessors* = *activity*.**Predecessors**
Set *predecessor* = *activity*.**Predecessors.Item**(*index*)

Remarks The **Predecessors** method uses these parts:

Part	Description
predecessors	An object variable of type **OPPredecessors**
predecessor	An object variable of type **OPPredecessor**
activity	An object variable of type **OPActivity**
index	An integer or variable of type **Integer** that uniquely identifies an OPPredecessor object within the OPPredecessors collection

PrintToFile Method

Applies To **OPApplication** object

Description Providing the current printer driver supports printing to a file, this method will ask the driver to create a file with the user-supplied name. This feature is primarily designed for creating .pdf (Adobe Acrobat) files.

Syntax application.**PrintToFile** filename

Remarks The **PrintToFile** method uses this part:

Part	Description

filename	A string or variable of type **String** specifying the name of a print file to be created. The file should not currently exist, but its path must be valid.

Projects Method

Applies To **OPFileCabinet** object, **OPCreateApplication33** object

Description Retrieves the **OPFProjects** or **OPProjects** collection object. When *index* is used as an argument to the **Projects** method, retrieves the **OPIcon** or **OPProject** object mapped into that index. When *name* is used as an argument to the **Projects** method, retrieves the **OPIcon** or **OPProject** object that matches the specified name.

Syntax

Set *projects* = *object*.**Projects**
Set *project* = *object*.**Projects.Item(***index***)**
Set *project* = *object*.**Projects.Item(***filename***)**

Remarks The **Projects** method syntax uses these parts:

Part	Description
object	An object variable of type OPFileCabinet or **OPCreateApplication33**
projects	An object variable of type **OPFCProjects** or **OPProjects**
project	An object variable of type **OPIcon** or **OPProject**
index	An integer or variable of type Integer that uniquely identifies an OPIcon object within the OPFCProjects collection or an OPProject object within the OPProjects collection
name	A string or variable of type **String** representing the name of the OPIcon or OPProject object to be retrieved

Refresh Method

Applies To **OPWebWindow** object

Description Refreshes the window represented by the specified object.

Syntax *webwindow*.**Refresh**

Remarks The **Refresh** method syntax uses these parts:

Part	Description
webwindow	An object variable of type **OPWebWindow**

Remove Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAssignment**, **OPAvailability**, **OPCodeRecord**, **OPCost**, **OPPredecessor, OPProjectResource**, **OPReportingCalendar**, **OPResourceRecord**, **OPResourceUsage,** and **OPStep**.

The following collections: **OPActivities**, **OPAssignments**, **OPAvailabilities**, **OPCalculatedFields**, **OPCalendar**, **OPCategories**, **OPCode**, **OPCosts**, **OPExtraWorkDays**, **OPFilters**, **OPHolidays**, **OPNotes**, **OPPredecessors**,

OPProjectCode, **OPProjectCodes**, **OPGlobalEdits, OPResource**, **OPResourceUsages**, **OPShifts**, **OPSkills**, **OPSteps**, and **OPSorts**

Description When applied to an object, the **Remove** method removes the specified object from its parent collection. When applied to collection, the **Remove** method removes the specified member object from that collection.

Syntax The general syntax for the **Remove** method is as follows:

object.Remove

boolean = *collection*.Remove(*index*)

boolean = *collection*.Remove(*uniqueidentifier*)

Remarks No arguments to the **Remove** method are required when it is applied to an object that is not a collection. Arguments that identify the object to be removed are required when applying the **Remove** method to a collection object. When *index* is used as an argument to the **Remove** method, the object mapped into that index is removed from the collection. All of the Open Plan collection objects listed above support the *index* argument. Some collection objects may also use a unique identifier as an argument. When this identifier is used as an argument to the **Remove** method, the object whose unique identifier matches the argument is removed from the collection.

The syntax of the **Remove** method using the unique identifier argument varies depending on the collection to which the method has been applied. The syntax for each of these collections is documented separately below.

When applied to collections, the **Remove** method returns a Boolean value indicating if the object was removed.

OPActivities collection

Syntax *boolean* = *activities*.**Remove**(*activityid)*

Remarks When applied to the **OPActivities** collection, the **Remove** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
activityid	A string or variable of type **String** representing the unique identifier of an activity object
boolean	A variable of type **Boolean**

OPCalculatedFields collection

Syntax *boolean* = *calcfields*.**Remove**(*calcfield*)

Remarks When applied to the **OPCalculatedFields** collection, the **Remove** method syntax uses these parts:

Part	Description
calcfields	An object variable of type **OPCalculatedFields**
calcfield	A string or variable of type **String** representing the unique identifier of a calculated field object
boolean	A variable of type **Boolean**

OPCalendar collection

Syntax *boolean* = *calendar*.**Remove(***calendarrecordname*)

Remarks When applied to the **OPCalendar** collection, the **Remove** method syntax uses these parts:

Part	Description
calendar	An object variable of type **OPCalendar**
calendarrecordname	A string or variable of type **String** representing the name of a calendar within a calendar file
boolean	A variable of type **Boolean**

OPCategories collection

Syntax *boolean =categories*.**Remove(***categoryname*)

Remarks When applied to the **OPCategories** collection, the **Remove** method syntax uses these parts:

Part	Description
categories	An object variable of type **OPCategories**
categoryname	A string or variable of type **String** representing the name of a category
boolean	A variable of type **Boolean**

OPCode collection

Syntax *boolean =code*.**Remove(***coderecordcode*)

Remarks When applied to the **OPCode** collection, the **Remove** method syntax uses these parts:

Part	Description
code	An object variable of type **OPCode**
coderecordcode	A string or variable of type **String** representing the unique identifier of a code record object
boolean	A variable of type **Boolean**

OPExtraWorkDays collection

Syntax *boolean = extraworkdays*.**Remove**(*date*)

Remarks When applied to the **OPExtraWorkDays** collection, the **Remove** method syntax uses these parts:

Part	Description
extraworkdays	An object variable of type **OPExtraWorkDays**
date	A date or variable of type **OPDate** representing the date object that is to be removed from the collection
boolean	A variable of type **Boolean**

OPFilters collection

Syntax *boolean =filters*.**Remove(***filtername*)

Remarks When applied to the **OPFilters** collection, the **Remove** method syntax uses these parts:

Part	Description
filters	An object variable of type **OPFilters**
filtername	A string or variable of type **String** representing the name of a filter
boolean	A variable of type **Boolean**

OPGlobalEdits collection

Syntax *boolean =globaledits*.**Remove(***globaleditname*)

Remarks When applied to the **OPGlobalEdits** collection, the **Remove** method syntax uses these parts:

Part	Description
globaledits	An object variable of type **OPGlobalEdits**
globaleditname	A string or variable of type **String** representing the name of a global edit definition
boolean	A variable of type **Boolean**

OPHolidays collection

Syntax *boolean = holidays*.**Remove**(*date*)

Remarks When applied to the **OPHolidays** collection, the **Remove** method syntax uses these parts:

Part	Description
holidays	An object variable of type **OPHolidays**
date	A date or variable of type **OPDate** representing the date object that is to be removed from the collection
boolean	A variable of type **Boolean**

OPNotes collection

Syntax *boolean =notes*.**Remove**(*category*)

Remarks When applied to the **OPNotes** collection, the **Remove** method syntax uses these parts:

Part	Description
notes	An object variable of type **OPNotes**
category	A string or variable of type **String** representing the name of the category to which the note that is to be removed belongs
boolean	A variable of type **Boolean**

OPPredecessors collection

Syntax *boolean = predecessors*.**Remove**(*activityid)*

Remarks When applied to the **OPPredecessors** collection, the **Remove** method syntax uses these parts:

Part	Description
predecessors	An object variable of type **OPPredecessors**
activityid	A string or variable of type **String** representing the unique identifier of an activity object
boolean	A variable of type **Boolean**

OPProjectCode collection

Syntax *boolean = projectcode*.**Remove**(*coderecordcode*)

Remarks When applied to the **OPProjectCode** collection, the **Remove** method syntax uses these parts:

Part	Description
projectcode	An object variable of type **OPProjectCode**
coderecordcode	A string or variable of type **String** representing the unique identifier of a code record object

boolean	A variable of type **Boolean**

OPProjectCodes collection

Syntax *boolean* = *projectcodes*.**Remove**(*codefilename*)

Remarks When applied to the **OPProjectCodes** collection, the **Remove** method syntax uses these parts:

Part	Description
projectcodes	An object variable of type **OPProjectCodes**
codefilename	A string or variable of type **String** representing the fully qualified path name of a code file of the code file to be removed from the project
boolean	A variable of type **Boolean**

OPResource collection

Syntax *boolean* = *resource*.**Remove**(*resourceid*)

Remarks When applied to the **OPResource** collection, the **Remove** method syntax uses these parts:

Part	Description
resource	An object variable of type **OPResource**
resourceid	A string or variable of type **String** representing the unique identifier of a resource record object
boolean	A variable of type **Boolean**

OPSkills collection

Syntax *boolean* = *skills*.**Remove**(*skillid*)

Remarks When applied to the **OPSkills** collection, the **Remove** method syntax uses these parts:

Part	Description
skills	An object variable of type **OPSkills**
skillid	A string or variable of type **String** representing the name (ID) of a defined skill
boolean	A variable of type **Boolean**

OPSorts collection

Syntax *boolean* = *sorts*.**Remove**(*sortname*)

Remarks When applied to the **OPSorts** collection, the **Remove** method syntax uses these parts:

Part	Description
sorts	An object variable of type **Sorts**
sortname	A string or variable of type **String** representing the name of a sort
boolean	A variable of type **Boolean**

OPSteps collection

Syntax *Boolean = steps.Remove(stepID* or *stepIndex)*

Remarks When applied to the **OPSteps** collection, the **Remove** method syntax uses these parts:

Part	Description
step	An object variable of type OPStep

stepID or stepIndex	Either the unique name of the step among steps assigned to a certain activity or the order (an integer starting from 1) of the step in the list of steps for the given activity.

RemoveAll Method

Applies To **OPShifts** collection

Description Removes all shifts from the day referenced by the **OPShifts** collection.

Syntax *shifts*.**RemoveAll**

Remarks The **RemoveAll** method syntax uses these parts:

Part	Description
shifts	An object variable of type **OPShifts**

RemoveSkill Method

Applies To **OPResourceRecord** object

Description Removes a specific skill record from the specified **OPResourceRecord** object

Syntax *boolean* = *resourcerecord*.**RemoveSkill***(skillid)*

Remarks The **RemoveSkill** method syntax uses these parts:

Part	Description
resourcerecord	An object variable of type **OPResourceRecord**
Skillid	A string or variable of type **String** representing the name (ID) of a defined skill
boolean	A variable of type **Boolean**

ReportingCalendars Method

Applies To **OPCreateApplication33** object

Description Retrieves the **OPReportingCalendars** collection object.

Syntax **Set** *reportingcalendars* = *application*.**ReportingCalendars**

Remarks The **ReportingCalendars** method syntax uses these parts:

Part	Description
reportingcalendars	An object variable of type **OPReportingCalendars**
application	An object variable of type **OPCreateApplication33**

Resources Method

Applies To **OPCreateApplication33** object, **OPFileCabinet** object, **OPProject** object

Description When applied to the **OPCreateApplication33** object, retrieves the **OPResources** collection object. When applied to the **OPFileCabinet** object, retrieves the **OPFCResources** collection object. When applied to the **OPProject** object, retrieves the **OPProjectResources** collection object. When *index* is used as an argument to the **Resources** method, retrieves the **OPIcon, OPResource**, or **OPProjectResource** object mapped into that index. When *name* is used as an argument to the **Resources** method, retrieves the **OPIcon** or **OPResource** object that matches the specified name. When

resourceid is used as an argument to the **Resources** method, retrieves the **OPProjectResource** object that matches the specified name.

Syntax

Set *resources* = *object.***Resources**
Set *resource* = *object.***Resources.Item**(*index*)
Set *resource* = *object.***Resources.Item**(*name*)
Set *resource* = *project.***Resources.Item**(*resourceid*)

Remarks The **Resources** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
resources	An object variable of type **OPFCResources**, **OPResources**, or **OPProjectResources**, depending on the parent object
resource	An object variable of type **OPIcon**, **OPResource**, or **OPProjectResource**, depending on the parent object
index	An integer or variable of type Integer that uniquely identifies the specified object within its parent collection
resourceid	The name of the **OPProjectResource** object to be retrieved
filename	The name of the OPIcon or OPResource object to be retrieved

ResourceSchedule Method

Applies To **OPProject** object

Description Performs resource scheduling on the project represented by the specified object.

Syntax *project.***ResourceSchedule** *schedulemethod, showdialog*

Remarks The **ResourceSchedule** method syntax uses these parts:

Part	Description
project	An object variable of type **OPProject**
schedulemethod	A string or variable of type **String** representing one of the following values: Time Limited or Resource Limited
showdialog	A boolean value or variable of type **Boolean**, where “True” indicates that the Resource Scheduling Options dialog box is to be displayed

Restore Method

Applies To The following objects: **OPCreateApplication33**, **OPFileCabinet**, **OPProject**, **OPView,** and **OPWebWindow**. The following collections: **OPCalendar**, **OPCode**, **OPProjectCode**, and **OPResource**.

Description Activates the window represented by the specified object and displays it.

Syntax *object.***Restore**

Remarks The **Restore** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above

Use the **Restore** method to undo the effects of the **Conceal**, **Maximize,** or **Minimize** methods.

RestoreFromBackup Method

Applies To **OPCreateApplication33** object

Description Restore data from an Open Plan backup file.

Syntax *boolean* = *application*.**RestoreFromBackup** (*filename, flags, ignorestructuredifferences, createmissingfields, overwriteexistingdata*)

Remarks The **RestoreFromBackup** method syntax uses these parts:

Part	Description
filename	A string or variable of type String that contains the full path name of the backup file to be restored.
flags	One or more option flags that defines which data will be restored from the backup. The following flag values are supported and they can be added together to allow restoring just the desired data. RESTORE_ALL = 0 RESTORE_PROJECTS = 1 RESTORE_CALENDARS = 2 RESTORE_CODES = 4 RESTORE_RESOURCES = 8 RESTORE_VIEWS = 16 RESTORE_SYSTEMLIBRARY = 32
ignorestructurediffe rences	A Boolean value that instructs the restore process to continue or abort the restore process if the data in the backup has fields with a different structure from the fields in the data source you are restoring the backup into. A value of true will cause structure differences to be ignored. A value of false will abort the restore process id structure differences are detected. Any structure differences that are detected will be written to the restore log file.
createmissingfields	A Boolean value that instructs the restore process to automatically create user defined fields for any fields in the backup that do not exist in the data source that you are restoring the backup into. A value of true will allow missing fields to be created automatically, A value of false will prevent missing fields from being created Any missing fileds that are detected will be written to the restore log file.
overwriteexistingda ta	A Boolean value that instructs the restore process to overwrite any data that already exists in the data source that you are restoring the backup into. A value of true will allow existing data to be overwritten. If the value is false, existing data will not be overwritten.

RiskAnalyze Method

Applies To **OPProject** object

Description Perform risk analysis on the project.

Syntax *project*.**RiskAnalyze** *showdialog*

Remarks The **RiskAnalyze** method syntax uses these parts:

Part	Description
project	An object variable of type **OPProject**
showdialog	A boolean value or variable of type **Boolean**, where "True" indicates that the Risk Analysis Options dialog box is to be displayed

Rollups Method

Applies To **OPCode, OPProjectCode, OPProject, OProjectResources** and **OPResource** objects

Description Retrieves the **OPRollups** collection object. When *index* is used as an argument to the **Rollups** method, retrieves the **OPBatchGlobalEdit** object mapped into that index. When *name* is used as an argument to the **BatchGlobalEdits** method, retrieves the **OPBatchGlobalEdit** object that matches the specified name.

Syntax **Set** *rollups* = *object*. **Rollups**
Set *rollup* = *object*. **Rollups.Item**(*index*)
Set *rollup* = *object*. **Rollups.Item**(*name*)

Remarks The **GlobalEdits** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
rollups	An object variable of type **OPRollups**
rollup	An object variable of type **OPRollup**
index	An integer or variable of type **Integer** that uniquely identifies an OPRollup object to be retrieved
name	A string or variable of type **String** representing the unique name of the requested OPRollup object

Save Method

Applies To The following objects: **OPCalendar**, **OPCode**, **OPProject**, **OPProjectCode**, and **OPResource**

Description Saves the file represented by the specified object.

Syntax *boolean* = *object.***Save**

Remarks The **Save** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
boolean	A variable of type **Boolean**

The **Save** method returns False if unsuccessful.

SaveAs Method

Applies To The following objects: **OPCalendar**, **OPCode**, **OPProject**, **OPProjectCode**, and **OPResource**

Description Saves the file represented by the specified object to the specified name.

Syntax *boolean* = *object.***SaveAs(***name, datasource)*

Remarks The **SaveAs** method syntax uses these parts:

Part	Description

object	An object variable of one of the types listed above
name	A string or variable of type **String** specifying the name in which to save the file
datasource	An empty string or string variable specifying an empty string.
boolean	A variable of type **Boolean**

The **SaveAs** method returns False if unsuccessful. Since Open Plan 3 does not use multiple data sources, the data source argument is ignored, but it cannot be omitted.

ScheduleDriverAnalysis Method

Applies To The **OPProject** object

Description Performs schedule driver analysis on an activity in the project.

Syntax *long = object*.**ScheduleDriverAnalysis** (*ActivityID, Target Date Field, Target Date, Number of Paths, Field to Store Results, Ignore Date Option*)

Remarks The **ScheduleDriverAnalysis** method syntax uses these parts:

Part	Description
activitId	Id of the target activity
Target Date Field	Manual or the name of a date field to be used as the target
Target Date	When the target date field is manual, this is the target used
Number of Paths	1 to N paths to analyze
Field to Store Results	Typically a User Character Field used to store the path number
Ignore Date Option	0 = All dates 1= Finish dates only

Select Method

Applies To **OPBaselinesList** object

Description Selects and loads the specifies **OPBaselines** object.

Syntax *selected = baselineslist*.**Select**(*index*)
selected = baselineslist.**Select**(*baselinename*)

Remarks The **Select** method syntax uses these parts:

Part	Description
baselineslist	An object variable of type **OPBaselinesList**
baselinename	A string or variable of type **String** specifying the name of an OPBaselines object
index	An integer or variable of type **Integer** that uniquely identifies an OPBaselines object within the OPBaselinesList collection
selected	A variable of type **Boolean**, where True indicates that the specified OPBaselines object was selected

This method also deselects and unloads the current **OPBaselines** object if it is not the one specified. If an out-of-range index or invalid name is specified, then all **OPBaselines** objects are deselected.

Selected Method

Applies To **OPActivity** object

Description Returns True if the activity represented by the specified object is selected in Open Plan.

Syntax *boolean* = *activity*.**Selected**

Remarks The **Selected** method syntax uses these parts:

Part	Description
activity	An object variable of type **OPActivity**
boolean	A variable of type **Boolean**

SetAssignmentFields Method

Applies To **OPActivities** collection

Description Retrieves the values from a list of fields in the assignment table and stores these values in a variable length, comma-delimited field in the activity table called **All_Asgns**. After performing the **SetAssignmentFields** method, the pseudo field **All_Asgns** is available to any method that reads other activity fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *activities*.**SetAssignmentFields**(*string*)

Remarks The **SetAssignmentFields** method syntax uses these parts:

Part	Description
activity	An object variable of type **OPActivities**
boolean	A variable of type **Boolean**
string	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetAssignmentFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetCalculatedFieldTo Method

Applies To **OPActivities** collection

Description Creates or updates a calculated field of a specified name with a specified *expression*.

Syntax *boolean* **=** *collection*.**SetCalculatedFieldTo**(*calcfieldname, expression, returntype)*

Remarks The **SetCalculatedFieldTo** method syntax uses these parts:

Part	Description
collection	An object variable of one of the types listed above
boolean	A variable of type **Boolean** indicating if the method was successful
calcfieldname	A string or variable of type **String** containing the name of the calculated field to be created or updated
returntype	A string or variable of type **String** containing the type of data that is to be returned by the calculated field
expression	A string or variable of type **String** containing the expression of the calculated field to be created or updated

Valid values for expression are: Character, Date, Decimal, Duration, Finish Date, Integer, or Logical.

> **Attention:** For more information on creating valid calculated field expressions, see **System Utilities » Calculated Fields** in the Open Plan online help.
>
> Note that *calcfieldname* represents a valid calculated field name in Open Plan and cannot include invalid characters or duplicate an existing field or filter name. This method is documented solely for OPP v1.2b compatibility. It has been superseded by the properties and methods belonging to the **OPCalculatedField** object.

SetCollectionGrowth Method

Applies To The following collections: **OPActivities**, **OPAssignments**, **OPCalendar**, **OPCode**, **OPCosts**, **OPPredecessors**, **OPProjectCode**, and **OPResource**

Description Enhances performance when adding objects to the specified collection by specifying the number of objects by which the potential size of the collection is incremented each time the current limit of the collection is exceeded.

Syntax *boolean* = *collection*.**SetCollectionGrowth**(*growth*)

Remarks The **SetCollectionGrowth** method syntax uses these parts:

Part	Description
collection	An object variable of one of the types listed above
boolean	A variable of type **Boolean** indicating if the method was successful
growth	An integer or variable of type Integer representing the number of objects by which to increase the potential size of the collection

The **SetCollectionGrowth** method returns 0 if unsuccessful.

SetCostFields Method

Applies To **OPActivities** collection

Description Retrieves the values from a list of fields in the cost table and stores these values in a variable length, comma-delimited field in the activity table called **All_Costs**. After performing the **SetCostFields** method, the pseudo field **All_Costs** is available to any method that reads other activity fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *activities*.**SetCostFields**(*string*)

Remarks The **SetCostFields** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
boolean	A Boolean value or variable of type **Boolean**
string	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetCostFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetCrosstabDates Method

Applies To **OPProject** object, **OPResource** collection

Description This method defines the reporting periods to be used when accessing time-phased earned value or resource data with the **GetEarnedValueCrosstabData** and **GetResourceCrosstabData** methods.

Syntax *object*.**SetCrosstabDates** *datearray*

Remarks The **SetCrosstabDates** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
datearray	A variant array

SetCrosstabDatesFromXML Method

Applies To **OPResource, OPProject objects**

Description This method reads an XML string to define the reporting periods to be used when accessing time-phased earned value or resource data with the GetEarnedValueCrosstabData and GetResourceCrosstabData methods.

Syntax *object*.**SetCrosstabDatesFromXML**(*dates*)

Remarks The **SetCrosstabDatesFromXML** method syntax uses these parts:

Part	Description
object	An object variable of one of the types specified above
dates	A string or variable of type **String**

SetCrosstabMinutesPerDefaultUnit Method

Applies To **OPResource object**

Description Sets the value for minutes per default duration unit.

Syntax *resource*.**SetCrosstabMinutesPerDefaultUnit***(minutes)*

Remarks The **SetCrosstabMinutesPerDefaultUnit** method syntax uses these parts:

Part	Description
resource	An object variable of type **OPResource**
minutes	An integer or variable of type **Integer**

SetCurrentFields Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAvailability**, **OPAssignment**, **OPCodeRecord**, **OPCost**, **OPPredecessor**, **OPProjectResource**, **OPResourceRecord**, **OPResourceUsage**, and **OPstep**.

Description Sets the values of a list of fields, which can include user-defined fields, from the data file represented by one of the object types listed above. The list of fields is defined by the **AssignCurrentFieldSet** method.

Syntax *boolean* = *object*.**SetCurrentFields**(*string*)

Remarks The **SetCurrentFields** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
boolean	A variable of type **Boolean** indicating if the method was successful

string	A string or variable of type **String** containing the new values for the fields to be updated, with each value separated by the pipe symbol (|)

The **SetCurrentFields** method returns 0 if unsuccessful.

SetDateAutoScale Method

Applies To **OPView** Object

Description Allows the user to specify the autoscale range with custom dates.

Syntax *bResult = OPView.***SetDateAutoScale***(dtFrom, dtTo)*

Remarks The method syntax uses these parts:

Part	Description
dtFrom	A date argument. It represents the start of the autoscale range.
dtTo	A date argument. It represents the end of the autoscale range.
bResult	The Boolean result of the method. The method will return true if the view is open, the view is a barchart, both dates are valid, and dtTo >= dtFrom. Note that the autoscale interval will be set to custom during this operation.

SetDateScaleOption Method

Applies To **OPView** Object

Description Allows the user to specify whether fixed time units or reporting calendar divisions will determine how the date axis is set up.

Syntax *bResult = OPView.***SetDateScaleOption***(sDateScaleOption)*

Remarks The method syntax uses these parts:

Part	Description
sDateScaleOption	A string argument. Valid choices are the following: ▪ Manual (use the time units that have previously been set up via the UI) ▪ Reporting Calendar (use the reporting calendar that has previously been set up via the UI)
bResult	The Boolean result of the method. The method will return true if the view is open, the view is a barchart, and one of the valid string options is passed as the argument. Note that, like the UI, the previously set manual units should be remembered, such that you may toggle back and forth between “Manual” and “Reporting Calendar” without losing the manual units that were previously set up. The UI may display an error message when the user switches between Manual and Reporting Calendar modes if the total height for all axes is below the minimum axis height. In this situation, automation adjusts the axis heights automatically instead of returning an error.

SetEarnedValueCrosstabOptions Method

Applies To **OPProject** object

Description This method defines the type of earned value data to be generated when accessing time-phased earned value data with the **GetEarnedValueCrosstabData** method.

Syntax *project*.**SetEarnedValueCrossTabOptions** *result, type, forecasttype*

Remarks The **SetEarnedValueCrossTabOptions** method syntax uses these parts:

Part	Description
Project	An object variable of type **OPProject**
result	A string or variable of type **String** representing one of the following case-sensitive values: QTY (quantity), COST, or ESCCOST (escalated cost).
type	A string or variable of type **String** representing one of the following case-sensitive values: PERIOD or CUMULATIVE
forecasttype	A string or variable of type **String** representing one of the following case-sensitive values: EARLY, LATE, or SCHEDULE

SetField Method

Applies To The following objects: **OPActivity**, **OPActivityResource**, **OPAssignment**, **OPAvailability**, **OPCalendar**, **OPCalendarRecord**, **OPCode**, **OPCodeRecord,** **OPCost**, **OPIcon, OPPredecessor**, **OPProjectResource**, **OPResource**, **OPResourceRecord**, **OPResourceUsage**, and **OPStep**.

Description Sets the value of any field, including user-defined fields, from the data file represented by one of the object types listed above.

Syntax *object*.**Setfield** *fieldname*, *fieldvalue*

Remarks The **Setfield** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
fieldname	A string or variable of type **String** containing the field name to be set
fieldvalue	The desired value for the field or a variable of the appropriate data type for the field to be set

SetFilterTo Method

Applies To **OPActivities** collection

Description Creates or updates a filter of a specified name with a specified filter *expression*.

Syntax *boolean* **=** *activities*.**SetFilterTo** *filtername, expression*

Remarks The **SetFilterTo** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
boolean	A variable of type **Boolean** indicating if the method was successful
filtername	A string or variable of type **String** containing the name of the filter to be created or updated

expression	A string or variable of type **String** containing the *expression* of the filter to be created or updated

If the **SetFilterTo** method is used to change the *expression* of a filter that is in use by a currently displayed view, the new filter *expression* is automatically applied to the records in that view.

> **Attention:** The *expression* string uses the same tables, fields, functions and operators available for use with Open Plan's calculated fields.
>
> For more information, see System Utilities » Calculated Fields in the Open Plan online help.
>
> Note that *filtername* represents a valid filter name in Open Plan and cannot include invalid characters or duplicate an existing field or filter name.

SetFocus Method

Applies To **OPWebWindow** object

Description Sets the focus to the WebWindow represented by the specified **OPWebWindow** object.

Syntax webwindow**.SetFocus**

Remarks The **SetFocus** method syntax uses these parts:

Part	Description
webwindow	An object variable of type **OPWebWindow**

SetNumberOfAxes Method

Applies To **OPView** Object

Description Allows the user to set the number of axes displayed.

Syntax *bResult* = *OPView*.**SetNumberOfAxes***(nNumOfAxes)*

Remarks The method syntax uses these parts:

Part	Description
nNumOfAxes	A short integer ranging from 1 to 6 that indicates the number of axes to be displayed in the date scale.
bResult	The Boolean result of the method. The method will return true if the view is open, the view is a barchart, and the number of axes to be displayed is a number between 1 – 6.

> **Note:** It is useful to set this before calling the SetDateAutoScale method to establish the desired number of axes.

SetPosition Method

Applies To **OPWebWindow** object

Description Sets the size and position of the window represented by the specified **OPWebWindow** object.

Syntax *webwindow*.**SetPosition** *top, left, height, width*

Remarks The **SetPosition** method syntax uses these parts:

Part	Description
webwindow	An object variable of type **OPWebWindow**
top	A variable of type **Long**
left	A variable of type **Long**
height	A variable of type **Long**
width	A variable of type **Long**

SetPredFields Method

Applies To **OPActivities** collection

Description Retrieves values from a list of relationship fields for the predecessors of the activity represented by the specified object and stores these values in a variable length, comma-delimited field in the activity table called **All_Preds**. After performing **SetPredFields**, the pseudo field **All_Preds** is available to any method that reads other fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *collection*.**SetPredFields**(*string*)

Remarks The **SetPredFields** method syntax uses these parts:

Part	Description
collection	An object variable of type **OPActivities**
boolean	A variable of type **Boolean**
string	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetPredFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetPrinter Method

Applies To **OPApplication** object

Description Sets the printer used for the current Open Plan session.

Syntax *bResult*=*OPApp*.**SetPrinter***(sPrinterName)*

Remarks The **SetPrinter** method syntax uses these parts:

Part	Description
sPrinterName	A string representing the printer name that will be set as the default printer in the Open Plan application
bResult	The Boolean result of the method. This method returns True if successful.

SetPrintDateOption Method

Applies To **OPView** Object

Description Allows the user to specify how the dates for the print range will be calculated.

Syntax *bResult* = *OPView*.**SetPrintDateOption***(sOption)*

Remarks The method syntax uses these parts:

Part	Description

sOption	A string argument. Valid choices are the following: ▪ Project (use the Project Start date and the calculated finish date) ▪ Range (use the range of the data being printed--which may be a filtered subset) ▪ Current (use the start and end dates of the view being displayed) Note that, to establish the calculated finish date for the "Project" option, the current logic takes the latest of the three following calculated dates (assuming all are valid)—early finish, late finish, and scheduled finish. For example, given a valid late finish date, if the scheduled finish date is not valid or earlier than the late finish date, the late finish date would be used instead.
bResult	The Boolean result of the method. The method will return true if the view is open, the view is a barchart, and one of the valid string options is passed as the argument.

SetPrintDateRange Method

Applies To **OPView** Object

Description Allows the user to specify custom dates for the print range.

Syntax *bResult = OPView.***SetPrintDateRange***(dtFrom, dtTo)*

Remarks The method syntax uses these parts:

Part	Description
dtFrom	A date argument. It represents the start of the print range.
dtTo	A date argument. It represents the end of the print range
bResult	The Boolean result of the method. The method will return true if the view is open, the view is a barchart, both dates are valid, and dtTo >= dtFrom.

SetPrintExpandColumns Method

Applies To **OPView** Object

Description Allows the user to specify whether spreadsheet columns will be padded out to fill a printed page.

Syntax *bResult = OPView.***SetPrintExpandColumns***(bPadColumnsToFillPage)*

Remarks The method syntax uses these parts:

Part	Description
bPadColumnsToFillPage	A Boolean argument. When set to true, columns will be increased in width so that they fill the printed page horizontally; if False, the columns will maintain their normal widths during printing.
bResult	The Boolean result of the method. This method should return True if a view is open, AND the view supports padding columns (i.e., a non-spreadsheet-based view such as the network view would return False).

SetPrintHeadingColorsOff Method

Applies To **OPView** Object

Description Allows the user to specify whether colors will be used when printing spreadsheet column headings.

Syntax *bResult = OPView.***SetPrintHeadingColorsOff***(bSuppressColHeadingColors)*

Remarks The method syntax uses these parts:

Part	Description
bSuppressColHeadingColors	A Boolean argument. When set to True, the foreground and background colors for a column heading will be set to black and white, respectively, regardless of the color settings in the spreadsheet preferences dialog; if False, the foreground and background colors set up in the preferences dialog will be honored.
bResult	The Boolean result of the method. This method should return True if a view is open, AND the view supports heading colors (i.e., a non-spreadsheet-based view such as the network view would return False).

SetPrintMargins Method

Applies To **OPView** Object

Description Sets the margins used in printing for the view

Syntax *bResult = OPView.***SetPrintMargins***(dTop, dLeft, dBottom, dRight)*

Remarks The method syntax uses these parts:

Part	Description
dTop	A double value representing the top margin of the printed page in inches or fractions of an inch (negative values are illegal).
dLeft	A double value representing the left margin of the printed page in inches or fractions of an inch (negative values are illegal)
dBottom	A double value representing the bottom margin of the printed page in inches or fractions of an inch (negative values are illegal)
dRight	A double value representing the right margin of the printed page in inches or fractions of an inch (negative values are illegal)
bResult	The Boolean result of the method. A value of True is returned if the method is successful; False is returned otherwise. The method will return False if the view is not opened, and, as implied above, it will also return False if any of the four values is negative.

Note: A double value is a number with two decimal places. For example, bResult=OPVIEW.SetPrintMargins(.10, .10, .20, .20)

SetPrintLineNumbers Method

Applies To **OPView** Object

Description Allows the user to specify whether line numbers will be printed.

Syntax *bResult = OPView.***SetPrintLineNumbers***(bPrintLineNumbers)*

Remarks The method syntax uses these parts:

Part	Description
bPrintLineNumbers	A Boolean argument. When set to True, line numbers will be printed for the spreadsheet section of the view. Setting the argument to False suppresses line number printing.
bResult	The Boolean result of the method. This method should return True if a view is open, AND the view supports line number printing (i.e., a non-spreadsheet-based view such as the network view would return False).

SetPrintNotes Method

Applies To **OPView** Object

Description Allows the user to specify which table-related notes will be printed with the view.

Syntax *bResult = OPView.***SetPrintNotes***(bIncludeProject, bIncludeActivity, bIncludeResource, bIncludeCode)*

Remarks The method syntax uses these parts:

Part	Description
bIncludeProject	A Boolean argument (True/False) indicating whether project-related notes should be included when printing. If this parameter is irrelevant to the view, it is ignored (no error is returned).
bIncludeActivity	A Boolean argument (True/False) indicating whether activity-related notes should be included when printing. If this parameter is irrelevant to the view, it is ignored (no error is returned).
bIncludeResource	A Boolean argument (True/False) indicating whether resource-related notes should be included when printing. If this parameter is irrelevant to the view, it is ignored (no error is returned).
bIncludeCode	A Boolean argument (True/False) indicating whether code-related notes should be included when printing. If this parameter is irrelevant to the view, it is ignored (no error is returned).
bResult	The Boolean result of the method. A value of True is returned if the view is open; False, otherwise.

SetResourceCrosstabOptions Method

Applies To **OPProject** object, **OPResource** collection

Description This method defines the type of earned value data to be generated when accessing time-phased earned value data with the **GetResourceCrosstabData** method.

Syntax *object*.**SetResourceCrosstabOptions** *result, type*

Remarks The **SetResourceCrosstabOptions** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
result	A string or variable of type **String** representing one of the following case-sensitive values: QTY (quantity), COST, or ESCCOST (escalated cost).
type	A string or variable of type **String** representing one of the following case-sensitive values: PERIOD, CUMULATIVE, or AVERAGE

SetResourceSelection Method

Applies To **OPCreateApplication33** object

Description The method selects specified resources in the active resource view.

Syntax *string* = *object*.**SetResourceSelection***(resourceasked)*

Remarks The **SetResourceSelection** method syntax uses these parts:

Part	Description
string	A variable of type **String** containing all selected resource names
object	An object variable of type **OPCreateApplication33**
resourceasked	A string or variable of type **String** representing resource names to be selected.

SetRights Method

Applies To **OPAccessControl** object

Description This method returns an integer error value.

Syntax *integer* = *object*.**SetRights** *(GroupID, UserID, RoleID, ReadOnly)*

Remarks The **SetRights** method syntax uses these parts:

Part	Description
object	An object variable of type **OPAccessControl**.
integer	A negative value indicates an error. Non-zero values are warnings or additional information. For example, the value OP_S_COLUMN_VALUE_IS_SAME (324032) indicates that one or more of the fields being set was not changed from the original value(s). SetRights is subject to the user's edit rights on the owner object.
GroupID	A string or variable of type **String** indicating the Group ID to be assigned to the associated access control record.
UserID	A string or variable of type **String** indicating the User ID to be assigned to the associated access control record.
RoleID	A string or variable of type **String** indicating the Role ID to be assigned to the associated access control record.

ReadOnly	A string or variable of type **String** is returned indicating whether the specified Group or User is to be assigned read-only access to the owner object (the owner object may be a project, calculated field, sort, etc.). The value will be either "Yes" or "No".

SetRiskFields Method

Applies To **OPActivities** collection

Description This method retrieves the values from a list of fields in the risk table and stores these values in a variable length, comma-delimited field in the activity table called **All_Risks**. After performing the **SetRiskFields** method, the pseudo field **All_Risks** is available to any method that reads other activity fields, for example, **GetFields** or **GetCurrentFields**.

Syntax *boolean* = *activities*.**SetRiskFields**(*string*)

Remarks The **SetRiskFields** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
boolean	A Boolean value or variable of type **Boolean**
string	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetRiskFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetSharedModeLock Method

Applies To **OPProject** object

Description This method sets or clears a shared mode lock for the specified **OPProject** object.

Syntax *long*= *object*.**SetSharedModeLock** (*setorclear, savechanges*)

Remarks The **SetSharedModeLock** method syntax uses these parts:

Part	Description
long	Error code where a zero value indicates success. A negative value indicates an error. Non-zero positive values are warnings or additional information.
setorclear	A Boolean value indicating whether or not the caller wishes to set or clear a shared mode lock. TRUE indicates that a lock should be set, while FALSE indicates that any existing lock should be cleared.
savechanges	When clearing a lock, the caller should set this variable to TRUE to save any changes created while the lock was active. Setting this parameter to false will cause any changes created while the lock was active to be discarded. This parameter is ignored when setting a shared mode lock.

SetSortFields Method

Applies To **OPActivities** collection, **OPPredecessors** collection

Description This method sorts the objects in the specified collection according to a specified sort expression.

Syntax *boolean* **=** *collection*.**SetSortFields**(*expression*)

Remarks The **SetSortFields** method syntax uses these parts:

Part	Description
collection	An object variable of one of the types listed above
boolean	A variable of type **Boolean** indicating if the method was successful
expression	A string or variable of type **String** containing the expression of the sort to be created or updated

The *expression* string consists of a field name, a forward slash, and a single character string designating the direction of the sort: A for ascending or D for descending. If the direction is omitted, ascending is assumed. Multiple fields/directions may be specified, each separated by commas. For example, “C1/A,C2/A,ESDATE/D”.

SetSortTo Method

Applies To **OPActivities** collection, **OPPredecessors** collection

Description This method creates or updates a sort of a specified name with a specified sort expression.

Syntax *boolean* **=** *collection*.**SetSortTo**(*sortname, expression*)

Remarks The **SetSortTo** method syntax uses these parts:

Part	Description
collection	An object variable of one of the types listed above
boolean	A variable of type **Boolean** indicating if the method was successful
sort	A string or variable of type **String** containing the name of the sort to be created or updated
expression	A string or variable of type **String** containing the expression of the sort to be created or updated

The *expression* string consists of a field name, a forward slash, and a single character string designating the direction of the sort: A for ascending or D for descending. If the direction is omitted, ascending is assumed. Multiple fields/directions may be specified, each separated by commas. For example, “C1/A,C2/A,ESDATE/D”.

SetStandardDays Method

Applies To **OPCalendarRecord** collection

Description Sets the Work flag for each day of the week in the specified calendar via a seven-character string representing the state of the Work flag for each day, where the first character in the string represents Sunday. Non-working days are represented by 0; working days are represented by 1.

Syntax *boolean* = *calendarrecord*.**SetStandardDays**(*string*)

Remarks The **SetStandardDays** method syntax uses these parts:

Part	Description
calendarrecord	An object variable of type **OPCalendarRecord**
Boolean	A variable of type **Boolean**
String	A string or variable of type **String**

SetStepFields Method

Applies To **OPActivities** collection

Description Retrieves values from a list of relationship fields for the steps of the activity represented by the specified object and stores these values in a variable length, comma-delimited field in the activity table called **All_Steps**. After performing **SetStepFields**, the pseudo field **All_Steps** is available to any method that reads other fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *collection*.**SetStepFields**(*string*)

Remarks The **SetStepFields** method syntax uses these parts:

Part	Description
collection	An object variable of type **OPActivities**
boolean	A variable of type **Boolean**
string	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetStepFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetSuccFields Method

Applies To **OPActivities** collection

Description Retrieves values from a list of relationship fields for the successors of the activity represented by the specified object and stores these values in a variable length, comma-delimited field in the activity table called **All_Succs**. After performing **SetSuccFields**, the pseudo field **All_Succs** is available to any method that reads other fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *collection*.**SetSuccFields**(*string*)

Remarks The **SetSuccFields** method syntax uses these parts:

Part	Description
Collection	An object variable of type **OPActivities**
Boolean	A variable of type **Boolean**
String	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetSuccFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetUsageFields Method

Applies To **OPActivities** collection

Description Retrieves the values from a list of fields in the usage table and stores these values in a variable length, comma-delimited field in the activity table called **All_Usages**. After

performing the **SetUsageFields** method, the pseudo field **All_Usages** is available to any method that reads other activity fields—**GetFields** or **GetCurrentFields**, for example.

Syntax *boolean* = *activities*.**SetUsageFields**(*string*)

Remarks The **SetUsageFields** method syntax uses these parts:

Part	Description
activities	An object variable of type **OPActivities**
Boolean	A Boolean value or variable of type **Boolean**
String	A string or variable of type **String** containing the names of the fields to be queried, with each field name separated by the pipe symbol (|)

The **SetUsageFields** method returns 0 if unsuccessful. The method will fail if an invalid field name is included in *string*.

SetViewHorizontalPage Method

Applies To **OPView** Object

Description Allows the user to specify whether the printed output will be fit to a restricted number of pages.

Syntax *bResult* = *OPView*.**SetViewHorizontalPage***(bFitToPage, nNumberofPagesToFit)*

Remarks The method syntax uses these parts:

Part	Description
bFitToPage	A Boolean argument indicating whether the view should be fitted to the page when printing (values are True or False).
nNumberOfPagesToFit	A short integer argument indicating how many pages will be used if the bFitToPage parameter = True. This number will be ignored if the bFitToPage parameter = FALSE, but if not, it must have a value >= 1.
bResult	The Boolean result of the method. The method will return True if the view is open, it is the proper type (barchart, tornado, or histogram), and, if the bFitToPage argument = TRUE, the nNumberOfPages argument >= 1.

SetViewLegend Method

Applies To **OPView** Object

Description Allows the user to specify where the legend will appear in the printed output

Syntax *bResult* = *OPView*.**SetViewLegend***(sViewLegendPosition)*

Remarks The method syntax uses these parts:

Part	Description
sViewLegendPosition	A string value that must be one of the following: ▪ Upper Left ▪ Upper Right ▪ Lower Left ▪ Lower Right ▪ None (do not include legend)

bResult	The Boolean result of the method. This method will return True if a view is open, it is a barchart or tornado view, one or more titleblock positions have been enabled for the position in question (e.g., “Top Left”, “Top Center”, or “Top Right” would need to be enabled if “Upper Left” or “Upper Right” is passed as argument to the method), and the argument passed is one of the four specified strings. The enabling of title block sections is consistent with what the UI enforces.

SetViewResourceData Method

Applies To **OPView** Object

Description Allows the user to specify whether the resource histogram will be printed.

Syntax *bResult = OPView.***SetViewResourceData***(bInclude, sPageOption)*

Remarks The method syntax uses these parts:

Part	Description
bInclude	A Boolean argument indicating whether the resource histogram should be included or not (values are True or False).
sPageOption	A string option indicating on which pages the histogram will be included. It is ignored unless bInclude = True. Valid choices are the following: ▪ Every Page ▪ Last Page
bResult	The Boolean result of the method. The method will return True if the view is open, if it is a barchart view, and, to be consistent with the UI, if the resource histogram is being included, it must be displayed with one or more valid resources selected.

SetViewSpreadsheet Method

Applies To **OPView** Object

Description Allows the user to specify whether the spreadsheet view will be included in printed output.

Syntax *bResult = OPView.***SetViewSpreadsheet***(bInclude, bVisibleColumnsOnly, sPageOption)*

Remarks The method syntax uses these parts:

Part	Description
bInclude	A Boolean argument indicating whether the spreadsheet should be included in the view (True = include, and False = exclude).
bVisibleColumnsOnly	A Boolean argument indicating whether printed output should show only columns that are visible on screen (True = print visible columns only, and False = print all columns).
sPageOption	A string argument that indicates on which page(s) the spreadsheet will be included. It will be ignored if argument bInclude = False. Valid choices are the following: ▪ Every Page ▪ First Page

bResult	The Boolean result of the method. The method will return True if the view is open, it is the proper type (barchart or tornado), and if bInclude = True, the sPageOption must be one of the valid choices.

Shifts Method

Applies To **OPDate** object, **OPStandardDay** collection

Description Retrieves the **OPShifts** collection object

Syntax Set *shifts* = *object.***Shifts**

Remarks The **Shifts** method syntax uses these parts:

Part	Description
object	An object variable of one of the types listed above
Shifts	An object variable of type **OPShifts**

Show Method

Applies To **OPCreateApplication33** object

Description Displays the Open Plan application window.

Syntax *object*.**Show**

Remarks The **Show** method syntax uses these parts:

Part	Description
Object	An object variable of type **OPCreateApplication33**

Shut Method

Applies To The following objects: **OPCalendar**, **OPCode**, **OPProject**, **OPProjectCode**, and **OPResource**.

Description Closes the file represented by the specified object.

Syntax *object.***Shut**

Remarks The **Shut** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above

ShutWOSave Method

Applies To The following objects: **OPCalendar**, **OPCode**, **OPProject**, **OPProjectCode**, and **OPResource**.

Description Closes the file represented by the specified object without saving it.

Syntax *object.***ShutWOSave**

Remarks The **ShutWOSave** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above

Snapshot Method

Applies To **OPView** Object

Description Creates a .jpg file "snapshot" of the view

Syntax *bResult = OPView.**Snapshot**(sFileName)*

Remarks The method syntax uses these parts:

Part	Description
sFileName	A fully qualified file name for the .jpg file that will be generated as a "snapshot" of the view.
bResult	The Boolean result of the method. The method may fail for the following reasons: ▪ The file name and/or folder name supplied is invalid ▪ Open Plan cannot write to the supplied file name for some reason ▪ An internal error occurs during JPEG generation

Sorts Method

Applies To **OPProject** object, **OPResource** and **ProjectResources** collections

Description Retrieves the **OPSorts** collection object. When *index* is used as an argument to the **Sorts** method, retrieves the **OPSort** object mapped into that index. When *name* is used as an argument to the **Sorts** method, retrieves the **OPSort** object that matches the specified name.

Syntax
Set *sorts* = *object.***Sorts**
Set *sort* = *object.***Sorts.Item(***index***)**
Set *sort* = *object.***Sorts.Item(***name***)**

Remarks The **Sorts** method syntax uses these parts:

Part	Description
Object	An object variable of one of the types listed above
Sorts	An object variable of type **OPSorts**
Sort	An object variable of type **OPSort**
Index	An integer or variable of type Integer that uniquely identifies an OPSort object within the OPSorts collection
Name	A string or variable of type **String** representing the name of the OPSort object to be retrieved

Spreadsheets Method

Applies To **OPFileCabinet** object, **OPProject** object

Description Retrieves the **OPSpreadsheets** or **OPFCSpreadsheets** collection object. When *index* is used as an argument to the **Spreadsheets** method, retrieves the **OPFCView** or **OPView** object mapped into that index. When *name* is used as an argument to the **Spreadsheets** method, retrieves the **OPFCView** or **OPView** object that matches the specified name.

Syntax **Set** *spreadsheets* = *object*.**Spreadsheets**
Set *spreadsheet* = *object*.**Spreadsheets.Item(***index***)**
Set *spreadsheet* = *object*.**Spreadsheets.Item(***name***)**

Remarks The **Spreadsheets** method syntax uses these parts:

Part	Description
object	An object variable of type **OPFileCabinet** or **OPProject**
spreadsheets	An object variable of type **OPFCSpreadsheets** or **OPSpreadsheets**
Spreadsheet	An object variable of type **OPFCView** or **OPView**
Index	An integer or variable of type **Integer** that uniquely identifies an OPFCView or OPView object within the OPFCSpreadsheets or OPSpreadsheets collection
Name	A string or variable of type **String** representing the identifier for the requested OPFCView or OPView object

StandardDay Method

Applies To **OPCalendarRecord** object

Description Retrieves the **OPStandardDay** collection.

Syntax **Set** *standardday* = *calendarrecord*.**StandardDay(***daystring***)**

Remarks The **StandardDay** method syntax uses these parts:

Part	Description
standardday	An object variable of type **OPStandardDay**
calendarrecord	An object variable of type **OPCalendarRecord**
daystring	A string or variable of type **String** containing one of the following values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday or Saturday

Steps Method

Applies To **OPActivity** object

Description Retrieves the **OPSteps** collection object. When *index* is used as an argument to the **Steps** method, retrieves the **OPSteps** object mapped into that index.

Syntax **Set** *steps* = *activity*.**Steps**
Set *steps* = *activity*.**Steps.Item**(*index*)

Example The first two lines (index) are equivalent to the next two lines (stepID):

Set *step* = myActivitiesSteps(index)
Set *step* = myActivites.Steps.Item(index)

Set *step* = myActivities.Steps(stepID)
Set *step* = myActivites.Steps.Item(stepID)

Note: The “myActivities.Steps” without any argument gives you the steps collection (**OPSteps**), whereas “myActivities.Steps” with an argument gives you the step object (**OPStep**).

Remarks The **Steps** method uses these parts:

Part	Description
steps	An object variable of type **OPSteps**
step	An object variable of type **OPStep**

activity	An object variable of type **OPActivity**
index	An integer or variable of type **Integer** that uniquely identifies an OPSteps object within the OPSteps collection

SysDir Method

Applies To **OPCreateApplication33** object

Description Returns the location of Open Plan's executable.

Syntax *sysdir* = *application*.**SysDir**

Remarks The **SysDir** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
Sysdir	A variable of type **String**

TimeAnalyze Method

Applies To **OPProject** object

Description Perform time analysis on the project.

Syntax *project*.**TimeAnalyze** *showdialog*

Remarks The **TimeAnalyze** method syntax uses these parts:

Part	Description
Project	An object variable of type **OPProject**
showdialog	A boolean value or variable of type **Boolean**, where "True" indicates that the Time Analysis Options dialog box is to be displayed

UpdateBaseline Method

Applies To **OPProject** object

Description Updates an existing baseline in the project represented by the specified **OPProject** object.

Syntax *boolean*= *object*.**UpdateBaseline**(*name, filter, progress, complete, useActuals*)

Remarks The **UpdateBaseline** method syntax uses these parts:

Part	Description
boolean	A variable of type **Boolean**
object	An object variable of type **OPProject**
name	A string or variable of type **String** that specifies the unique name of baseline.
filter	A string or variable of type **String** that specifies the name of a filter used to limit the activities to be updated in the baseline to a specific group of activities. If the name of the specified filter does not exist in the project, the method will fail.
progress	A Boolean value or variable of type **Boolean** that indicates whether in-progress activities are to be included in the baseline update. A value of FALSE will exclude in-progress activities from the new baseline.
complete	A Boolean value or variable of type **Boolean** that indicates whether completed activities are to be included in the

	baseline update. A value of FALSE will exclude complete activities from the new baseline.
useActuals	A Boolean value or variable of type **Boolean** that indicates whether progressed dates should be stored in the baseline for completed activities. A value of TRUE in this parameter indicates that *progress dates* will be stored in the baseline for in-progress and complete activities. A value of FALSE in this parameter indicates that *existing baseline dates* will be stored in the baseline for in-progress and complete activities. This parameter has no effect if completed is equal to FALSE. These parameters correlate to selection of the following options on the Update Baseline dialog box: FALSE = Keep Existing Baseline Dates TRUE = Update Baseline Dates with Current Plan Dates

The **UpdateBaseline** method returns False if unsuccessful.

UpdateBaselineEx Method

Applies To **OPProject** object

Description Updates an existing baseline in the project represented by the specified **OPProject** object.

Syntax *boolean= object*.**UpdateBaselineEx**(*name, filter, progress, complete, existing, rollup, delete*)

Remarks The **UpdateBaselineEx** method syntax uses these parts:

Part	Description
boolean	A variable of type **Boolean**
object	An object variable of type **OPProject**
name	A string or variable of type **String** that specifies the unique name of baseline.
filter	A string or variable of type **String** that specifies the name of a filter used to limit the activities to be updated in the baseline to a specific group of activities. If the name of the specified filter does not exist in the project, the method will fail.
progress	A Boolean value or variable of type **Boolean** that indicates whether in-progress activities are to be included in the baseline update. A value of FALSE will exclude in-progress activities from the new baseline.
complete	A Boolean value or variable of type **Boolean** that indicates whether completed activities are to be included in the baseline update. A value of FALSE will exclude complete activities from the new baseline.
existing	A Boolean value or variable of type **Boolean** that indicates whether progressed dates should be stored in the baseline for completed activities. A value of TRUE in this parameter indicates that *progress dates* will be stored in the baseline for in-progress and complete activities. A value of FALSE in this parameter indicates that *existing baseline dates* will be stored in the baseline for in-progress and complete activities. This parameter has no effect if completed is equal to FALSE. These parameters correlate to selection of the following options on the Update Baseline dialog box: FALSE = Keep Existing Baseline Dates TRUE = Update Baseline Dates with Current Plan Dates

rollup	A Boolean value or variable of type **Boolean** that is enabled only when updating existing baselines. A value of FALSE will exclude data from being rolled up for the existing baseline. A value of TRUE will roll up all dates, budget costs, and labor quantity data in the baseline to its parent level.
delete	A Boolean value or variable of type **Boolean** that is enabled only when updating an existing baseline. A value of FALSE indicates that deleted activities will not be deleted from the baseline. A value of TRUE indicates that any activities that no longer exist in the project will be deleted from the current baseline.

The **UpdateBaselineEx** method returns False if unsuccessful.

UpdateBaselineWithOptionFlags Method

Applies To **OPProject** object

Description Updates an existing baseline in the project represented by the specified **OPProject** object.

Syntax *boolean= object*.**UpdateBaselineWithOptionFlags**(*name, filter, flags*)
When more than one flag is passed the syntax looks like this

boolean= object.**UpdateBaselineWithOptionFlags**(*name, filter, flag + flag + flag*)

Remarks The **UpdateBaselineWithOptionFlags** method syntax uses these parts:

Part	Description
boolean	A variable of type **Boolean**
object	An object variable of type **OPProject**
name	A string or variable of type **String** that specifies the unique name of baseline.
filter	A string or variable of type **String** that specifies the name of a filter used to limit the activities to be updated in the baseline to a specific group of activities. If the name of the specified filter does not exist in the project, the method will fail.
Flags	An integer or variable of type **Integer** that specifies which of the OpenPlanBaselineOptionFlags will be used in creating the baseline.

The **UpdateBaselineWithOptionFlags** method returns **False** if unsuccessful.

> **Attention:** For a list of flags supported by Open Plan for the UpdateBaselineWithOptionFlags method, see Appendix A: Create/Update Baseline Option Flags.

Sample Script

```
OPProject.UpdateBaselineWithOptionFlags "PMB", "<None>",
Baseline_Include_In_Progress_Activities + Baseline_Include_Completed_Activities +
Baseline_Update_Data
```

User Method

Applies To **OPCreateApplication33** object

Description Returns the login name of the current user.

Syntax *username* = *application.***User**

Remarks The **User** method syntax uses these parts:

Part	Description
application	An object variable of type OPCreateApplication33
username	A variable of type String representing the name of a user

Version Method

Applies To **OPCreateApplication33** object

Description Returns the version number of the Open Plan executable, followed by a string designating the particular Open Plan product edition..

Syntax *version* = *application.***Version**

Remarks The **Version** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
version	A variable of type **String**

ViewCategory Method

Applies To **OPView** Object

Description Returns the friendly name for the category of the view object.

Syntax *sViewCategory* = *OPView.***ViewCategory**

Remarks The **ViewCategory** method syntax uses these parts:

Part	Description
sViewCategory	A string returned by the function. The function returns one of the following values: ▪ barchart • browser ▪ calendar • code ▪ histogram • multi-table barchart ▪ multi-table spreadsheet • network ▪ resource • risk ▪ spreadsheet

- tornado
- unknown

ViewClass Method

Applies To **OPView** object

Description Returns an integer indicating the class of the view: 501 (Network view), 502 (Spreadsheet), 505 (Code view), 507 (Resource view), 508 (Histogram), 509 (Barchart), 513 (Multi-table spreadsheet), 514 (Multi-table barchart), 515 (Risk view), 516 (Tornado view)

Syntax *integer* = *view*.**ViewClass**

Remarks The **ViewClass** method syntax uses these parts:

Part	Description
view	An object variable of type **OPView**
integer	A variable of type **Integer**

Views Method

Applies To **OPFileCabinet** object, **OPProject** object

Description Retrieves the **OPViews** or **OPFCViews** collection object. When *index* is used as an argument to the **Views** method, retrieves the **OPFCView** or **OPView** object mapped into that index. When *name* is used as an argument to the **Views** method, retrieves the **OPFCView** or **OPView** object that matches the specified name (which is the descriptive label that appears under the view's icon in the File Cabinet or project notebook).

Syntax
Set *views* = *object*.**Views**
Set *view* = *object*.**Views.Item(***index***)**
Set *view* = *object*.**Views.Item(***name***)**

Remarks The **Views** method syntax uses these parts:

Part	Description
object	An object variable of type **OPFileCabinet** or **OPProject**
views	An object variable of type **OPFCViews** or **OPViews**
view	An object variable of type **OPFCView** or **OPView**
index	An integer or variable of type Integer that uniquely identifies an OPFCView or OPView object within the OPFCViews or OPViews collection
name	A string or variable of type **String** representing the identifier for the requested OPFCView or OPView object

WebWindows Method

Applies To **OPCreateApplication33** object

Description Retrieves the **OPWebWindows** collection object.

Syntax **Set** *webwindows* **=** *application***.WebWindows**

Remarks The **WebWindows** method syntax uses these parts:

Part	Description
application	An object variable of type **OPWebWindows**

webwindows	An object variable of type **OPCreateApplication33**

WindowMinimizeAll Method

Applies To **OPCreateApplication33** object

Description Minimizes all windows within the Open Plan application window.

Syntax *application*.**WindowMinimizeAll**

Remarks The **WindowMinimizeAll** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**

WindowTile Method

Applies To **OPCreateApplication33** object

Description Tiles the open windows within Open Plan application window.

Syntax *application*.**WindowTile**

Remarks The method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**

WindowTileVertical Method

Applies To **OPCreateApplication33** object

Description Vertically tiles the open windows within Open Plan application window.

Syntax *application*.**WindowTileVertical**

Remarks The method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**

WorkDir Method

Applies To **OPCreateApplication33** object

Description Returns the location of Open Plan's WorkDir folder specified in the registry.

Syntax *workdir* = *application*.**WorkDir**

Remarks The **WorkDir** method syntax uses these parts:

Part	Description
application	An object variable of type **OPCreateApplication33**
workdir	A variable of type **String**

Examples

Constructing Filter Strings

The OPFilter object's Add method and Expression properties, and the OPActivities object's SetFilterTo method all require a filter string as a parameter. These filter strings can be quite complex, using the same functions and syntax available for use in Open Plan's calculated fields. Valid operators for Open Plan's data types are as follows:

	Enumerations	Booleans	Strings	Dates	Durations	Numerics
BETWEEN()			X	X	X	X
CONTAINS()			X		X	X
=	X	X	X	X	X	X
>=	X		X	X	X	X
>	X		X	X	X	X
IS_EMPTY			X	X		
<=	X		X	X	X	X
<	X		X	X	X	X
NOT_BETWEEN()			X	X	X	X
NOT_CONTAINS()			X		X	X
NOT_EMPTY			X	X		
<>	X	X	X	X	X	X

Literal dates should be enclosed in curly braces {}. Literal strings should be enclosed in quotes (single or double). Literal durations should be enclosed in piping symbols | |. If the duration identifier (m, w, d, h, t) is not specified, the default duration unit will be assumed. Enumerated or Boolean values should be enclosed in square brackets [ ].

Sometimes an enumerated value can apply to more than one field. For example, [Complete] can apply to the Computed Status (COMPSTAT) field as well as to the Progress Type (PROGTYPE) field. To avoid errors caused by ambiguous enumerated values, use the following data type abbreviations that clearly identify the desired enumeration:

Data Type Abbreviation	Open Plan Field	Open Plan Label
ACTS	COMPSTAT	Computed Status
ACTT	ACT_TYPE	Activity Type
BOOL	OUTOFSEQ	Boolean
CRIT	CRITICAL	Critical Flag
DIST	DSHAPE	Duration Distribution Shape

EVTE	EVT	Earned Value Technique
LOGI	LOGICFLAG	Activity Logic Flag
PRJS	OPSTAT	Project Status
PROG	PROGTYPE	Progress Type
RELT	RELTYPE	Relationship Type
REST	RES_TYPE	Res. Type
RESC	RES_CLASS	Resource Class
RSCL	RSCLASS	R/S Type
RSCL	COMP_RS_C	Scheduled Actions
TARG	TARGFTYPE	Target Finish Type
TARG	TARGSTYPE	Target Start Type

For example, when building a filter you might want to specify that the Computed Status is complete. In this case, you would use `COMPSTAT = [ACTS.Complete].`

Another way to avoid ambiguity with enumerated values is to convert the enumeration to a string. Again, to specify that the Computed Status is complete, you would use `str(COMPSTAT) = "Complete"`.

The BETWEEN(), CONTAINS(), NOT_BETWEEN(), and NOT_CONTAINS() operators are always followed by an expression enclosed in parentheses. There cannot be a space between the operator and the left parenthesis.

The logical operators **.and., .or., and .not.** may be used to link multiple expressions. The .not. operator is always used in conjunction with the .and. operator, as follows: **.and. .not.**

The following examples are all valid filter strings:

```
ACT_TYPE = [ACTT.ASAP]
ACT_TYPE = [ACTT.ASAP] .or. ACT_TYPE = [ACTT.ALAP]
str(ACT_TYPE) CONTAINS("Subproject")
str(ACT_TYPE) NOT_CONTAINS("Subproject")
DESCRIPTION BETWEEN("A", "D") .and. .not. EFDATE = {12/31/99 16:00}
ORIG_DUR = |17|
ORIG_DUR NOT_BETWEEN(|1d|, |5d|)
USER_NUM02 <> LEVEL(C1)
```

Excel Macro Example

The following listing is an example of an Excel macro written in Visual Basic for Applications. This macro, LoadStatusFromOPP(), automatically launches Open Plan, imports activity data from a specific project into an existing spreadsheet, and then closes Open Plan.

```
Sub LoadStatusFromOPP()
    'Declare objects and variables

    Dim OPApp As Object          'OPCreateApplication33 object
    Dim OPProject As Object      ' OPProject object
    Dim OPAct As Object          'OPActivities object
    Dim OPActRec As Object       'OPActivity object
    Dim iCurrProj As Integer     'Project count from OP
```

```
    Dim i As Integer                ‘Increment activity count
    Dim NumActs As Integer          ‘Activity count from OP
    Dim CurRow As Integer

    ‘Create the application object and suppress the OP window
    Set OPApp = GetObject(“”,”opp.application”)
    OPApp.Minimize

    ‘Load project

    OPApp.FileOpen “pharmacy”, “Project”, “Exclusive”

    ‘Define project and activity object variables
    iCurrProj = OPApp.Projects.Count
    Set OPProject = OPApp.Projects(iCurrProj)
    Set OPAct = OPProject.Activities

    ‘Create a variable for the number of activities in the project
    NumActs = OPAct.Count

    ‘Read data from OP into spreadsheet
        Call SetupSheet
        CurRow = 2
        For i = 1 to NumActs
            Set OPActRec = OPAct.Item(i)
            Cells(CurRow, 1).Value = OPActRec.ID
            Cells(CurRow,2).Value = OPActRec.Description
            Cells(CurRow,3).Value = OPActRec.EarlyStart
            Cells(CurRow,4).Value = OPActRec.EarlyFinish
            CurRow = CurRow + 1
         Next
    ‘Close project and reclaim memory from objects
    Finish:
    OPProject.Shut
    Set OPActRec = Nothing
    Set OPAct = Nothing
    Set OPProject = Nothing
    Set OPApp = Nothing
    Sheets("Project Information").Select
    Columns("B:B").EntireColumn.AutoFit
    Range("A1").Select

    End Sub
    Sub SetUpSheet()
     Sheets("Project Information").Select
     Cells.Select
     Selection.ClearContents
     Sheets("Template").Select
     Columns("A:D").Select
     Application.CutCopyMode = False
     Selection.Copy
     Sheets("Project Information").Select
     Cells(1).Select
    ActiveSheet.Paste
End Sub

Sub ClearSheet()
```

```
    Sheets("Project Information").Select
    Cells.Select
    Selection.ClearContents
    Range("A1").Select
End Sub

Sub StartDemo()
    Call LoadStatusFromOPP
End Sub

Sub Auto_Open()
    Call ClearSheet
End Sub
```

Early Binding

You can use either "early" or "late" binding to start an Automation session. Prior to version 2.0c, all Visual Basic code using Open Plan's automation objects used late binding to communicate with the Open Plan application. Late binding uses either the GetObject or CreateObject function to initialize Open Plan. Each object is then declared as a generic Object-type variable. For example, the following code sets an object to the Open Plan program, which is the highest level object in the Open Plan object model. All late-bound Automation code must first define an Opp.Application object in order to access any of the other objects below that.

```
Sub LateBound()

    'Declare objects and variables

    Dim OPCreateApplication33 As Object
    Dim OPProjects As Object
    Dim OPProject As Object

    'Get the OPCreateApplication33 object
    Set OPCreateApplication33 = GetObject("", "opp.application")
    'Get an OPProject object from the OPProjects collection
    Set OPProjects = OPCreateApplication33.Projects
    Set OPProject = OPProjects.Item(1)

    'Using the Maximize method to Maximize the Project window
    OPProject.Maximize

    'Disassociate object variables to release resources
    Set OPProject = Nothing
    Set OPCreateApplication33 = Nothing

End Sub
```

Early binding also uses the GetObject or CreateObject function to initialize Open Plan, but each object is then declared as a specific type of object. Early binding has two main advantages. First, code using early binding runs faster than code using late binding. Second, Visual Basic can perform various type checking operations before an application is run, cutting down on potential errors in your finished code. To use early binding, you first need to reference the Open Plan type library. The methods for doing this will vary depending on your programming environment.

The following code is an example of using early binding to initialize Open Plan.

```
Sub EarlyBound30()
```

```
    'Declare objects and variables
    Dim OPCreateApplication33 As New OPP30.OPCreateApplication33
    Dim OPProjects As OPP30.OPProjects
    Dim OPProject As OPP30.OPProject

    'Get an OPProject object from the OPProjects collection
    Set OPProjects = OPCreateApplication33.Projects
    Set OPProject = OPProjects.Item(1)

    'Using the Maximize method to Maximize the Project window
    OPProject.Maximize

    'Disassociate object variables to release resources
    Set OPProject = Nothing
    Set OPProjects = Nothing
    Set OPCreateApplication33 = Nothing
End Sub
```

Notice that the New keyword was used to create the Open Plan application object. This top-level object (OPCreateApplication33) is the only object permitted to be created using the New keyword, and this object must be created before any of the other objects can be created. It is not necessary to use the New keyword. If “New” is omitted, you must use CreateObject or GetObject to create the Open Plan application object, as shown in the following example.

```
Sub EarlyBound30_2()

    'Declare objects and variables
    Dim OPCreateApplication33 As OPP30.OPCreateApplication33
    Dim OPProjects As OPP30.OPProjects
    Dim OPProject As OPP30.OPProject

    'Get the OPCreateApplication33 object
    Set OPCreateApplication33 = CreateObject("opp.application")

    'Get an OPProject object from the OPProjects collection
    Set OPProjects = OPCreateApplication33.Projects
    Set OPProject = OPProjects.Item(1)

    'Using the Maximize method to Maximize the Project window
    OPProject.Maximize

    'Disassociate object variables to release resources
    Set OPProject = Nothing
    Set OPProjects = Nothing
    Set OPCreateApplication33 = Nothing
End Sub
```

Using the SetCrosstabDates Method

The following example shows how to use the SetCrosstabDates method to define the date array used by the GetEarnedValueCrosstabData and GetResourceCrosstabData methods. This method allows you to specify the dates used when generating earned value or resource crosstab data, similar to using a reporting calendar in Open Plan.

```
Sub NewTest()
    'declare objects and variables
    Dim OppApp As Object
```

```
Dim OppProject As Object
Dim OppProjResources As Object
Dim OppProjResource As Object
Dim MyDateArray As Variant
Dim MyFloatArray As Variant
Dim sExportFileName As String
Dim iFileNumber As Integer
Dim sDateArrayString As String
Dim x As Integer
Dim y As Integer
Dim z As Integer
Dim sResID As String
Dim Quote As String
Dim Comma As String
Dim sFloatArrayString As String

'get the application and the project
Set OppApp = CreateObject("opp.application")
Set OppProject = OppApp.Projects("clean")

Quote = Chr$(34)
Comma = ","

'define and open the export file
sExportFileName = "c:\output.txt"
iFileNumber = FreeFile
Open sExportFileName For Output As iFileNumber

'load a variant array with dates & use the SetCrosstabDates method
ReDim MyDateArray(7) As Date
MyDateArray(1) = #1/1/95#
MyDateArray(2) = #3/1/95#
MyDateArray(3) = #6/1/95#
MyDateArray(4) = #8/1/95#
MyDateArray(5) = #12/1/95#
MyDateArray(6) = #6/1/96#
MyDateArray(7) = #12/1/96#
OppProject.SetCrosstabDates MyDateArray

'write the dates to the export file, with each element in quotes & separated by commas
sDateArrayString = Quote & "Res. ID"
'notice that the first element in the array is skipped
For y = LBound(MyDateArray) + 1 To UBound(MyDateArray)
sDateArrayString = sDateArrayString & Quote & Comma & Quote & Format$(MyDateArray(y))
Next y
sDateArrayString = sDateArrayString & Quote
Print #iFileNumber, sDateArrayString

'set the options for the method you'll use
OppProject.SetResourceCrosstabOptions "QTY", "PERIOD"

'get the project resources
Set OppProjResources = OppProject.Resources
For x = 1 To OppProjResources.Count
    z = 0
    ReDim MyFloatArray(0) As Single
```

```
                'get a single project resource
                Set OppProjResource = OppProjResources.Item(x)
                If x = 1 Then
                    'the first item in the collection is always the activity-level costs
sFloatArrayString = Quote & "Activity-level costs"
                Else
sFloatArrayString = Quote & OppProjResource.ID
                End If
                'get that resource's crosstab data into the array
                OppProjResource.GetResourceCrosstabData "EARLY", MyFloatArray

                'write the crosstab data to the file
                'notice that the last element in the array is skipped
                For z = LBound(MyFloatArray) To UBound(MyFloatArray) - 1
                    sFloatArrayString = sFloatArrayString & Quote & Comma & Quote & Format$(MyFloatArray(z))
                Next z

                sFloatArrayString = sFloatArrayString & Quote
                Print #iFileNumber, sFloatArrayString

        Next X
End Sub
```

Using the AssignCurrentFieldSet Method

The AssignCurrentFieldSet method is used to define a list of fields that can then be manipulated with the GetCurrentFields or SetCurrentFields methods. These methods make it possible to set or retrieve the values of several fields at one time, which can enhance the performance of automation applications.

The following example shows how to retrieve values using GetCurrentFields.

```
Sub m_GetCurrentFields()

        'Declare objects and variables
        Dim OPCreateApplication33 As Object
        Dim OPProject As Object
        Dim OPActivities As Object
        Dim OPActivity As Object
        Dim sFieldValues As String
        Dim sFieldSet As String
        Dim bSuccess As Boolean

        'Get the OPCreateApplication33 object
        Set OPCreateApplication33 = GetObject("", "opp.application")

        'Get an OPProject object from the OPProjects collection
        Set OPProject = OPCreateApplication33.Projects(1)

        'Get the OPActivities collection from the OPProject object
        Set OPActivities = OPProject.Activities

        'Get the OPActivity object from the OPActivities collection
        Set OPActivity = OPProject.Activities(1)

        'Using the AssignCurrentFieldSet method to define the list of fields to be used
        'by the GetCurrentFields & SetCurrentFields methods
```

```
        sFieldSet = "TFDATE|TARGFTYPE"
        bSuccess = OPActivities.AssignCurrentFieldSet(sFieldSet)
        'The method returns a Boolean indicating if the method was successful
        Debug.Print bSuccess

        'Use the GetCurrentFields method to return the values for the fields specified
        'with the AssignCurrentFieldSet method
        sFieldValues = OPActivity.GetCurrentFields
        Debug.Print sFieldValues

        'Disassociate object variables to release resources
        Set OPActivity = Nothing
        Set OPActivities = Nothing
        Set OPProject = Nothing
        Set OPCreateApplication33 = Nothing

End Sub
```

The following example shows how to set values using SetCurrentFields.

```
Sub m_SetCurrentFields()
        'Declare objects and variables
        Dim OPCreateApplication33 As Object
        Dim OPProject As Object
        Dim OPActivities As Object
        Dim OPActivity As Object
        Dim sFieldValues As String
        Dim sFieldSet As String
        Dim bSuccess As Boolean

        'Get the OPCreateApplication33 object
        Set OPCreateApplication33 = GetObject("", "opp.application")
        'Get an OPProject object from the OPProjects collection
        Set OPProject = OPCreateApplication33.Projects(1)
        'Get the OPActivities collection from the OPProject object
        Set OPActivities = OPProject.Activities
        'Get the OPActivity object from the OPActivities collection
        Set OPActivity = OPProject.Activities(1)

        'Using the AssignCurrentFieldSet method to define the list of fields to be used
        'by the GetCurrentFields & SetCurrentFields methods
        sFieldSet = "TFDATE|TARGFTYPE"
        'The method returns a Boolean indicating if the method was successful
        bSuccess = OPActivities.AssignCurrentFieldSet(sFieldSet)
        Debug.Print bSuccess

        'Use the SetCurrentFields method to return the values for the fields specified
        'with the AssignCurrentFieldSet method
        sFieldValues = "Fixed Target|12/31/2002 16:00"
        bSuccess = OPActivity.SetCurrentFields(sFieldValues)
        'The method returns a Boolean indicating if the method was successful
        Debug.Print bSuccess
        Debug.Print OPActivity.GetCurrentFields

        'Disassociate object variables to release resources
        Set OPActivity = Nothing
```

```
    Set OPActivities = Nothing
    Set OPProject = Nothing
    Set OPCreateApplication33 = Nothing

End Sub
```

Import and Export Facilities

Open Plan includes several facilities for importing and exporting project data:

- An import/export facility for MSP projects
- An import/export facility for Primavera P6 projects
- An export facility for IPMDAR projects
- A basic import facility capability for projects created with Primavera Project Planner®
- A customizable import/export facility based on user-defined scripts that can be used to transfer project data to and from a wide range of external applications.
- An import/export facility for XML documents

Note: In order to import data into an existing project, the project and all of its ancillary files must first be opened in exclusive mode.

For more information see *Importing and Exporting Project Data* in the Deltek Open Plan online help.

Integration with Microsoft Project

You can use Open Plan's import and export interface to import Microsoft Project .mpp files and Project Server projects. The import facility provides a seamless integration with Microsoft's current standard data formats.

Attention: For more information about integrating with Microsoft Project, see **Importing and Exporting Project Data » Microsoft Project Integration** in the Deltek Open Plan online help.

Importing Microsoft Project or Project Server Data

You can display the first page of the Microsoft Project Import Wizard by clicking **MS Project** on the **Integration tab Import** menu group.

Exporting Open Plan Data to Microsoft Project or Project Server

You can export data from an Open Plan project to a Microsoft Project file or Project Server project. You can display the first page of the Microsoft Project Export Wizard by clicking **MS Project** on the **Integration tab Export** menu group.

Before you can export a project to Microsoft Project, it must first be saved as an Open Plan project.

View Exported Data

If you want to review the data that was exported to Microsoft Project Professional or Server, add **_KEEP_MSP_OUTPUT_FILES** with a value of **1** to the System Preferences dialog box ( »

Preferences » Application » Advanced tab). Open Plan creates .csv integration files and saves them to the Open Plan working folder, for example, **Documents » Deltek » Open Plan Professional 8.6**. Each file has a unique name that includes the Open Plan project name and a time stamp.

If the setting is enabled, the generated .csv files will remain in the folder until you delete them (disabling the setting will not delete accumulated files). You may want to periodically delete them to keep the folder size from becoming too large.

Limitations When Using Project Server Directly for Import/Export

The following are limitations when you use Project Server directly for import or export:

- Project Server does not support external subprojects.
- Calendar limitations:
  - Local (non-enterprise) calendars cannot have an enterprise base calendar.
  - Calendar shift, working-day hours, and work weeks are not available.
  - Calendars cannot be assigned to tasks or resources.
- **EVM** (Earned Value Method) cannot be specified for tasks.
- Although the task Type field is read-only, Open Plan specifies a Fixed Duration default task type for the project.
- Resource time-phasing information is not available.
- Notes limitations:
  - **Task notes** may be assigned when the task is created, but task notes may not be subsequently added.
  - **Resource notes** cannot be created from Open Plan.
  - When a note contains special characters, it gets truncated before the first special character.
- The **WBS** field is read-only, both for new and existing tasks.
- The resource type (work, material, or cost) cannot be set for non-enterprise (local) resources.
- Although a resource may have start and end availability dates, it is not associated with a collection of multiple availability records.
- Relationship lags are not available.
- Assignment data cannot be changed after the assignment has been created, but Open Plan can delete and recreate the assignment.
- The Task Actual Duration field is read-only.
- Task baseline fields (Baseline Start and Baseline Finish) are read-only.
- Task, resource, and project custom fields may be created within the Project Server environment. Open Plan imports and exports task and resource custom fields, but not project custom fields. User fields that are specific to MSP (for example, TEXT1, NUMBER1, and so on) are not available.

MSP Import/Export Modes

The export module was previously limited to a single create/overwrite mode when exporting data from Open Plan to MSP. You are now able to update status in existing MSP projects. Additionally, you may limit the activities exported to MSP by setting a filter.

For both the import and export modules, Open Plan processes resource assignments differently by taking advantage of the MSP time-scaled objects collection. When exporting, this allows the simulation of offsets and periods on multiple resource assignments in OP for the same activity/resource description combination by setting different values in the MSP time-scaled buckets for that activity/resource. When importing, conversely, multiple assignments can be created in OP from a single assignment in MSP based on differences between the time buckets.

With importing, you can now create or update the PMB baseline for the OP project if a baseline (the master baseline, not baselines 1 through 10) has been established in MSP for the imported project. Baselines are not exported back to MSP.

Finally, in order to preserve the export functionality that customer who are used to the MSPRIF utility expect, MSPRIF Compatible export option has been added, which is intended to give backwards compatibility with OP3.3.2 and 3.4.1 for activity progress and resource assignments.

MSP Export: Expected Behavior in Each Export Mode

Exported Collection	Create/Replace Mode	Status Update Mode	MSPRIF Compatibility Notes
Calendars (if an export filter is specified, the result is similar to using the _EXPORT_ONLY_CALS_IN_USE preference, except that the calendars must be in use **and** associated with activities included by the filter).	All calendars and calendar exceptions1 exported	New calendars and their calendar exceptions1 exported (we do not update/delete exceptions on an existing calendar in MSP).	N/A
Resources (if an export filter is specified, the result is similar to using the _EXPORT_ONLY_RES_IN_USE preference, except that the resources must be in use **and** associated with activities included by the filter).	All resources and availabilities on resources exported (calendars assigned to availabilities will also be exported)	New resources, their availabilities, and availability calendars are added; user-mapped fields are exported.	N/A
Tasks (may be filtered by the export filter)	All tasks and their progress information are exported.	New tasks are exported in their entirety (may be filter). User-mapped fields on all existing activities are exported (may be filter).	With MSPRIF turned on, task progress is exported as it was in 3.4.1. If the option is NOT enabled, remaining duration, percent complete, and

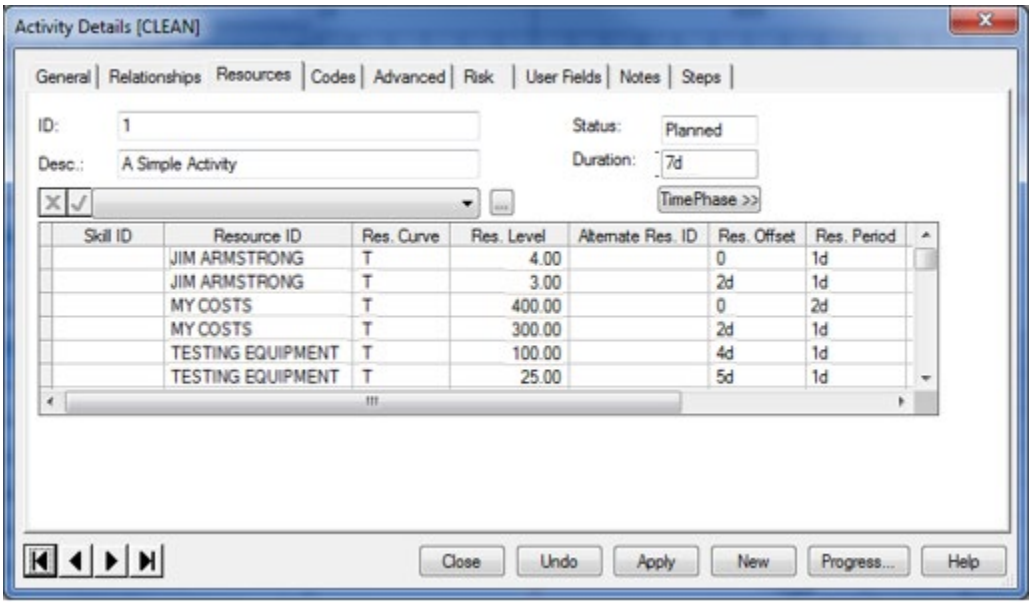
Exported Collection	Create/Replace Mode	Status Update Mode	MSPRIF Compatibility Notes
		The following fields in MSP **may** be updated without mappings as a part of the progress update: duration, actual dates, progress values as noted in the MSPRIF column, constraint type, and constraint date. See comments in "**MSPRIF Compatibility Notes**" mode column for information on updating progress in MSP.	complete progress types are exported
Relationships (will be included if no export filter or if associated activities have been included by the export filter)	All relationships for included activities are added (both predecessor and successor must be included by the export filter, if specified).	Relationships are added in cases where a newly added activity is either a predecessor or successor and the other end of the relationship is an included activity. Existing relationships are not updated.	N/A
Assignments (will be included if no export filter or if associated activities have been included by the export filter)	Assignments for all included tasks are exported. If resource progress is enabled, either remaining or % complete will be exported to the MSP assignment. See comments in "**MSPRIF Compatibility Notes**" mode column for information on updating progress in MSP.	Assignments for an included activity will be created or replaced in MSP with current data.	With MSPRIF turned on, resource assignments are exported as they were in 3.4.1. This also means that resource progress is not exported. If the option is NOT enabled, resource progress may be exported and we write detailed time phased data for assignments

[1] "Exceptions" refers to special days, such as holidays or normal non-working days (e.g., Saturday and Sunday) that become a working day.

Exporting OP Assignments to Microsoft Project

Exporting Levels, Offsets, and Periods

Let's begin by looking at several assignments for an activity in OP.



There are four resource categories in Open Plan:

- Labor
- Material
- Other direct costs
- Subcontract

There are three resource categories in MSP:

- Work
- Material
- Cost

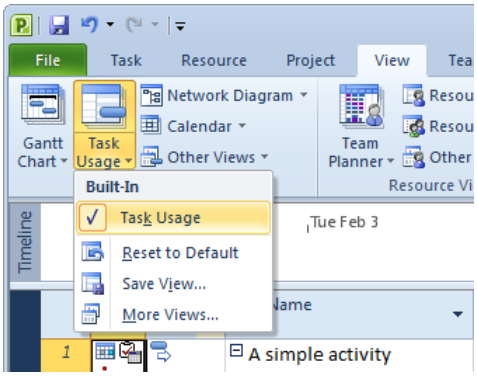
In our example above, *Jim Armstrong* represents a **labor** resource in OP (corresponding to a **work** resource in MSP). *MY COSTS* is an **other direct costs** resource (corresponding to a **cost** resource in MSP). *TESTING EQUIPMENT* is a **material** resource in both OP and MSP. A **subcontract** resource (not shown in the example) in OP is treated as a material resource in MSP.

OP Resource Type	MSP Resource Type
Labor	Work
Material	Material
ODC (other direct cost)	Cost
Subcontract	Material

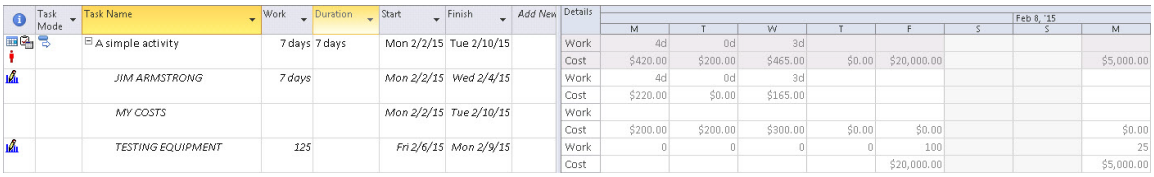
In OP, the combination of resource ID + Skill ID + Cost Class defines a resource assignment of a certain type. For the purpose of creating or updating a resource assignment within MSP, we only use the resource ID or Skill ID to define a resource of a certain type. The Skill ID will be ignored, unless the Resource ID field is blank. The *Cost Class* field is ignored, so the first two assignments above for *Jim Armstrong* represent a single assignment in MSP. Since OP does not export *pool* (parent) resources to MSP, if the resource ID listed is a pool (*not* the case in this example), we will try to resolve the pool to a child resource. We may be able to do this by examining usage records for the activity.

Therefore, in this example, we will create or replace three resource assignments for activity 1 since we have three different assignment combinations. In order to translate the offsets and periods shown into properly allocated time-scaled buckets in MSP, we must deselect the **MSPRIF Compatibility** option in the **Microsoft Project Export — Options Page**.

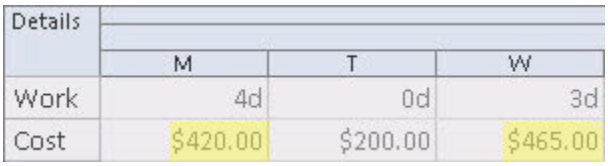
After exporting, open the project within MSP. From the standard Gantt chart (go to the **View** tab if the **Task Usage** option is not available on the **File** tab), choose to display *task usage*:



The **Task Usage** grid will appear to the right of the spreadsheet with **Work** displayed under the **Details** column. The following will now be displayed in MSP:



Looking first at *Jim Armstrong* in OP, note that for one of *Jim's* assignments (offset = 0, period = 1d), the offset/period is reflected in the first day of the period shown in MSP, with the level per day of 4 (as in OP). Since the other assignment for *Jim* has an offset of 2 days, the second day of the period is empty, and the third period contains the level of 3:



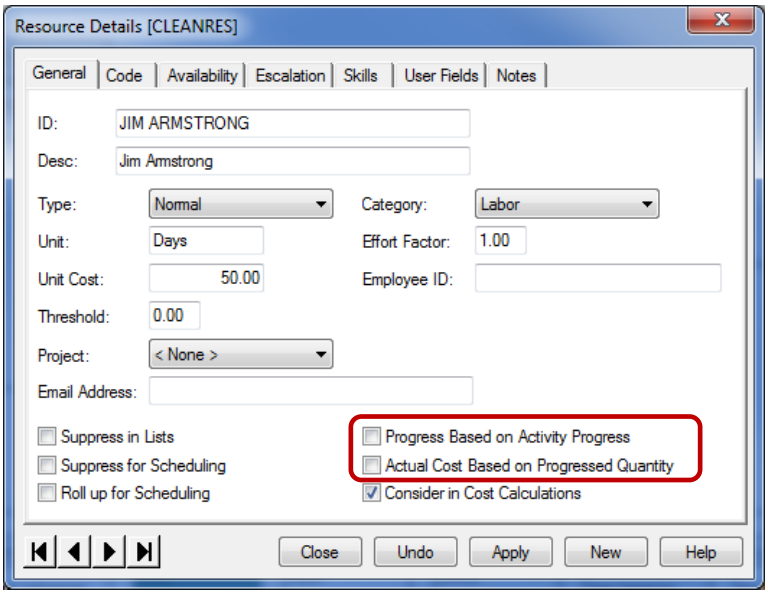
Note: MSP will apply the escalated cost as opposed to the standard cost (if escalated costs exist for the resource), we need to make sure we are comparing the cost values shown in MSP to the OP *Resource Escalated Cost*.

One other thing to note with the export of resource assignments: when exporting with the **MSPRIF Compatibility** option enabled, the assignments *units* field in MSP is set to represent the level per day in OP. When this option is disabled, and time-scaling (as above) is also being done, MSP is allowed to set

the units field based on the work or cost values that was furnished. Typically for labor (work) resources, this will amount to the units value being set to 1. This is not a problem since the work and cost values are the more important values for MSP. The units field also does not reflect levels accurately for offsets and periods in OP.

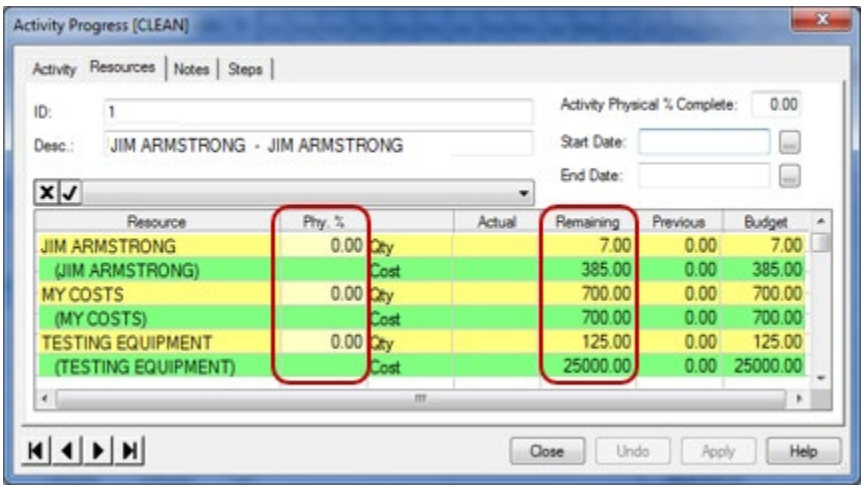
Exporting Resource Assignment Progress

As mentioned earlier, the **Resource Progress** option in the **Microsoft Project Export — Options Page** is enabled only if the **MSPRIF Compatibility** option is *disabled,* AND you choose to export resources. You may choose either to export *remaining duration* or *resource physical percent complete* to MSP. In order to set either of these values within the OP Activity Progress dialog, the **Progress Based on Activity Progress** option in the **Resource Details** dialog box must be *deselected* to modify values in the **quantity** rows. The **Actual Cost Based on Progressed Quantity** option must be *deselected* to modify values in the **cost** rows.



In the **Activity Progress** dialog box in OP, each resource group will appear on its own row (assignments for the same group will be summarized).

With the six assignments shown earlier in the example, we will see the following in the **Activity Progress** dialog box:



The values in the **Phy. %** and **Remaining** columns can be changed to apply progress to the resources. If we change the values as follows:

Resource	Phy. %		Actual	Remaining	Previous	Budget
JIM ARMSTRONG	33.00	Qty		4.00	0.00	7.00
(Jim Armstrong)		Cost		0.00	0.00	385.00

The changes made can be seen in the Resource Assignment spreadsheet (RESSPDVW):

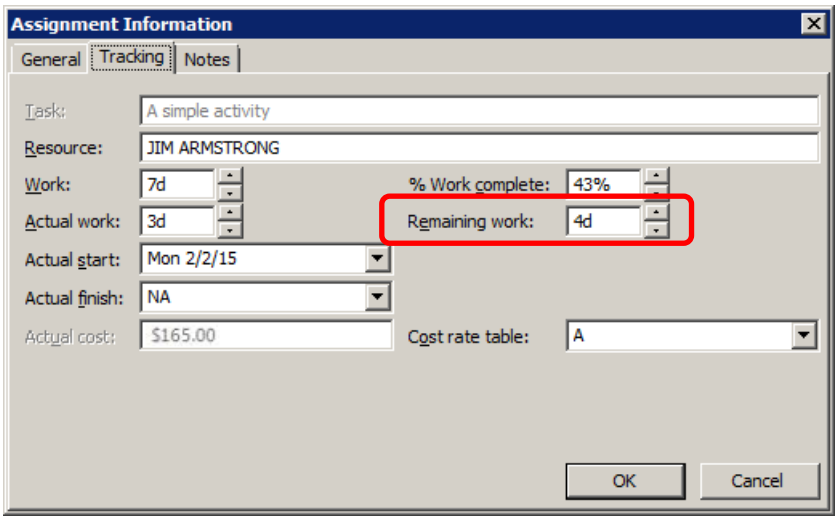
Resource ID	Res. Level	Res. Offset	Res. Period	Remaining Requirement	Physical % Complete
JIM ARMSTRONG	3.00	2d	1d	0.00	33.00
JIM ARMSTRONG	4.00	0	1d	4.00	33.00

Note: Change in remaining duration has been applied to one of the two assignments, while the change in physical % complete has been applied to both.

If the project is exported to MSP and the **Resource Progress** option is set to **Use Assgn. Remaining**, the following will be seen in MSP if you choose to display T**ask Usage** and add **Actual Work** to the **Details** column. While there is no direct display of remaining duration, you can calculate this by subtracting *work* – *actual work* (7 days – 3 days = 4 days)

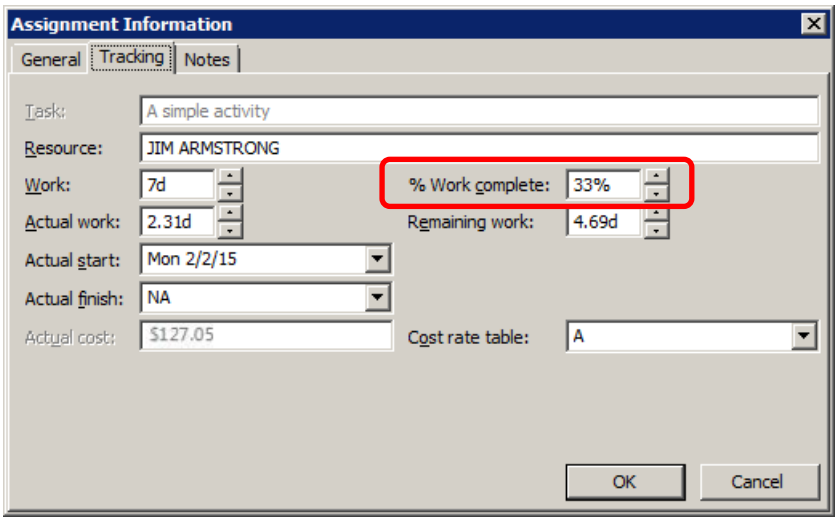
Task Name	Work	Details	M
JIM ARMSTRONG	*7 days*	Work	4d
		Act. W	3d

You can see remaining work by double-clicking the **Task Usage** row for *Jim Armstrong* to invoke the **Assignment Information** dialog box in MSP:



MSP calculates **% Work Complete** based on the remaining work that we set when exporting.

If you rerun the export, this time setting the **Resource Progress** option to **Assgn. PCC**, **% Work Complete** reflects the value we exported from OP, and remaining work is calculated.



If multiple assignments exist in OP for the same activity/resource combination, the PPC will be calculated for each of the multiple assignments by multiplying the PPC for the assignment by the fraction of the total level (across the multiple assignments) that the single assignment represents.

For example:

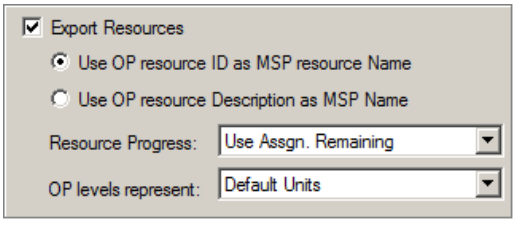
Assign. #	Total Level	PPC	Contribution
1	15	20%	.2 * (15/55) = 5.45%
2	10	15%	.15 * (10/55) = 2.73%
3	30	25%	.25 * (30/55) = 13.64%
Total	55	21.82%	

MSP does not allow you to set remaining cost on cost resources. Instead, we set the % work complete (calculated from remaining cost). Progress on cost resources is NOT time-scaled because of this.

Additional Notes on Export Functionality

Duration Units and Costs while Exporting

The *Default Units* chosen for an OP project (in the **Project Properties** dialog box on the **Preferences** tab) may affect resource costs. As far as MSP export is concerned, the units that are of interest to us are *days* or *hours* (any other time unit is exported as *days* for purposes of the export). On the **Options** page of the MSP Export Wizard in the Export Resources group, we may set the option “OP levels represent”:



If *Default Units* is chosen here, MSP Export will use the value chosen in the **Project Properties** dialog box on the **Preferences** tab for the Default Duration Unit (days or hours). Otherwise, it will use the user

selected value (days or hours). The option chosen in the **Microsoft Project Export — Options Page** may override the setting in **Project Properties**. OP does not specify a **time** unit on its resources, simply a unit. The unit is interpreted based on the **Project Properties** *default units* setting. MSP, however, always specifies a time unit for resource costs. Depending on the setting in the **Microsoft Project Export — Options Page**, the cost for resource *Jim Armstrong*, for example, may be specified as $50.00/**day** or $50.00/**hour** in MSP. So, if *Jim Armstrong* has a level of 5/day, the cost per day of that resource per day will be $50.00 * 5 units/day ($250.00) if the levels represent days, and $50.00 * 8 hours/day * 5 units/day ($2000.00) if the levels represent hours.

For best compatibility, the resource unit chosen in OP should represent the same time unit for all resources (if possible), and the *default units* (in OP) should reflect the chosen unit. If the MSP option is then set to *Default Units*, then everything should line up.

Using Filters while Exporting

A filter in OP may be applied in various ways. In an outlined view, such as the bar chart, if a child activity is included in a filter, its parent(s) will normally be included as well (this is referred to as filtering from the "bottom up"). The export filter is implemented for MSP Export to include only those activities exactly matching the filter (parents are excluded unless they also match the filter).

If an export filter is defined, this affects not only the activities that are exported, but also limits the calendars, resource definitions, relationships, and assignments that are exported. For records from any of these ancillary or child collections to be exported to MSP, it is necessary that they be associated with activities included by the filter. For example, if the filter limits the activity collection in OP to a single activity, only assignments for that single activity will be created or updated in MSP. In turn, only resource definitions needed for those assignments will be exported.

MSPRIF Compatibility while Exporting

The **MSPRIF Compatibility** option (available on the **Microsoft Project Export — Options Page**) is designed to suppress new functionality in MSP export that might undermine the main purpose of the old MSPRIF Visual Basic application, now incorporated into the OP product, which is to produce a schedule in MSP whose dates reflect, as closely as possible the schedule in OP. If the **MSPRIF Compatibility** option is enabled, assignments will be exported to MSP in the same fashion as they are currently in the 3.4.1 and 3.3.2 versions of OP

- multiple assignments for the same activity/resource combination will be combined, but the time-scaled collection on the MSP assignment will NOT be used to simulate offsets and periods in OP;
- resource progress will NOT be applied since it could conceivably cause task dates and durations to change (with the option enabled, the Resource Progress option in the Microsoft Project Export — Options Page is disabled). In addition, with the option enabled, MSP export will look at the computed status (COMPSTAT) field on the OP activity table to determine whether an activity is planned, started, or complete. COMPSTAT is accurate only if time analysis is performed, and we cannot assume that Time Analysis has been performed prior to export the data. If the MSPRIF option is not enabled, we will ignore the COMPSTAT field when exporting progress and instead use the values in the PROGTYPE and PROGVALUE fields on the activity table.

Additional Export Notes When Updating Existing MSP Files

- The update mode assumes that the Open Plan project being exported to MSP contains MSP unique ID values on the activity table (OPP_ACT), and if resources are also being exported, the MSPUNIQUEID field should also be populated on the resource table (OPP_RES). This will only be true if the OPP project being exported was previously imported from MSP. These values are used to connect OP records to the corresponding MSP records.

- If a record has been added in OP, it will have an MSPUNIQUEID value of 0, indicating that the task (or resource) needs to be created in MSP.
- If a previously imported MSP task record has been deleted in MSP, the activity in OP will have an MSPUNIQUEID value > 0 in OP, but the record will not be able to be matched in MSP (the unique ID for the deleted record is “retired” by MSP). We do not delete the OP activity but we do change the unique ID stored in OP back to 0. If a subsequent export is performed, OP will recreate the activity in MSP as it will appear to be new.
- If a record is deleted in OP, we make no attempt, at this time, to delete the record in MSP as well.

Integration with Primavera P6

You can use Open Plan's integration interface to import Oracle Primavera P6 projects (including steps) from a XER or an XML file or P6 web. The import facility provides a seamless integration with Primavera's current standard data formats.

You can import P6 XER or XML files or you can import multiple projects from P6 Web as long as they are all from the same P6 web site and database. When you import, you can create a new Open Plan project, or update an existing Open Plan project (if it was originally imported from P6) and share resources, codes, and calendars.

Display the first page of the Oracle Primavera P6 Import Wizard by clicking **Oracle Primavera » P6 Import** on the Integration tab Import menu group.

Attention: For information about best practices for importing/exporting data to/from P6, see the Open Plan online help topic titied: *Best Practices for Importing/Exporting Data To/From P6* (Importing and Exporting Project Data » Oracle Primavera Integration » Oracle Primavera P6).

Exporting Open Plan Activity Status to Oracle Primavera P6

You can export from an Open Plan project (including steps) to an Oracle Primavera P6 XER or XML file or P6 Web.

- Web export: A P6 web export only exports activity status/resource assignment status. The project that you are exporting must have previously been imported into Open Plan from P6.
- XER or XML export: A XER or an XML export exports the entire Open Plan project.

Display the first page of the Oracle Primavera P6 Export Wizard by clicking **Oracle Primavera** on the Integration tab Export menu group.

View Exported Data

If you want to review the data that was exported to Oracle Primavera P6, add **_KEEP_P6_OUTPUT_FILES** with a value of **1** to the System Preferences dialog box (» Preferences » Application » Advanced tab). Open Plan creates .csv integration files and saves them to the Open Plan working folder, for example, Documents » Deltek » Open Plan Professional 8.6. Each file has a unique name that includes the Primavera P6 project name and a time stamp.

If the setting is enabled, the generated .csv files will remain in the folder until you delete them (disabling the setting will not delete accumulated files). You may want to periodically delete them to keep the folder size from becoming too large.

Attention: For more information about integrating with Primavera P6, see **Importing and Exporting Project Data » Oracle Primavera Integration** in the Deltek Open Plan online help.

Limiting Calendar and Resource Export

You can add these options to the System Preferences dialog box (» Preferences » Application » Advanced tab) and specify whether you want Open Plan to export all of the calendars and resources in the calendar and resource file, or only those being referenced by the data.

_EXPORT_ONLY_P6_CALS_IN_USE

- **0** = Open Plan exports all of the calendars in the calendar file assigned to the project, whether or not they are being used by an activity. This is the default.
- 1 = Open Plan exports only those calendars that are being referenced by the data being exported. Open Plan follows these rules:
  - If a child calendar is exported, the parents in the hierarchy are also included in the export.
  - Calendars are included for all activities being exported.
  - The projects default calendar is always exported if it is not blank.
  - The projects default relationship calendar is always exported if it is not blank.
  - Calendars are included for all selected baselines on the open project.
  - A relationship calendar where both ends of the relationship are associated with activities being exported will be exported as long as it is not blank and is not <Calendar of Successor>.
  - If resources are being exported, calendars linked to availability records linked to included resources are also exported.

_EXPORT_ONLY_P6_RES_IN_USE

- **0** = Open Plan exports all of the resources in the resource file assigned to the project, whether or not they are being used by an activity. This is the default.
- 1 = Open Plan exports only those resources that are being referenced by the data being exported. Open Plan follows these rules:
  - If a child resource is exported, the parents in the hierarchy are also included in the export.
  - Export resources assigned to activities being exported.
  - If a resource assignment, resource usage record, or baseline usage record is associated with an activity that is being exported, Open Plan exports all resources referenced in the following fields: **RES_ID**, **RES_SKL_ID**, and **ALT_RES_ID**.
  - If a cost record is associate with an included activity, Open Plan exports resources referenced in the **RES_ID** and **RES_SKL_ID** fields.

Updating Existing Data When You Import Data From a XER or an XML File Into P6

By default, when you import data from a XER or an XML file into P6 and update an existing project, it keeps the existing values for the following fields:

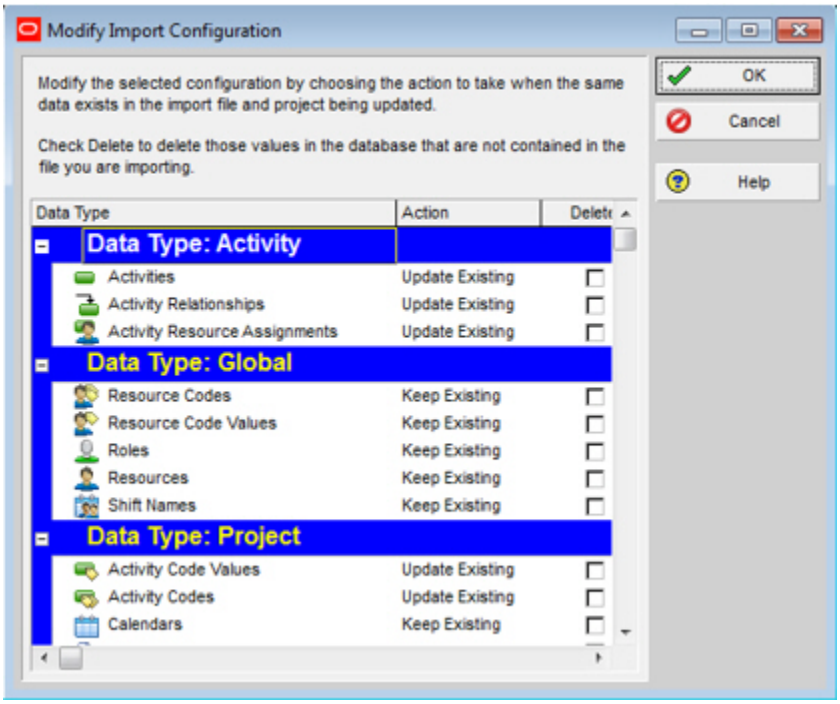
- Resource Codes

- Resource Code Values
  - Rates
- Resources
  - Short Names
- Calendars

That means that if you have made changes to any of the above fields in Open Plan before exporting to the file you are importing into P6, the data will not be updated in the existing project.

To update the above fields:

1. In the P6 Import wizard, when you get to the Update Project Options page, click **Modify**.
2. Click in the Actions column for any of the fields you want to update that have **Keep Existing** selected and change the setting to **Update Existing**.



Using the P6 API to Assign the P6 WBS Hierarchy to a User-Specific Index

By default, the P6 WBS code is stored at code index #1 in Open Plan; however, you can change the WBS code index number using the Open Plan API.

The **P6WBSIndex** element specifies the code index in a project where the P6 WBS structure is initially stored (for import) or retrieved (for export). The range of valid values for the index is **1** through **90** with the default value being **1**.

When the code assignment is made during a new import, the prompt for the code file is set to **P6WBS**. Once the **P6WBS** prompt exists on the project, its index position is used for future imports and exports instead of the **P6WBSIndex** value. If needed, you can use the Open Plan API to assign the **P6WBS** code prompt to any code index you want. Make sure that you use that prompt only once within any given project. If Open Plan finds this prompt either within an Open Plan project being updated, or a project

being created from a project template, it will use the corresponding code index to locate existing WBS elements or store new WBS elements.

Integrating with IPMDAR

The Integrated Program Management Data Analysis Report format (IPMDAR) is a Department of Defense contractor reporting format that uses the JSON data-exchange format.

Export Schedule Data to an IPMDAR File

You can export schedule data from an Open Plan project to an IPMDAR .zip file. You can display the IPMDAR Schedule Export Options Dialog Box by clicking **IPMDAR** on the Integration tab in the Export menu group.

Before you can export a project to IPMDAR, the project must first be saved as an Open Plan project.

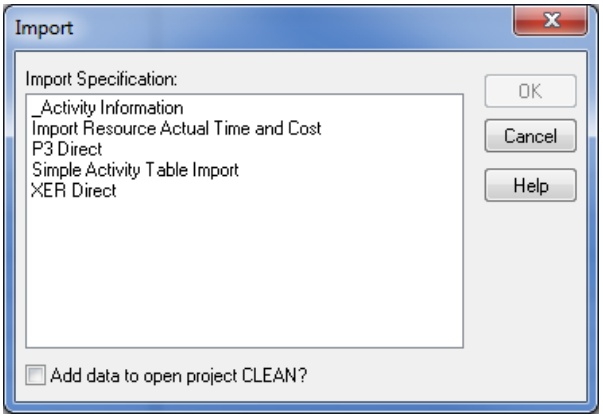
Custom Import/Export Options

In addition to importing/exporting Microsoft Project and Primavera projects, Open Plan also offers a more general capability based on user-defined import/export specifications. Each specification consists of a series of commands and parameters that allow Open Plan to import and export project, resource, code, and calendar information to a wide range of data formats.

Open Plan provides a number of standard import/export specifications that allow you, for example, to export a simple comma-delimited text file containing activity IDs and descriptions that can be read by many types of applications. It is also possible to create your own scripts for special-purpose operations.

General Import

To import project information, select the Integration tab and click **General** in the Import menu group to display the following dialog box:



Attention: For a description of the scripts, see Custom Import/Export Scripts.

You can use this dialog box to select an import specification. When you click **OK**, Open Plan allows you to open a source data file and create an untitled project containing the imported information. If the project includes references to ancillary data files such as resource or code files, Open Plan creates untitled versions of these files as well.

If you have a project open when you issue the **Import General** command, Open Plan allows you to indicate if the imported data should be used to update the current project. Note, however, that whether or

not the imported data overwrites existing records is determined by a parameter set in the import specification.

To import project data, complete the following steps:

1. On the **Integration** tab, click **General** in the **Import** menu group.
2. Select the specification for the operation.

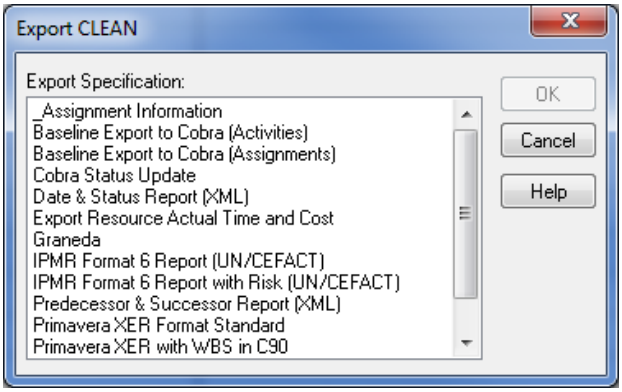
   Select the **Add data to open project <project name>** option if you want Open Plan to update the current project with the imported data.
3. Click **OK**.
4. Select the source data file and click **Open**.

   Open Plan displays an untitled project for the imported project.
5. To save the project as an Open Plan project, click  » **Save** or **Save As**.

   If the project includes references to ancillary data files, you must save the ancillary data files as well.

General Export

To export project information, select the Integration tab and click **General** in the Export menu group to display the following dialog box:



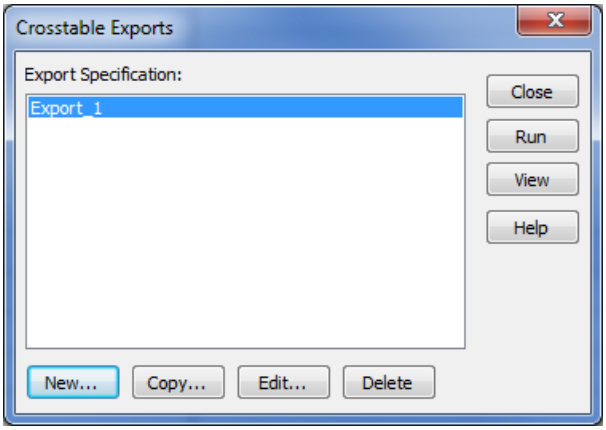
Attention: For a description of the scripts, see Custom Import/Export Scripts.

To export project data, complete the following steps:

1. Open the project you want to export.
2. On the **Integration** tab, click **General** in the **Export** menu group.
3. Select the export specification for the operation.
4. Click **OK**.
5. Select a destination folder for the project and click **Save**.

Crosstable Export

To export crosstable data, click **Crosstable Data** from the **Export** group of the **Integration** tab. This opens the **Crosstable Exports** dialog box.



The export specifications listed on this dialog box include those that you have created as well as those that other users have created and opted to share. When you select an export specification and click **Run**, you are given the opportunity to save the export file in either .xml or .csv format. You may also click on **View** to see those export files that have been utilized in the past.

> **Note:** Elements that have a value of zero are exported as a blank value instead of a zero (0).

The buttons along the bottom of the dialog box allow you to create, copy, edit, and delete export specifications. When you create, copy, or edit an export specification, a tabbed dialog box allows you to select the elements to include in the export file. The default view is the **General** tab.

> **Attention:** For more information including field descriptions for each tab, see **Importing and Exporting Project Data » Crosstable Exports** in the Deltek Open Plan online help.

Custom Import/Export Scripts

When you click **General** in the Import or Export menu group on the **Integration** tab, Open Plan displays a list of processing scripts that can be used to import or export project data. By default, all general import or export scripts are based on the contents of the Transfer.dat file, located in the local system folder.

Transfer.dat is a text file that can be edited using any text editor. It can contain both script definitions and pointers to separate include files. Each line of a script must start with a command word.

> **Warning:** All import/export command scripts are case-sensitive. If you enter these commands in lower or mixed case, you may receive error messages during the import or export procedure.

In cases where a local version of Transfer.dat is not present, for example, workstation installations, use a copy of Transfer.dat stored in the working folder of the Open Plan server. The contents of Transfer.dat is displayed in two dialog boxes:

- The Import dialog box lists the following import scripts by default:
  - **_Activity Information** — This script creates activities in Open Plan with the following information:

- ID
- Description
- Early Dates
- Late Dates
- Scheduled Dates
- User Character Fields 1 and 2

- **Import Resource Actual Time and Cost** — This script imports actual time and cost data into the following fields of the Open Plan CST table:
  - ACT_ID (Activity ID)
  - RES_ID (Resource ID)
  - ACWP_QTY (Actual Quantity of Work Performed for the period)
  - ACWP_CST (Actual Cost of Work Performed for the period)
  - START_DATE (Period Start date)
  - END_DATE (Period Finish date)

> **Note:** This script adds new records to the existing data. It does not update existing records.

- **Simple Activity Table Import**— This script imports a comma-delimited text file for use when importing data from the following ACT table fields:
  - OPP_ID (Activity ID)
  - DESCRIPTION (Activity Description)
  - ORIG_DUR (Original Duration)
  - ESDATE (Early Start date)
  - EFDATE (Early Finish date)
  - COMPSTAT (Computed Status)

> **Note:** Selecting the **Add data to open project <name>** option on the Import dialog box allows Open Plan to add the data to the open project but does not overwrite the existing data.

- The Export dialog box lists the following export scripts by default:
  - **Baseline Export to Cobra (Activities)** — This script prepares a transaction file for exporting activity baseline information into Cobra.
  - **Baseline Export to Cobra (Assignments)** — This script prepares a transaction file for exporting resource assignment baseline information into Cobra.
  - **Cobra Status Update** — This script prepares a transaction file for exporting status update information into Cobra.

- **Date &Status Report (XML)** — This script exports activity date and status information to an XML file that uses the supplied stylesheet Actdata2.xsl to display the exported data.
- **Export Resource Actual Time and Cost** — This script prepares a comma-delimited text file for use as a transaction file when exporting data from the following cost table fields:
  - ACT_ID (Activity ID)
  - RES_ID (Resource ID)
  - ACWP_QTY (Actual Quantity of Work Performed for the period)
  - ACWP_CST (Actual Cost of Work Performed for the period)
  - START_DATE (Period Start date)
  - END_DATE (Period Finish date)
- **GRANEDA** — This script exports project, activity, resource assignment, and relationship data in the format that is used by American Netronic's GRANEDA.
- **IPMR Format 6 Report (UN/CEFACT)** — This script exports project information that meets the US Government IPMR Format 6 requirements.
- **IPMR Format 6 Report with Risk (UN/CEFACT)** — This script exports project information that meets the US Government IPMR Format 6 requirements with the addition of Optional risk data for optimistic, pessimistic and most likely durations.
- **Predecessor & Successor Report (XML)** — This script exports activity and relationship data to an XML file that uses the supplied stylesheet Actdata1.xsl to display the exported data.
- **Resource/Activity Report (XML)** — This script exports information about activities and resource assignments to an XML file that uses the supplied stylesheet Rdsdata.xsl to display the export data.
- **Simple Activity Table Export**— This script prepares a comma-delimited text file for use as a transaction file when exporting data from the following OPP_ACT table fields:
  - OPP_ID (Activity ID)
  - DESCRIPTION (Activity Description)
  - ORIG_DUR (Original Duration)
  - ESDATE (Early Start date)
  - EFDATE (Early Finish date)
  - COMPSTAT (Computed Status)
- **Primavera XER Format Standard** — This script creates a Primavera P6 .xer file from the currently opened Open Plan Project. Code fields 1 – 20 (if available) are included.
- **Primavera XER with WBS in C90** — This script creates a Primavera P6 .xer file from the currently opened Open Plan Project. Code fields 1 – 20 (if available) are included and code field 90 (if available) is included.

> **Note:** A sample application (RunXMLReports.vbs) that uses the XML Export scripts is added to the **Add-Ins** tab during the Open Plan installation.

Basic Import and Export Scripts

> **Note:** If used as delimiters in any field, certain symbols may cause problems when importing and exporting data. These symbols are quotation marks (“), commas (,), piping symbols (|), and semicolons (;).

The following example is designed to help you understand some of the basic import and export scripting features in Open Plan. Assume that your installation includes the following scripts:

- Exporting basic activity information using a comma-delimited text file
- Importing basic activity information using a comma-delimited text file

An example of the Transfer.dat contents might display as follows:

```
EXPORT csv Simple Activity Export to Excel
REM exports ID,Desc,Duration,ESDate,EFDate,Activity Cost
                    TABLE ACT
SORT ACT_ID
                    FIELD ACT_ID
FIELD DESCRIPTION
FIELD ORIG_DUR
FIELD ESDATE
IMPORT csv Simple Activity Import from Excel
REM imports ID, Description, and Duration
TABLE ACT
FIELD ACT_ID
FIELD DESCRIPTION
FIELD ORIG_DUR

EXPORT xml Example XML Activity Export 1
INCLUDE ACTDATA1.XFR
```

> **Note:** The following conventions are used in the following discussion of commands:
>
> <…> Required parameters
>
> […] Optional parameters
>
> […|…] Choice of optional parameters

Notice that two special commands define the start of a new import or export script. Each import script must begin with an **IMPORT** command that uses the following syntax:

```
IMPORT <extension> <name>
```

Everything from this command to the next IMPORT or EXPORT command (or to the end of file) will define the import script. The *<name>* appears in the Open Plan **Import** dialog box.

The IMPORT command must have a default extension associated with the ASCII source file, for example:

IMPORT csv Excel file

This extension can be a wild card character such as an asterisk (*), for example:

IMPORT * Excel

Each export script must begin with an **EXPORT** command using the following syntax:

EXPORT <*extension*> <*name*>

Everything from this command to the next IMPORT or EXPORT command (or to the end of file) will define the export script. The <*name*> appears in the Open Plan **Export** dialog box.

As in the case of the IMPORT command, EXPORT must have a default extension associated with the ASCII destination file, for example:

EXPORT XML Extended Markup Language (HTML)

This extension can be a wild card character such as an asterisk (*), for example:

EXPORT * XML

Following the IMPORT or EXPORT command, the two scripts for comma-delimited data include one or more instances of the following commands:

- TABLE
- RECORD_TYPE
- FIELD

The **TABLE** command indicates the current table for the script using the following syntax:

TABLE <optabletype>

The <*optabletype*> parameter can refer to either one of Open Plan's internal table types or to one of a small number of pseudo-tables defined in the section "Table Types" in this document. Everything from this command to the next TABLE, IMPORT, EXPORT, or end of file applies to this table.

The **RECORD_TYPE** command defines an identifier for the record type for this table, which is assumed to be the first item of each record. This command uses the following syntax:

RECORD_TYPE <*recordtype*>

The **FIELD** command designates the next field from the current table to be processed. This command uses the following syntax:

FIELD <opfieldname> [<width>|<header>][<format>]

The <*width*> and <*header*> settings are optional parameters. If the file is a fixed format, <*width*> indicates the width of the field. If the file is a delimited format (for example, with commas), and a preceding HEADER command has defined the header record type, the <*header*> parameter represents the identifier for this field.

The <*format*> parameter is an optional setting used when exporting numeric, date, or enumerated field data from Open Plan. You can specify different numeric formats using the following format conventions:

Format	Result
(	If the numeric value is negative, suppress the minus sign and print left parentheses instead. Ignore if the numeric value is positive.

)	If the numeric value is negative, print right parentheses. Ignore if the numeric value is positive.
9	Print the digit from the source data, replacing leading zeros with spaces.
0	Print the digit from the source data, replacing leading spaces with zeros.
_(underscore)	Print the digit from the source data, suppressing leading spaces
. or,	Inserts a period (.) or a comma (,)
%X	Inserts the local currency symbol
%%	Inserts the percent sign (%)
%(	Inserts a left parenthesis
%)	Inserts a right parenthesis

For example, assume a 10-character field named PPC contains the value 27.42. The following table demonstrates how to use the formats given in the preceding table and the result they would each return.

Command	Result
FIELD PPC 10 _____	27^^^^^^^
FIELD PPC 10 0000000000	0000000027
FIELD PPC 10 9999999999	^^^^^^^27
FIELD PPC 10 999,999.99	^^^^^27.42
FIELD PPC 10 %X_____	$27^^^^^^

You can use the <*format*> parameter to define the format of dates. For example, %M%D%Y produces dates such as 123103.

The <*format*> parameter can also be used to export the short form of an Open Plan enumerated field. For example, FIELD REL_TYPE S will export “FS” instead of “Finish To Start.”

Attention: For a description of date format strings in Open Plan, see **Views and Reports » Tabs and Dialog Boxes » Date Formats Dialog Box** in the Deltek Open Plan online help.

For export, the order of the FIELD commands defines the order of the fields to be output. This is also true for a fixed format import, or for an import without a header record. In the case of an import with a header record, only those fields defined on the header record are expected.

Notice that the previous example of a Transfer.dat file ends with an **INCLUDE** command. The INCLUDE command allows script files to be included by reference using the following syntax:

INCLUDE <*filename*>

INCLUDE files may be nested but must not include an IMPORT or an EXPORT command.

Note: To see an example of an INCLUDE file, open one of the .xfr files provided with your installation of Open Plan.

Additional Commands

In addition to the basic commands described above, custom import/export scripts can include any of the following commands:

- **ADD_MISSING_KEYS** — This **Import General** feature supports the keyword ADD_MISSING_KEYS. This keyword allows Open Plan to automatically create records for data that is missing from an import file. For example, assume that the import file has resources assigned to its activities but that the resource file assigned to the project contains no data. In this case, Open Plan can automatically create the referenced records in the resource file.

- **DATE_FORMAT —** Defines the format dates using the following syntax:

DATE_FORMAT <*format*>

The <*format*> parameter uses the date formats available in Open Plan. For example, %M%D%Y produces dates such as 123103.

Attention: For a description of date format strings in Open Plan, see **Views and Reports » Tabs and Dialog Boxes » Date Formats Dialog Box** in the Deltek Open Plan online help.

The DATE_FORMAT command ignores all values occurring after a space. To include a space in a date, put the <*format*> syntax within quotation marks. For example:

Command	Result
DATE_FORMAT %M/%D/%C_%H:%T	06/02/2003_08:00
DATE_FORMAT %M/%D/%C %H:%T	06/02/2003
DATE_FORMAT "%M/%D/%C %H:%T"	06/02/2003 08:00

This command applies to the current table and all tables until the next occurrence of DATE_FORMAT.

Be aware that Open Plan requires two-digit month, day, and time formats when importing information from a comma-delimited file. For example, Open Plan will not import 2/5/04. It will, however, correctly interpret 02/05/04. Open Plan correctly interprets year information in either two-digit or four-digit formats.

Note: If you are creating or editing your import data in Excel, create a custom mm/dd/yyyy hh:mm format. Then apply this format to all date columns before saving the import file.

- **DELIMITED** — Specifies a character to be used as a delimiter using the following syntax:

DELIMITED <*character*>

The <*character*> parameter can be any character. In addition, two special characters can also be used:

\t uses a tab as the delimiter
\0 uses the binary zero as a delimiter

> **Note:** The default delimiter for import and export scripts is the comma. This default will be used when the DELIMITED command is not included or when it is included without specifying a character.

- **DURATION_FORMAT** — Defines the format of duration data being exported with the following syntax:

DURATION_FORMAT <*durunit*> [<*durstring*>]

The <*durunit*> parameter indicates if Open Plan should round the duration information to one of the following duration units:

- m — months
- w — weeks
- d — days
- h — hours
- t — minutes

For example, the statement:

DURATION_FORMAT h

results in all exported durations being rounded to the nearest hour.

The optional <*durstring*> parameter allows you to substitute single-character abbreviations for the five duration units supported by Open Plan: months, weeks, days, hours, and minutes. You must enter all five characters for this parameter. Thus, the statement:

DURATION_FORMAT thdwm

It allows you to export durations rounded to the nearest hour using the French abbreviations for weeks and days.

> **Note:** The primary purpose of the parameter **thdwm** is to support foreign language duration formats.

If you want to change the duration units, but do not want Open Plan to round the duration values, set the <*durunit*> parameter to an asterisk as in the following example:

DURATION_FORMAT *thdwm

It is also possible to set the <*durunit*> parameter to an exclamation point (!), which results in Open Plan exporting duration values as zeros rather than blanks. For example, the following statement:

DURATION_FORMAT !

results in the exportation of zero duration values using the default duration abbreviations. By contrast, the statement:

DURATION_FORMAT !thdwm

results in the exportation of zero duration values using the French duration abbreviations.

This command applies to the current table and all tables until the next occurrence of DURATION_FORMAT.

- **FIELD_SPECIAL** — Transfers information not stored in a specific field using the following syntax:

FIELD_SPECIAL <specialfieldname>

The <specialfieldname> can be one of a set of special fields. This set currently includes:

For any table:

- ROW_NUMBER (applies to exports only)
- $PROJECT_NAME$

For the activity table only:

- MPX_PREDS
- $MPX_PREDS_REPLACES$
- MPX_PRED_ROWS
- MPX_TARGET_DATE
- MPX_TARGET_TYPE
- MPX_FIXED
- $BAAN_CODE_NUMBER$
- $BAAN_CODE_VALUE$
- $BAAN_CODE_DESC$

For code tables only:

- $CODE_NUMBER$
- $BAAN_HIERARCHY_PARENT$
- $BAAN_CODE_NUMBER$
- $BAAN_CODE_NAME$
- $BAAN_CODE_VALUE$
- $BAAN_CODE_DESC$

For the resource description table only:

- $BAAN_HIERARCHY_PARENT$
- $MPX_RES_UNIQUEID$
- MPX_RES_LEVEL
- MPX_CAL

Special fields are also used to export constant values to an external application.

Special fields are recognized by being at least two characters long and by the dollar sign ($) sign at the start and finish of the name. Anything else is treated as a literal constant. If you want to output a constant starting with a dollar sign use two dollar signs. Thus, both “$$” and “$” will print as a “$.”

- **FILTER** — Applies a filter to the export data using the following syntax:

FILTER [<filtername>] [<filterdefinition>]

The filter definition should contain no spaces.

The **Export General** feature supports the keyword filter. This keyword can be used to filter Open Plan data when creating an export file to be used when importing the project data into another application.

- **FIXED** — Defines the file as fixed-format using one of the following syntaxes:

FIXED <fieldwidth>

or

FIXED <margin> <fieldwidth>

If only one parameter is present, <*fieldwidth*> represents the number of characters needed to represent the data record type for each record. If both the <*margin*> and the <*fieldwidth*> parameters are present, the first parameter indicates how many characters to ignore before identifying the data record type.

The FIXED command must precede the first TABLE command. In the absence of the FIXED command, Open Plan assumes the file to be comma-delimited.

- **HEADER** — Defines the record type of the header record, using the following syntax:

HEADER <headerrecordtype>

Header records indicate the fields to be imported or exported and their order, using identifiers that match the <*header*> parameter of the FIELD command.

- **LINK** — Links the current table to a previously defined table using the following syntax:

LINK [<*fieldname*>]

Typically, links are based upon the optional <*fieldname*> parameter. For example, the following statement in the definition of an assignment table:

LINK ID

will link the assignment table to the most recently defined activity table. Linking means that the assignment records appear directly after the activity record to which they apply.

A special form of LINK without a field name will link to the most recently defined table of the same type, on a record-by-record basis. This allows more than one record type to be generated from a single record in the table.

A table may be linked to a linked table, creating a chain of any desired length.

- **LITERAL_END —** Terminates literal data introduced by the LITERAL_HEADER or LITERAL_FOOTER commands using the following syntax:

LITERAL_END

- **LITERAL_FOOTER —** Introduces a set of lines to be transferred at the end of the output for the current table using the following syntax:

LITERAL_FOOTER

The effect of this command is terminated by a LITERAL_END command.

- **LITERAL_HEADER —** Introduces a set of lines to be transferred at the beginning of the output for the current table using the following syntax:

LITERAL_HEADER

The effect of this command is terminated by a LITERAL_END command.

> **Note:** The three LITERAL commands apply only to the exporting of data. Data between the LITERAL_HEADER command and the LITERAL_END command is transferred to the export file at the start of the data for the table. Data between the LITERAL_FOOTER command and the LITERAL_END command is transferred to the export file at the end of the data for the table.

- **MPX_CALENDAR_DEFINITION** — Processes calendar definitions in the format required by .mpx record type 20 using the following syntax:

MPX_CALENDAR_DEFINITION

This command is used only in conjunction with the CLH table type.

> **Note:** For information about the CLH table type, see Table Types.

- **MPX_CALENDAR_HOURS** — Processes the definition of standard working hours in the format required by .mpx record type 25 using the following syntax:

MPX_CALENDAR_HOURS

This command is used only in conjunction with the CLH table type.

> **Attention:** For information about the CLH table type, see Table Types.

- **MPX_CALENDAR_EXCEPTIONS** — Processes calendar exceptions (for example, holidays) in the format required by .mpx record type 26 using the following syntax:

MPX_CALENDAR_EXCEPTIONS

This command is used only in conjunction with the CLH table type.

> **Attention:** For information about the CLH table type, see Table Types.

- **REM** — Allows you to add comment lines to a script using the following syntax:

REM <comment>

When executing a script, Open Plan ignores all lines starting with REM. REM cannot be used in any position other than the start of a line.

- **SKIP** — Defines the number of items to be skipped over before the next defined field using the following syntax:

SKIP <numberoffields>|<numberofcharacters>

If the table is delimited, <*numberoffields*> defines the number of fields to skip over. If the table is fixed-format, <*numberofcharacters*> defines the number of characters to skip over.

This command is typically used to skip over unnecessary fields during an import operation.

- **SORT** — Allows you to apply a sort to the export data using a sort name or the following syntax:

[-]<sort_field1>[,[-]<sort_field2>[,…

where:

- Square brackets indicate that the contents are optional
- Each of the <*sort_fieldx*> values is either the name of an existing field on the appropriate table or the definition of a calculated field for that table
- Negative signs indicate a descending rather than an ascending sort

The sort definition should contain no spaces. For example, the following expression:

SORT C1.codedesc,(ssdate-esdate)

would sort descending on the difference between the scheduled start date and the early start date within the code 1 description. The parentheses in this example are included to make the expression easier to read. In actual usage they are optional.

The **Export General** feature supports the keyword sort. This keyword can be used to sort Open Plan data when creating an export file that will be used to import project data into another application.

- **UPDATE** — Specifies that any existing records matching the incoming ones should be updated using the following syntax:

UPDATE

This command applies to import scripts only. Incoming data that matches existing records are automatically updated. If the data does not match any existing records, the new record is added.

This command should be applied to each table that needs to be updated.

- **UPDATE_ONLY —** Specifies that the incoming data should be used only to update existing records with the same key values using the following syntax:

UPDATE_ONLY

This command applies to import scripts only. Incoming data that does not match an existing record is ignored.

This command should be applied to each table that needs to be updated.

> **Attention:** When tables are linked as a parent-child (for example, activities and assignments), if the parent is in UPDATE or UPDATE_ONLY mode and the child is not, all existing children are deleted before adding the new child records. When the LINK command is used to link two instances of the same table, the second and subsequent instances are automatically set into UPDATE_ONLY mode. Notice that the UPDATE command must specify all fields necessary (usually the key fields) to allow the creation of a record.

- **UPDATE_REMAINING —** Specifies that Remaining Quantity on activity Resource Assignments should be updated when Actual Quantities are imported for the Activity Resources and the resource is configured for manual progress by disabling the **Progress Based On Activity Progress** option on the resource.

UPDATE_REMAINING

This command is only supported on the CST table type and applies to import scripts only.

Table Types

A table type is represented by a 3-character table identifier, which corresponds to the extensions used by projects stored in the actual database. Supported table types include tables that can be associated directly or indirectly with a project, plus a number of “pseudo-tables.” These pseudo-tables do not display in Open Plan but are recognized for the purpose of importing and exporting.

Recognized table types include:

Table Type	Description
ACT	Activity details
REL	Activity relationships
ANN	Baseline activities (for baseline 1, use A01)
UNN	Baseline resource usage (for baseline 1, use U01)
ASG	Resource assignments
CST	Resource actuals
USE	Resource usage
RSK	Risk key activities
RES	Resource definitions
AVL	Resource availabilities
RSL	Resource cost escalation (ESC is recognized as a pseudonym for this table type.)
CLH	Calendar header (CAL is recognized as a pseudonym for this table type.)
CLR	Calendar detail
SCA	Code structure association (CDH is recognized as a pseudonym for this table type.)
CNN	Refers to any number of attached code tables.
<*nn*>	Numeric that can be used to define a specific code table.
PRJ	Project directory details

The order of TABLE commands in an export script defines the order in which the data is exported. In import scripts, the order of TABLE commands has no significance, except as already described for linked tables.

Tables can be defined more than once in a script. For example, to specify two sets of data using two different record types from the activity table, use two TABLE commands.

Tables can also be linked (see the discussion of the LINK command) so that data is interleaved. For example, in the .mpx format it is necessary for the assignment records to immediately follow the activity record to which they apply.

Data from normal tables is accessed by the field name, and any field that exists on the table can be accessed. However, data from a pseudo-table is accessed by means of special pseudo-field-names defined for that table type only.

The preferred way of accessing the PRJ table is by using the real field names. However, for backward compatibility we support:

- PROJECT_NAME for DIR_ID
- PATH_NAME for DIR_ID
- CALENDAR_PATH_NAME for CLD_ID
- RESOURCE_PATH_NAME for RDS_ID
- DESCRIP for DESCRIPTION
- MANAGER for OPMANAGER
- COMPANY for OPCOMPANY
- CLIENT for OPCLIENT

The PROJ_NOFCODES field name has no direct representation on the project table but can now be accessed using the “special field.”

$PRJ_NOFCODES$

In order to access this table you need to enter the following syntax:

FIELD_SPECIAL $PRJ_NOFCODES$

Not

FIELD PRJ-NOFCODES

Pseudo-fields currently defined for the CDH pseudo-tables are:

- CODE_NUMBER
- CODE_NAME for COD_ID
- PATH_NAME for COD_ID
- DESCRIP for DESCRIPTION

Note: The CODE_NAME field allows a user to specify a name in place of a code number. A code name would typically be the unqualified name of a code file; although, this is not a requirement if the name matches the CNN and ACT records. You cannot, however, use both CODE_NUMBER and CODE_NAME to import code information.

Code numbers are assigned based on the order in which they are encountered.

Notice that calendar data is normally accessed using the CLH table type, and this can be done (for a default calendar) even when there is no calendar attached to the project. Users who want to read and interpret the raw calendar data in a calendar table should use the CLR table type.

Notice also that the use of the CNN table name will import and export data to or from any number of code files but all with the same record type. It is necessary to use the special field $CODE_NUMBER$ (see the discussion on the FIELD_SPECIAL command) to distinguish different code files. Alternatively, one can select specific code files (using the <*nn*> table type) and relate each file to specific record types.

Note: Although a combination of fields may form a unique identifier for an incoming record, such a combination may not serve as the key field. You must include the key fields for a given table in the import script.

Importing and Exporting using XML

The generic import/export utility now supports XML documents. Each row of a table represents an element while each field represents an attribute. The table itself can be considered as a collection of elements, which in XML terms is just another element.

Note: The default working directory created by Open Plan is the **USER** folder within the Open Plan destination folder. The **USER** folder contains style sheets (*.xls) and gif files Open Plan uses when exporting XML documents. If you have changed your output directory, you will need to copy the *.xsl and *.gif files from the **USER** folder to that output directory.

Keyword ATTRIBUTE

This keyword is synonymous with FIELD and is provided to support the normal XML nomenclature and to distinguish it from the ELEMENT keyword defined below.

Keyword ELEMENT

This keyword is similar to the FIELD keyword (see below) except that it causes the field to be transferred as an XML element rather than as an attribute. For example, if the ELEMENT keyword had been used for the activity ID field instead of the FIELD keyword in the examples in this section, it would look like this:

<ACTIVITY <ID>1</ID>

DESCRIPTION="Environment Management System"

ESDATE="02Jan97" />

Note: Regardless of the order of ELEMENT and FIELD definitions, the elements are always output first. For input, the elements are always expected first.

Keyword HIERARCHICAL

The keyword HIERARCHICAL used within a table definition causes export to be nested using the hierarchical key of that table. (It is ignored if the table is not hierarchical and thus works only for activity, resource description, and code tables.) Any linked tables are also nested, as can be seen in the following example script:

```
EXPORT XML XML Export (With Style Sheet)
DATE_FORMAT %D%A%Y
LITERAL_HEADER
<?xml version="1.0" ?>
LITERAL_END
STYLESHEET op1.xsl

TABLE ACT
HIERARCHICAL
XML_TAG ACTIVITIES
XML_TAG ACTIVITY
```

```
FIELD ID ID
FIELD ACT_DESC DESCRIPTION
FIELD ESDATE ESDATE

TABLE REL
XML_TAG PREDECESSORS
XML_TAG PREDECESSOR
LINK SUCC_ID
FIELD PRED_ACT_UID PRED_ACT_UID

TABLE REL
XML_TAG SUCCESSORS
XML_TAG SUCCESSOR
LINK PRED_ACT_UID
FIELD SUCC_ID SUCC_ID
```

Notes

1. “STYLESHEET op1.xsl” causes the following to be included in the output:

   <?xml-stylesheet type="text/xsl" href="C:\Program Files\Deltek\Open Plan Professional\op1.xsl"?>

2. “HIERARCHICAL” causes child activities to be nested within their parents. This is not shown by further indentation, but by the logical structure indicated by the tags. Child <ACTIVITY> tags appear between the <ACTIVITY> tag of the parent and its end tag <\ACTIVITY>.

The following example shows the output produced by the previous script.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href=" C:\Program Files\Deltek\Open Plan Professional\op1.xsl"?>
<ACTIVITIES>
    <ACTIVITY
        ID="1"
        DESCRIPTION="Environment Management System"
        ESDATE="02Jan01" />
    <ACTIVITY
        ID="1.01"
        DESCRIPTION="Requirements Development"
        ESDATE="02Jan01" >
        <SUCCESSORS>
```

```
                    <SUCCESSOR
                        SUCC_ID="1.03" />
                </SUCCESSORS>
            </ACTIVITY>
            <ACTIVITY
                ID="1.01.01"
                DESCRIPTION="Requirements Coordination"
                ESDATE="02Jan01" >
                <SUCCESSORS>
                    <SUCCESSOR
                        SUCC_ID="1.01.02" />
                </SUCCESSORS>
            </ACTIVITY>
            <ACTIVITY
                ID="1.01.02"
                DESCRIPTION="Site Coordination &amp;amp; Design"
                ESDATE="13Feb01" >
                <PREDECESSORS>
                    <PREDECESSOR
                   PRED_ACT_UID="1.01.01" />
                </PREDECESSORS>
            </ACTIVITY>
        … … … etc.
            </ACTIVITY>
            </ACTIVITY>
        </ACTIVITIES>
```

The keyword HIERARCHICAL has no effect on import; although, it is possible to import nested data. For example, it is possible to import the data shown in the above example regardless of whether HIERARCHICAL is specified in the import script. It should be noted, however, that on import, the nesting of data does not in itself imply a hierarchy, which must be explicit in the activity ID.

Keyword STYLESHEET

The keyword STYLESHEET allows a style sheet to be defined, which will format the XML data when it is displayed in a browser. It would normally appear after the LITERAL_END following the XML definition, as shown in the previous example.

The text following the STYLESHEET keyword should either be a file name or a URL. If it is a local file name, indicated by no forward or backward slashes, it is interpreted as appearing in the Open Plan global

directory. (Therefore, if you want to use the current directory either define it explicitly using the fully qualified file name or precede the file name by ".\")

The Keyword STYLESHEET has no effect on import.

Keyword XML_TAG

The keyword XML_TAG defines XML import or export. The keyword can appear up to twice for each table definition. The first defines the tag for the collection, while the second defines the tag for each element in the collection. (If only one is present, it is interpreted as applying to each element.)

The XML_TAG keyword cannot be used in conjunction with any of the following keywords:

- RECORD_TYPE
- FIXED
- HEADER
- MPX_CALENDAR_DEFINITION
- MPX_CALENDAR_HOURS
- MPX_CALENDAR_EXCEPTION

All XML documents must have a root element that contains everything else.

Keyword FIELD

If the XML_TAG keyword has been specified for a particular table, the second parameter on the FIELD statement, if present, contains the XML attribute name. It may be omitted if this is the same as the Open Plan field name.

On export, blank character string attributes are not exported.

On input, it is not necessary for the input file to contain all the defined attributes for every element (for example, some activities could have descriptions and others not), but there has to be at least one attribute specified or no record will be created. (Even the ID can be missing, making use of the new auto-numbering feature, but this is no use if you want to specify relationships.)

Attributes that appear on the input file but are not defined in the script (which incidentally includes any attributes associated with the collection as a whole, that is with the first of two tags defined for a given table) are ignored. There is no log message.

On input, relationships cannot be created prior to both activities. If possible, the relationships should appear after all the activities. However, if predecessors or successors need to be specified as belonging to the activity element, it is necessary to make sure the activities are in a suitable order. (Alternatively, it is possible to specify predecessors and successors for each activity, in which case half of them work and the other half result in messages on the log.)

Keyword LINK

The LINK keyword embeds the table inside the element from the parent table. For example, the set of resource assignments might belong to the activity element. A special form of the LINK statement:

```
LINK PRJ
```

enables the elements in a table to be embedded inside the project element, which would normally be the root element. (All XML documents must have a root element that contains everything else.)

Note: In a valid XML file, all the data must appear within the root tag, which in Open Plan's case would normally be the tag associated with the PRJ table. Therefore, every table apart from the PRJ table must contain a LINK keyword, linking either to the PRJ table or to a previously-defined table.

Keywords LITERAL_HEADER & LITERAL_FOOTER

Literal headers and footers can be defined outside of the scope of a table definition. This is not specific to XML but facilitates the inclusion of the XML declaration and other heading information in an XML output.

Import/Export Considerations

1. The offending data in the log messages may not be on the line being processed and the line being processed may not be at all unique.
2. On input, you can nest top-level tags within other tags (for example, activities within a parent activity) and it works, but this does not itself imply a hierarchy. The hierarchy must be implied in the ID.
3. On export, there is no way to nest activities within other activities.
4. On import, log messages in general may not be as useful as might be desired. In particular, since carriage return/line feed has no significance in an XML file, the line that is displayed as being in error may not be the original cause of the error.

XML Example 1

The following script produces output like that shown on the next page.

```
EXPORT XML XML Export
LITERAL_HEADER
<?xml version="1.0" ?>
LITERAL_END

TABLE PRJ
XML_TAG PROJECT

TABLE ACT
XML_TAG ACTIVITIES
XML_TAG ACTIVITY
LINK PRJ
FIELD ID ID
FIELD ACT_DESC DESCRIPTION

TABLE REL
XML_TAG PREDECESSORS
```

```
XML_TAG PREDECESSOR
LINK SUCC_ID
FIELD PRED_ACT_UID PRED_ACT_UID

TABLE REL
XML_TAG SUCCESSORS
XML_TAG SUCCESSOR
LINK PRED_ACT_UID
FIELD SUCC_ID SUCC_ID
```

Notes

1. A line that introduces the file as an XML file and specifies the version introduces the export file. (This is just a literal string to Open Plan.)
2. The next tag defines the root element PROJECT. (It is not strictly necessary in this case because the root element could be ACTIVITIES, but this would not work if we wanted, for example, to output resource pool information, which would have to be part of the PROJECT element.)
3. In this example, we are exporting both successors and predecessors, which seems redundant. However, for re-importing to Open Plan this is a good idea since it ensures that all relationships are imported regardless of the order of the activities. (Another way to handle this would be not to link the relationships to the tasks, which is another scenario where you would need the PROJECT element as the root.)
4. Clicking on the XML file will invoke your web browser and display the information in a hierarchical format. The format can be greatly enhanced by defining a style sheet, which can be referenced as part of the literal header.

The following example shows the output produced by the previous script.

```
<?xml version="1.0" ?>
<PROJECT>
<ACTIVITIES>
    <ACTIVITY
        ID="1.01.02"
        DESCRIPTION="Site Management" >
        <SUCCESSORS>
            <SUCCESSOR
                SUCC_ID="1.01.01" />
        </SUCCESSORS>
    </ACTIVITY>
    <ACTIVITY
        ID="1.02"
        DESCRIPTION="Coordination Planning" >
```

```
                    <PREDECESSORS>
                        <PREDECESSOR
                            PRED_ACT_UID="1.01" />
                    </PREDECESSORS>
                    <SUCCESSORS>
                        <SUCCESSOR
                            SUCC_ID="1.03" />
                    </SUCCESSORS>
            </ACTIVITY>
            <ACTIVITY
                    ID="1.02.01"
                    DESCRIPTION="Engineering Field Testing" >
                    <SUCCESSORS>
                        <SUCCESSOR
                            SUCC_ID="1.02.02" />
                    </SUCCESSORS>
            </ACTIVITY>
    .....
    etc.
    .....
    </ACTIVITIES>
    </PROJECT>
```

XML Example 2

The following script produces output like that shown on the next page.

```
LITERAL_HEADER
<?xml version="1.0" ?>
LITERAL_END

TABLE PRJ
XML_TAG PROJECT

TABLE ACT
XML_TAG ACTIVITIES
```

```
XML_TAG ACTIVITY
LINK PRJ
FIELD ID ID
FIELD ACT_DESC DESCRIPTION

TABLE REL
XML_TAG PREDECESSORS
XML_TAG PREDECESSOR
LINK PRJ
FIELD SUCC_ID SUCC_ID
FIELD PRED_ACT_UID PRED_ACT_UID
```

Notes

1. Note that this time we definitely need the Project tag, which was optional in the previous example. It is this which makes the collection of activities and the collection of relationships all part of a single root.
2. Note the use of & for an ampersand in the original description ="Req. Coordination & Planning" of activity 1.01.01. XML uses certain characters for specific purposes, and represents these characters by a sequence beginning with an ampersand and ending in a semicolon. The ampersand itself is one of these characters. Open Plan translates these automatically. The complete list of special characters recognized by Open Plan is:

```
&amp; for &
&lt; for <
&gt; for >
&quote; "
&apos; for ‘
```

```
<?xml version="1.0" ?>
<PROJECT>
    <ACTIVITIES>
        <ACTIVITY
        ID="1"
        DESCRIPTION="Environmental Management System" />
        <ACTIVITY
        ID="1.01"
        DESCRIPTION="Requirements Development"/>
        <ACTIVITY
        ID="1.01.01"
```

```
                DESCRIPTION="Req. Coordination &amp;amp; Planning" />
                <ACTIVITY
                ID="1.03.02"
                DESCRIPTION="Geochemical Properties" />
                <ACTIVITY
                ID="1.03.03"
                DESCRIPTION="Spent Fuel Modeling" />
                …… etc.

        </ACTIVITIES>
        <PREDECESSORS>
                <PREDECESSOR
                SUCC_ID="1.01.01"
                PRED_ACT_UID="1.01.02" />
                <PREDECESSOR
                SUCC_ID="1.02"
                PRED_ACT_UID="1.01" />
                <PREDECESSOR
                SUCC_ID="1.02.02"
                PRED_ACT_UID="1.02.01" />
                <PREDECESSOR
                SUCC_ID="1.03"
                PRED_ACT_UID="1.02" />
                <PREDECESSOR
                SUCC_ID="1.03.02"
                PRED_ACT_UID="1.03.01" />

                …… etc.

        </PREDECESSORS>
</PROJECT>
```

Field Mappings Between MSP and Open Plan

Many of the standard MSP fields are mapped to Open Plan fields through the Microsoft Project Import/Export utility in Open Plan.

MSP to Open Plan Field Mapping

The following table maps fields from MSP to fields in Open Plan during import:

MSP	Open Plan
Unique ID	MSPUNIQUEID or PWAGUID
Outline Number	Activity ID (ACT_ID)
Actual Start	Actual Start (ASDATE)
Actual Finish	Actual Finish (AFDATE)
Physical % Complete	Physical % Complete (PPC)
Early Start	Early Start (ESDATE)[1]
Early Finish	Early Finish (EFDATE) [1]
Baseline Start	Baseline Start (BSDATE) [1]
Baseline Finish	Baseline Finish (BFDATE) [1]
Late Start	Late Start (LSDATE) [1]
Late Finish	Late Finish (LFDATE) [1]
Finish Slack	Finish Total Float (FINTOTFLT) [1]
Free Float	Free Float (FREEFLOAT) [1]
Total Float	Total Float (TOTALFLOAT) [1]
Name	Description (DESCRIPTION) [1]
Notes	[Note Category]Default Note Category (< Default >)[1, 8]
Critical	Critical (CRITICAL) – “Critical” or “Not Critical” [2]
Earned Value Method	Earned Value Technique (EVT) [3]
Priority	Priority (PRIORITY) – Range: 0 – 999
Duration	Original Duration (ORIG_DUR)
Task Calendar	Calendar ID (CLH_ID) [1]

MSP	Open Plan
Constraint Type	Different fields, according to context[4]
Constraint Date	Different fields, according to context[4]
% Complete	Progress Type/Progress Value (PROGTYPE/PROGVALUE) [5]
Start	Scheduled Start (SSDATE) [6]
Finish	Scheduled Finish (SFDATE) [6]
WBS	<Prompt assigned to code file> (C1) [7]
Milestone	Activity Type (ACT_TYPE) is set to "Finish Milestone"

[1] This is not set during a status-only update.

[2] MSP has a critical FLAG (which value can either be **Yes** or **No**). Open Plan has an enumeration with four values: **Not Critical**, **Critical**, **Most Critical**, and **Controlling Critical**. Only the first two values (**Not Critical** and **Critical**) are used for MSP.

[3] If an existing activity is being updated and the EVT in Open Plan is set to **steps**, the Earned Value Method in MSP is ignored. If a new activity is being created and the default EVT type is set to **steps**, the incoming MSP value is ignored. Otherwise, only two incoming MSP Earned Value Method values are recognized: **% Complete** becomes the EVT **% Complete** and **Physical % Complete** becomes the EVT **Resource % Complete**.

[4] Normally, the Activity Type (**ACT_TYPE**) field is set to either **ASAP** (for child activities) or **Subproject** (for parent activities). If the MSP constraint type is **as late as possible**, the Activity Type is set to **ALAP**. For other constraint types and dates, the Open Plan Target Start Type (**TARGSTYPE**), Target Finish Type (**TARGFTYPE**), Target Start (**TSDATE**), and Target Finish (**TFDATE**) fields are used. For the corresponding field mappings, see the MSP Constraint Type to Open Plan Target Type and Date Field Mapping section.

[5] If the MSP **% Complete** field is set to **100%**, the Open Plan Progress Value (**PROGVALUE**) is set to **100%**, and the Progress Type (**PROGTYPE**) field is set to **Complete**.

If the MSP **% Complete** field is set between **1** and 99, the Open Plan Progress Value (**PROGVALUE**) is set to that percentage, and the Progress Type (**PROGTYPE**) field is set to **Percent Complete**.

If the MSP **% Complete** field is set to **0**, the Open Plan Progress Value (**PROGVALUE**) is set to **0%**, and the Progress Type (**PROGTYPE**) field is set to **Planned**.

[6] Scheduled fields are set only if resources are being imported (and are present), and the user is not doing a status-only update.

[7] If the **Store WBS to Code 1** option is selected in Open Plan's Microsoft Project Import Wizard, a code file is created at index 1, and MSP **WBS** values is assigned to the **C1** field in the activity table. Otherwise, the **WBS** field can be assigned to Open Plan code fields and user-defined fields by creating a field mapping during import.

[8] Using field mapping, the user may override the default behavior of importing notes (assignment to the **< Default >** category), and assign notes to a different category.

MSP Constraint Type to Open Plan Target Type and Date Field Mapping

The following table maps fields from the MSP constraint type to the Open Plan Target Type and Date fields:

MSP Constraint Type	*Open Plan Target Type and Date*
Finish No Earlier Than	TARGFTYPE = Not Earlier Than TFDATE = MSP Constraint Date
Finish No Later Than	TARGFTYPE = Not Later Than TFDATE = MSP Constraint Date
Must Finish On	TARGFTYPE = On Target TFDATE = MSP Constraint Date
Start No Earlier Than	TARGSTYPE = Not Earlier Than TSDATE = MSP Constraint Date
Start No Later Than	TARGSTYPE = Not Later Than TSDATE = MSP Constraint Date
Must Start On	TARGSTYPE = On Target TSDATE = MSP Constraint Date

Open Plan to MSP Field Mapping

The following table maps fields from Open Plan to MSP during export:

Open Plan	MSP
MSPUNIQUEID/PWAGUID[1]	Unique ID
(Depends on Options)[2]	WBS
Description (DESCRIPTION)	Name
Calendar ID (CLH_ID)[3]	Task Calendar
Original Duration (ORIG_DUR)	Duration
Outline_Level (calculated value)[4]	Outline Level
N/A	“Ignore Resource Calendar” is set unconditionally to “Yes” for new tasks
N/A	“Type” is set unconditionally to “Fixed Duration” for new tasks
Activity Type (ACT_TYPE) = “Start Milestone” or “Finish Milestone”	“Milestone” is set to “Yes”

Open Plan	MSP
N/A	“Effort Driven” is unconditionally set to “No” for new tasks
Activity Type (ACT_TYPE)	“Earned Value Method” is set to “Physical % Complete” if activity type = “ASAP”, “Start Milestone”, “Finish Milestone”, “Discontinuous”, “Effort Driven”, or “Foreign Activity”
(Various Conditions) MSPRIF mode only[5]	Deadline
Priority (PRIORITY)	Priority
Physical % Complete (PPC)	Physical % Complete
Progress Type (PROGTYPE) and Progress Value (PROGVALUE) Non-MSPRIF mode only[6]	% Complete
Target Start (TSDATE), Target Start Type (TARGSTYPE), Target Finish (TFDATE), and Target Finish Type (TARGFTYPE)[7]	Constraint Type and Constraint Date
Actual Start (ASDATE)	Actual Start
Actual Finish (AFDATE)	Actual Finish
Non-MSPRIF mode only[8]	Actual Duration
Non-MSPRIF mode only[9]	Remaining Duration
Early Start (ESDATE)/ Early Finish (EFDATE)	Not directly set[10]
Baseline Start (BSDATE)	Baseline Start
Baseline Finish (BFDATE)	Baseline Finish

[1] The **Unique ID** field stored in Open Plan is used only when performing an update export to link tasks that exist in both the Open Plan and MSP projects. A unique ID is generated for new tasks that were created in Open Plan when they are added to the MSP project (the newly generated unique ID is written back to the Open Plan project during export).

[2] If the user checks the **Update MS Project WBS column** option in the Open Plan Microsoft Project Export Wizard, the **WBS** field in MSP is normally mapped to either the activity ID or code 1 (**C1**) field in the exported project, depending on the user’s selection. It is possible to assign a different code or user field to the MSP **WBS** field, however, using the Field Mapping dialog box that overrides the Wizard setting. If none of the preceding conditions has occurred, Open Plan maps the sequential index of the activity being processed to the MSP WBS.

[3] The MSP Task Calendar is set only when a new project is being created or when the user is performing a **FULL** update.

[4] The **Outline_level** field is a field that Open Plan calculates so that MSP places child tasks under the correct parent task. Open Plan determines this value by counting the number of hierarchical separators (".") in an activity ID.

[5] Two preliminary conditions must be true in order for the **Deadline** field to be set in MSP. First, the _USE_MSP_DEADLINES user preference must either be unset (which it is by default) or not set explicitly to **0**. Second, the export must be performed in **MSPRIF** mode. If those conditions are met, set the **Deadline** field when an activity has a target finish type = **Not Less Than** and a valid target finish date (the deadline is set to the target finish date).

[6] If MSPRIF mode is NOT active, set the MSP **% Complete** field based mainly on the Progress Type and the following rules:

Open Plan Progress Type	*MSP % Complete Calculation*
Planned	% Complete is calculated by using a custom formula to determine whether any progress has occurred.
Percent Complete	Progress Value (**PROGVALUE**) field is used.
Complete	Set to **100%**.

[7] Open Plan target dates are not transferred to MSP in all situations. When they are, the **Constraint Type** and **Constraint Date** fields are used. The rules are very complicated and based on a number of internal calculations. These rules evolved over time to try to keep the exported dates as close as possible between the two products. When constraints are applied, however, an informational message displays in the log file.

[8] In non-MSPRIF mode, set the **MSP Actual Duration** field to the Progress Value (**PROGVALUE**) value if there is an activity with a Progress Type (**PROGTYPE**) = **Elapsed Duration** provided that you are not exporting directly to a PWA (web) database. In the latter case, the field is not set (likely because you are not allowed to).

[9] In non-MSPRIF mode, set the **MSP Remaining Duration** field to the Progress Value (**PROGVALUE**) value if there is an activity with a Progress Type (**PROGTYPE**) = **Remaining Duration**.

[10] Do not explicitly set start or finish dates in MSP when creating or updating activities. MSP does this during its calculation (time analysis) phase based on the durations you have given it and other factors (such as relationships, assignments, and target dates.). After the calculation is performed, if you are in MSPRIF mode, compare the start and finish dates for each activity in MSP with the original dates recorded in Open Plan. Constraints (such as "must start on" and "finish no later than") are applied to MSP task dates that deviate from the Open Plan dates so that the MSP schedule comes as close as possible to mirroring the Open Plan schedule.

Extensibility

Adding Utilities to the Open Plan User Interface

The AddIns.dat file is a script that lets you add custom tools to the Open Plan **Add-Ins** tab. These tools enable you to launch up to 30 external applications from within Open Plan. To add options, select **Edit AddIns.DAT** on the **Add-Ins** tab.

An AddIns.dat file may be located in two folders:

- Modifying the copy of the AddIns.dat file in the same folder as the Open Plan executable file creates tab items accessible to all users of the same installation of Open Plan. This copy of the AddIns.dat file is installed by default.
- Modifying the copy of the AddIns.dat file located in your Open Plan **User** folder creates tab items that are accessible only to you. While this copy of the file is not installed by default, you can create it by copying the AddIns.dat file located in the Open Plan executable folder and pasting it into your **User** folder.

> **Note:** The **Edit AddIns.Dat** tool on the **Add-Ins** tab will open the AddIns.dat file in a text editor.

The tools are composed of commands in strings within the AddIns.dat file. The strings contain a description, location of the file, and parameters. The following format is used when creating custom tools:

Tool1...

Location: AddIns.dat

Type: String

Description: A ribbon group name, application description, filename, and optional parameters – formatted as follows:

```
Tool1=Group:Label;Drive:\Path\Filename %A %P %M
```

The Group is used to organize similar Add Ins in the ribbon menu.

The `Label` represents the tab text for the application.

Any number of the following automatic parameters can be passed to the application. Open Plan supplies these parameters on the command line when launching the application.

Parameter	Description
%P	The currently selected project (View or Open Plan Explorer)
%S	The currently selected project in Open Plan Explorer
%A	The currently selected Activity ID
%R	The currently selected Resource file (View or Open Plan Explorer)
%r	The currently selected resource

Parameter	Description
%b	The currently selected code
%B#	Passes the name of the code file in the position # in the active open project.
%C	The currently selected Calendar name (View or Open Plan Explorer)
%c	The currently selected calendar
%U	The currently logged in user's ID
%W	The encrypted password for the current user
%V	The current view
%E	Launch the file/link in an external Internet browser window.
%I	Allows Internet browser to open within an Open Plan window. If the %I is specified, Open Plan formats the other parameters as a URL query string (for example, %P and %A will be passed as ?P=ProjectName&A=ActivityId). **Note:** If data specific parameters are used, Open Plan will reload the page that was initially loaded in the internal window with an updated query string each time the selected value (project, activity ID, resource, etc.) changes.
%T	When %I is used, tiles the internal browser window.
%X	Maximizes either an internal or external browser window.
%D	Instructs Open Plan to execute the WebWindow in a docked window. The position of the docked window can be further specified by using parameters: DOCKSIDE=[TOP, BOTTOM, LEFT, or RIGHT] DOCKLENGTH=[Length or width specified in pixels]
%Z	Information following this parameter is to be passed as a web parameter. **Note:** There must not be a space between the %Z and the parameter being passed. For example: ▪ Good: "%Zpage=22" 1. BAD: "%Z page=22"
%M	Multiple instances of the add-in may be run concurrently.
<SYSDIR>	Open Plan System Directory: Deprecated Syntax
<USERDIR>	Open Plan User Directory: Deprecated Syntax

Parameter	Description
%SYSTEMDIR%	Open Plan System Director (Location of OPP30.EXE)
%USERDIR%	Open Plan User Directory (Location of Open Plan user files)
%APPLICATIONDIR%	Open Plan Application Directory (Location of Config.dat)
%LOGDIR%	Open Plan Log Directory (Location of Open Plan log files)
%WINUSERNAME%	Windows User Name
%OPPUSERNAME%	Open Plan User Name
%OPPUSEREMAIL%	Open Plan user Email (From the Open Plan User account)

Note: The macros <USERDIR> and <SYSDIR> may be used to shorten the path names given for the executables or documents.

Once you have saved an **Add-Ins** tool, click **Reload Add-Ins** on the **Add-Ins** tab to refresh the tab. You can execute the new tool by either:

- Clicking it on the **Add-Ins** tab
- Pressing the keyboard shortcut if one has been defined for the item

Examples:

Tool1= System:Renumber Activities;%SYSTEMDIR%\Sample Tools\vbapps\Renumber\actrenumber.exe %P

Tool2= System:Deltek;www.deltek.com/ %I

Tool3=System:Edit AddIns.Dat;notepad.exe %SYSTEMDIR%\addins.dat

Note: The copy of the AddIns.dat file opened depends on whether or not the item is to be accessible to all users of the same data source or to an individual user. If the custom item is to be accessible to all users of the same data source (shared items), the copy of the file should be stored in the same folder as the Open Plan executable file. If the custom item is to be accessible only to one particular user (user specific items), the file should be stored in that user's Open Plan **User** folder. Tools in the user's file will be added to the end of the list of shared tools and their positions will be moved down automatically. For example, if the last tool in the shared file is Tool10, then Tool1 from the user file will become the 11th tool.

Defining Calculated Fields

User-defined calculated fields allow you to calculate and display data not stored in the standard versions of the project database tables. With calculated fields, you can extend the flexibility of any view by displaying data that is the result of a custom calculation.

After you have defined a calculated field, you can select that field as a column in a spreadsheet view or display it in an activity box just as you can any standard field. You can also include calculated fields in custom filter and sort expressions.

The expression for a global edit uses the same elements (constants, field names, and functions) as calculated field definitions.

The Elements of Calculated Field Expressions

An expression defining a calculated field can include the following elements:

- Constants
- Field names
- Functions
- Other calculated fields
- Mathematical operators
- Character operators
- Duration operators
- Relational operators
- Logical operators
- User-Defined Variables

This section discusses each of the elements of a calculated field expression with the exception of functions, which are discussed in the following section.

Constants

You can include text, date, numeric, and logical constants in a calculated field expression according to the following guidelines:

Constant	Description
Text	All text must be enclosed in either single or double quotes. For example: ▪ “Programmers” ▪ ‘Phase I’
Dates	Dates must be enclosed in curly brackets ({ and }) and can be expressed in any valid date format. For example: ▪ {01JAN01} ▪ {12/01/01}
Durations	Durations must be enclosed between pipe characters (|) and can be expressed in any valid duration format. For example: ▪ |4h| ▪ |3.5d|

Constant	Description
Numerics	Numeric constants can include any positive or negative value and can appear with or without decimal places. For example: ▪ 1 ▪ 320000 ▪ 12.1 ▪ –123.78
Logical Values	Valid logical constants are as follows: ▪ [TRUE] ▪ [FALSE]
Enumerated Values	Values for enumerated fields (that is, fields with a limited number of valid values) must be enclosed between square brackets ([and]). For example, an expression can include references to any of the possible values for the Act_type field: ▪ [ASAP] ▪ [ALAP] ▪ [Start Milestone] ▪ [Finish Milestone] ▪ [Discontinuous] ▪ [Subproject] ▪ [Hammock] ▪ [Effort Driven] ▪ [External Subproject] You can include text, date, numeric, and logical constants in a calculated field or global edit according to the following guidelines: Open Plan recognizes the following enumeration types: ▪ ACTS — Activity Status ▪ ACTT — Activity Type ▪ BOOL — Boolean ▪ CRIT — Critical Flag ▪ CURV — Curve ▪ DIST— Risk ▪ EVTE — Earned Value Technique ▪ LOGI — Activity Logic Flag ▪ PRJS — Project Status ▪ PROG — Progress

Constant	Description
	▪ RELT — Relationship type ▪ RESC — Resource class ▪ RSCL — Resource scheduling type ▪ TARG — Target type

When including a field in an expression for a calculated field, you must enter the name of the field and not the descriptive name that is displayed in spreadsheet column headings. For example, if you want to define a calculated field expression that references the field containing early start dates, you must identify the field as ESDATE, not Early Start.

If you need to refer to fields stored in tables other than the primary data table, use the name of the linking field, followed by a period (.) and the field name in the linked table. For example:

- **C1.DESCRIPTION** — the description for a code stored in the C1 field.
- **RES_ID.DESCRIPTION** — the description for a resource ID stored in the Res_ID field.

Note: If you display the list of available tables and fields by clicking **Fields** on the **Calculated Field Expression** dialog box, linking fields are distinguished by a double chevron (») next to the field name.

Other Calculated Fields

Expressions defining calculated fields can include references to previously defined calculated fields.

Warning: A calculated field cannot include a reference to itself.

Mathematical Operators

You can include the following mathematical operators in a calculated field expression:

- Add (+)
- Subtract (-)
- Multiply (*)
- Divide (/)
- Group (( ))
- Exponentiate (^ or **)

Expressions containing mathematical operators are evaluated according to the normal rules of precedence:

- Grouping operations are performed before multiplication and division operations.
- Multiplication and division operations are performed before addition and subtraction operations.

Character Operators

You can include the following character operators in a calculated field expression:

- Concatenate (+)
- Is contained in ($)

The **Is contained in** operator returns a logical result.

Duration Operators

You can include the following duration operators in a calculated field expression:

- Add (+)
- Subtract (-)
- Multiply (*)
- Divide (/)

You can use these operators in expressions involving durations as follows:

Operation	Result
Duration + Duration	Duration
Duration + Date	Date
Duration – Duration	Duration
Date – Duration	Date
Duration / Duration	Decimal
Duration / Number	Duration
Duration * Number	Duration

Relational Operators

Expressions for calculated fields can include the following relational operators:

- Equal to (=)
- Not equal to (<>)
- Greater than (>)
- Greater than or equal to (>=)
- Less than (<)
- Less than or equal to (<=)
- Contained in ($)

Logical Operators

You can include the following logical operators in a calculated field expression:

- AND

- NOT
- OR
- AND NOT
- Group (())

User-Defined Variables

When Open Plan parses a calculated field (CF) expression, it optimizes the expression internally by looking for multiple occurrences of a subexpression and replacing these occurrences with an internal variable. For example, let's say you have defined another CF called "TimeNowPlusFive":

Timenow() + |5d|

You then use this CF in another CF:

IIF(EFDATE > TimeNowPlusFive, TimeNowPlusFive, EFDATE)

Internally, Open Plan will evaluate the TimeNowPlusFive CF only once for a particular cell, and use the result of this evaluation each time the same subexpression is encountered within the CF. In order for Open Plan to perform this optimization, each occurrence of the subexpression must be identical in terms of capitalization and spacing (for example, "EFDATE > ASDATE" is not the same as "efdate > asdate"). This optimization is performed automatically and assures efficient evaluation provided that the user has been consistent as explained above in the use of subexpressions that occur multiple times.

Open Plan also provides a second, more pro-active way for the user to write a CF expression by using variables. Two advantages of using this second option are that 1) the expression becomes much more readable and easier to maintain, since redefining the variable needs to be done in only one place, and 2) the parsing of a very complicated expression that contains variables takes less processing time than the same expression using repeated subexpressions.

These are the rules for using variables within a CF:

- Variables are defined above the main CF expression and are contained within a BEGIN VARIABLES/END VARIABLES block.
- Each variable definition must be on its own line.
- The result type of a variable definition (for example, character, duration, integer, decimal, date, finish date, and logical) is determined automatically by Open Plan.
- Variable names should not contain spaces or be identical to field names, function names, or other items used within Open Plan where ambiguity may occur. For example, a variable named "123" could be confused with a numeric value. "act_id" would be confused with the Open Plan field of the same name.
- Variables are not case sensitive.
- The format for a variable definition is: <variable name> = <variable expression>.
- The definition of a variable expression may reference other variables, but referenced variables must have been previously defined.

Here is an example of a CF containing a variable block. Note that the previously defined variable X is included in the definition of variable Y:

BEGIN VARIABLES

X = DATEDIFFERENCE(ASDATE, TIMENOW())

Y = IIF(x<=|2d|, IIF(x<=|1d|, 2, 1), -1)

END VARIABLES

IIF(Y>0, “OK”, “Warning”)

Calculated Field Functions Reference

Open Plan provides a number of functions that can appear in calculated field expressions. This section lists the available functions followed by a detailed description of each function.

ABS()
BASELINE_FIELD()
CDOW()
CMONTH()
CTOD()
DATE()
DATEADD()
DATEDIFFERENCE()
DATEFORMAT()
DAY()
DOW()
DURATION()
EVAL()
FAIL_EVALUATE()
FISCALPERIOD()
FORMAT_HEADING_ITEM()
GET_ASSGNS()
GET_CHILDREN()
GET_COSTS()
GET_FIELD()
GET_FIRST_RECORD_ IN_SUMMARY()
GET_NOTE()
GET_PREDS()
GET_RELATED()
GET_RISKS()
GET_SUCCS()
GET_USAGES()
GO_MONTH()
HAS_NOTE()
IIF()
INLIST()
INSTR()
LEFT()
LEN()
LEVEL()
LOCAL()
LOWER()
LTRIM()
MAX()
MID()
MIN()
MONTH()
NEWLINE()
NUMBER_FORMAT()
OCCURS()
PARENT()
RECORD_NUMBER()
RIGHT()
ROUND()
SPACE()
SQRT()
STR()
STRTRAN()
STUFF()
SUBSTR()
TIMENOW()
TRIM()
UPPER()
USER_ID()
VAL()
YEAR()

ABS()

Purpose	Returns the absolute value of a numeric variable.
Data type	Decimal or integer
Syntax	ABS(<value>)
Where	<value> is a numeric variable
Example	If y contains the value [4] or [-4], the statement:

ABS(y)

returns the value [4].

BASELINE_FIELD()

Purpose	Returns a field from the baseline directory table (OPP_BAS) record associated with the first, second, or third selected project baseline. Note that this function may be used with any table whose records can be associated with a project. This includes the Project Directory, Activity, Relationship, Assignment, Cost, Risk, and Subproject tables.
Data Type	Native type of field requested.
Syntax	BASELINE_FIELD(<SelectedBaselineIndex>, <BaselineFieldName>)
Where	▪ SelectedBaselineIndex is a number between 1 and 3 indicating from which of the selected baseline objects the data is to be retrieved (note that “0” is also accepted, and will return the same results as “1”). ▪ BaselineFieldName is a single or double quote delimited field name indicating the field to be returned. Fields that might be of interest include “BASETYPE” (0 = early, 1 = late, 2 = schedule), “BAS_ID” (baseline name), and “DESCRIPTION."
Example	BASELINE_FIELD(1, "DESCRIPTION") If the baseline selected at index 1 for the current project is “PMB,"the returned string would be “Performance Measurement Baseline."

CDOW()

Purpose	Returns the full day of the week (for example, Tuesday).
Data type	Character
Syntax	CDOW(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming that ESDATE is 07OCT04 (Tuesday), the statement: CDOW(ESDATE) returns the value **Tuesday**.

CMONTH()

Purpose	Returns the full month of the year (for example, October).
Data type	Character
Syntax	CMONTH(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming ESDATE is 07OCT04, the statement: MONTH(ESDATE) returns the value **October**.

CTOD()

Purpose	The Calendar to Date function converts a character string to a value with a DATE data type. This is useful in calculated field expressions where we want to coerce the result of an operation to type DATE.
Data type	Date
Syntax	CTOD (<String Expression>)
Where	<String Expression> is a quotation-mark delimited string with a valid date format.
Example	CTOD(STR(USER_NUM01) + "/" + STR(USER_NUM02) + STR(YEAR(TIMENOW())))
	If USER_NUM01 = 12, USER_NUM02 = 31, and Time Now = 1/1/2006, the date result returned will be {12/31/2006}.

DATE()

Purpose	Returns the current date.
Data type	Date
Syntax	DATE ( )
Example	Assuming the current date is October 4, 2004, the statement:
	DATE( )
	returns **04OCT04**.

DATEADD()

Purpose	Allows you to add a duration to a date and returns a date.
Data type	Date
Syntax	DATEADD (<start date>, <duration>, <calendar >)
Where	<start date> is a date field or user-entered date
	<duration> is a duration field or user-entered value
	<calendar name> is the name of the calendar to be used in the calculation. This parameter is optional; if no calendar is specified, Open Plan performs the calculation using the appropriate calendar as follows:

- If a valid calendar is attached to the project, the function uses the calendar named < Default >.
- If no calendar is attached to the project, a 40-hour week is assumed.

> **Note:** If the calculated field or global edit is based on the activity table, you can also specify the fieldname **CLH_ID** to have Open Plan use the calendar assigned to the specific activity. However, if the global edit is based on the assignment table, specify **ID.CLH_ID** instead of **CLH_ID**.

Example	Assuming that ESDATE is 04OCT04, the statement:
	DATEADD (ESDATE,|2d|, “CAL1”)

returns **05OCT04** based on calculations using the calendar named CAL1.

DATEDIFFERENCE()

Purpose: Returns the difference between two dates as a duration.

Data type: Duration

Syntax: DATEDIFFERENCE (<date1>, <date2>, <calendar>)

Where: <date1> is a date field or user-entered date

<date2> is a date field or user-entered date

<calendar name> is the name of the calendar to be used in the calculation. This parameter is optional; if no calendar is specified, Open Plan uses the appropriate calendar as follows:

- If a valid calendar is attached to the project, the function uses the calendar named < Default >.
- If no calendar is attached to the project, a 40-hour week is assumed.

Note: To perform the calculation using the calendar currently assigned to the activity or resource, set the calendar parameter to Calendar. To determine working days based on a specific calendar in the calendar file, set the calendar parameter to the name of the calendar.

Example: Assuming that ESDATE is 07OCT04 and EFDATE is 08OCT04, the statement:

DATEDIFFERENCE (ESDATE, EFDATE, “CAL1”)

returns **2d** based on calculations using the calendar named CAL1.

DATEFORMAT()

Purpose: Returns a date in a specified format.

Data type: Character

Syntax: DATEFORMAT (<date field>, <format string>)

Where: <date field> is a date field or user-entered date

<format string> is the date format

Note: DATEFORMAT() uses the same date formats as those used when specifying dates on the **Date Scale Preferences** dialog box.

Example: Assuming that ESDATE is October 7, 2004, the statement:

DATEFORMAT (ESDATE, “%D%A%Y”)

returns **07OCT04**.

> **Attention:** For a description of date format strings in Open Plan, see **Views and Reports » Tabs and Dialog Boxes » Date Formats Dialog Box** in the *Deltek Open Plan online help*. This information is also available in Date Formats in Open Plan.

DAY()

Purpose	Returns a numeric value 1 through 31 for the day of the month.
Data type	Integer
Syntax	DAY(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming that ESDATE is 04OCT04, the statement: DAY(ESDATE) returns **4**.

DOW()

Purpose	Returns a numeric value 1 through 7 for the day of the week. The actual number returned depends on settings in the Open Plan .ini file for the first day of the week.
Data type	Integer
Syntax	DOW(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming that ESDATE is 07OCT04 (Tuesday), the statement: DOW(ESDATE) returns **3**.

DURATION()

Purpose	Returns the number of minutes corresponding to a given duration
Data type	Integer
Syntax	DURATION(<Duration Value>)
Where	<Duration Value> is a duration variable or user-entered duration
Example	Assuming an 8-hour workday, the statement: Duration(|2d|) returns **960 (8 hours x 2 days x 60 minutes/hour)**.

EVAL()

Purpose	Evaluates its string expression argument as a dynamic calculated field rather than as a string literal.
Data type	Character

Syntax	EVAL(<exp>)
Where	<exp> evaluates a string expression argument to be calculated as a dynamic calculated field rather than as a string literal.
Example	Assuming that the value of USER_CHR01 is “ESDATE” and USER_CHR02 is “EFDATE” and that ESDATE is 04OCT04 and EFDATE is 06OCT04, the statement: EVAL('CTOD(USER_CHR02) - CTOD(USER_CHR01)') returns a duration of **2d**.

FAIL_EVALUATE()

Purpose	Test for a condition that normally would cause the calculated field to return either a blank value or an invalid result (the latter would keep the remainder of the expression from being evaluated).
Data type	Boolean
Syntax	FAIL_EVALUATE(<String Expression>)
Where	<String Expression> is an expression in string format.
Example	FAIL_EVALUATE("C25") If there is no code file assigned at index 25, this calculated field will return True. Note that if we substituted the expression "C25 IS_EMPTY" as an alternative to using FAIL_EVALUATE, the returned value would be True whether C25 does not exist or C25 exists but is blank.

FISCALPERIOD()

Purpose	This function returns the user-defined label for a specific fiscal reporting period for a specific date.
Data type	Character
Syntax	FISCALPERIOD(<Date>,<Reporting Calendar>)
Where	<Date> is the search date to be located within a fiscal period. < Reporting Calendar> is an optional parameter (defaulting to the project reporting calendar).
Example	FISCALPERIOD(ESDATE) FISCALPERIOD(ESDATE, "REPORTING_CALENDAR_NAME") The function returns the LABEL field from the reporting calendar that includes the supplied date. For example, given the following DATE and LABEL fields: DATE LABEL 01/26/2001 Jan 2001 02/23/2001 Feb 2001 03/23/2001 Mar 2001

FORMAT_HEADING_ITEM()

Purpose	Creates custom grouping heading with summary rows (i.e., the summary type for the grouping row is marked as "Heading").

Data type Character

Syntax FORMAT_HEADING_ITEM(<Expression>,<WidthInCharUnits>, <Wordwrap>,<Summarize>)

Where <Expression> is the character string to be formatted, <WidthInCharUnits> is the width of the expression in character units, <Wordwrap> is the numeric value 0 (no word wrap) or 1 (word wrap active), and <Summarize> is an OPTIONAL argument indicating whether <Expression> (if not a constant) should NOT be summarized (numeric value of 0), or should be summarized (numeric value of 1). The default value is 1.

Example Given the following calculated field expression:

FORMAT_HEADING_ITEM(C2 + " - ," 20,0,0) +
FORMAT_HEADING_ITEM(C2.DESCRIPTION, 40, 1,0) +
FORMAT_HEADING_ITEM(C2.< Default >,20,1,0)

The output in a spreadsheet grouping on C2, using the above calculated field as the Heading Field would look similar to this:

	Activity ID	Activity Desc.		Orig Dur	Early Start
-	1.2.1 -	1.2.1 SYSTEM ENGINEERING	These activities were delayed due to various technical problems.		21Feb05
	1.16.02	Project Final Review		3d	21Feb05

GET_ASSGNS()

Purpose Returns assignment data for an activity.

Data type Character

Syntax GET_ASSGNS(<fieldname1>[|<fieldname2>…])

Where <fieldname1>, <fieldname2> are names of fields in the Resource Assignment table

> **Note:** Because the GET_ASSGNS() function works within a specified data table, it is important that the table portion of the fieldname be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that the resource assignments for the activity are ENG (a level 1 resource pool) and TECH.MARY (a level 2 resource), the statement:

GET_ASSGNS("RES_ID|RES_LEVEL")

It then returns **ENG,1.00;TECH.MARY,2.00**.

GET_CHILDREN()

Purpose	Returns a list of fields from the immediate children of the current hierarchical record (activities, resources, and codes).
Data type	Character
Syntax	GET_CHILDREN(<FieldList>)
Where	FieldList is a pipe-delimited string containing the list of fields to be returned. Fields are separated by commas, and records are separated by semi-colons in the returned data.
Example	GET_CHILDREN("ACT_ID|DESCRIPTION") Given an activity "1.01,"with children, "1.01.01" and "1.01.02,"the returned string would be the following: "1.01.01,First child of 1.01;1.01.02,Second child of 1.01"

GET_COSTS()

Purpose	This function returns cost records for an activity.
Data type	Character
Syntax	GET_COSTS(<fieldname1>[| <fieldname2>…])
Where	<fieldname1>, <fieldname2> are names of fields in the Cost table

> **Note:** Because the GET_COSTS() function works within a specified data table, it is important that the table portion of the fieldname be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example	Assuming that there are two cost records for the activity, that the actual costs are 1,000 and 4,000, and that the actual quantities are 1 and 2, the statement: GET_COSTS("ACWP_CST|ACWP_QTY") It then returns **1000.00,1.00;4000.00,2.00.**

GET_FIELD()

Purpose	This function allows users to display data from other tables or data from other records in the same table.
Data type	Character
Syntax	GET_FIELD (<TableType>, <UniqueID>,<FieldName>)
Where	▪ TableType = "Activity","Resource","Calendar","B1","B2","B3" (Selected Baseline Activity Tables), "ProjDir","CodeDir","ResDir","CalDir".

> **Note:** TableType = "Resource" will pull data from the "Resource Data" table.

- UniqueID = a constant, expression, or field name that identifies a look-up key for the table type. Note that the only acceptable table types are those that have unique identifiers (for example, the activity ID on the activity table, or the project name on the Project Directory).
- FieldName = the field in Table TableType whose value we want returned. Note that the field name MUST be enclosed in quotation marks. Otherwise, the VALUE of the field, not the field name itself, will be passed as the argument.

Example GET_FIELD("C2,"PARENT(C2), "DESCRIPTION")

On the Activity table, if the current Activity has a code 2 value of "1.2.1.3,"the returned string would be "1.2.1. System Engineering."

GET_FIRST_RECORD_IN_SUMMARY()

Purpose This function returns the value for the requested field. For a grouping summary row, it returns the value of the requested field for the first child record.

Data type Same data type as input field

Syntax GET_FIRST_RECORD_IN_SUMMARY (<TableType>, <UniqueID>,<FieldName>)

Where FieldName = the field in Table TableType whose value we want returned. Note that the field name MUST be enclosed in quotation marks. Otherwise, the VALUE of the field, not the field name itself, will be passed as the argument.

Example The purpose of the GET_FIRST_RECORD_IN_SUMMARY function is to allow the user to construct a calculated field result in a summary row that uses a break column value rather than a summary value. For example, given the following filter in a barchart (with a visibility setting of "All levels of Rollup and Detail"), with a spreadsheet pane grouped on TOTALFLOAT:

TOTALFLOAT > |2d|

The filter would evaluate to TRUE for all child rows of groupings where the TOTALFLOAT break-on value is greater than |2d|. However, the filter would evaluate to FALSE on the grouping summary rows since the value of the TOTALFLOAT field on the summary row is undefined (durations cannot be summarized). Even for a numeric or date field (which can be summarized), the filter would have to be written differently to account for a summarized value on a summary row. To work around this problem, the following syntax will give us the result we want:

GET_FIRST_RECORD_IN_SUMMARY("TOTALFLOAT") > |2d|

GET_NOTE()

Purpose Returns an activity, resource, or code note.

Data type Character

Syntax GET_NOTE(<category>)

Where <category> is the name of a note category. This field is optional.

If you leave <category> blank, the function assumes the default category.

Example Assuming that a category <Document> has been set up for an activity table, the statement:

GET_NOTE(“Document”)

returns the Document category note about the activity.

Example Assuming that you want to use the default category for an activity note, the statement:

GET_NOTE(“”)

returns the default category note about the activity.

GET_PREDS()

Purpose Returns predecessor data for an activity.

Data type Character

Syntax GET_PREDS(<fieldname1>[|<fieldname2>…])

Where <fieldname1>, <fieldname2> are names of fields in the Relationship table

> **Note:** Because the GET_PREDS() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that the predecessor to an activity is A100 and that the relationship type is Finish to Start, the statement:

GET_PREDS(“PRED_ACT_ID|REL_TYPE”)

It then returns **A100,Finish to Start**.

GET_RELATED()

Purpose Returns a list of fields for a each member of a child collection related to the main collection. This is basically a more generic version of the more specialized functions GET_PREDS(), GET_SUCCS(), GET_ASSGNS(), etc.

Data type String

Syntax GET_RELATED(<CollectionType>, <FieldList>)

Where < CollectionType > is a quotation-mark delimited identifier.

Main Collection = "Activity"

- “A01”,"A02”,"A03" (Selected Baseline Activity Collection)
- "ASG" (assignment collection)
- "CST" (cost collection)
- "PRD" (predecessor collection)
- "RSK" (risk collection)
- "SUB" (subproject collection)

- "SUC" (successor collection)
- "U01","U02","U03" (Selected Baseline Usage Collection)
- "USE" (usage collection)

Main Collection = "Baseline Activity"

- "SUB" (subproject collection)
- "USE" or "BSU" (baseline usage)

Main Collection = "Resource Data"

- "PSU" (Project Summary Usage)
- "RSL" (Resource Escalations)
- "SKL" (All skill assignments for the given resource record)
- "SKR" (All skill assignments for the given skill record)

Main Collection = "Project"

- "ACT" (Activity)

<FieldList> is a pipe-delimited string containing the list of fields to be returned. Fields are separated by commas, and records are separated by semi-colons in the returned data.

Example — Assuming that we are displaying an activity spreadsheet, and that activity “A” is the current activity, the expression below will return the activity ID and early start fields for the record that matches the activity on the first selected baseline.

GET_RELATED(“A01”,“ACT_ID|ESDATE”) returns “A,12Oct04”

GET_RELATED_COUNT()

Purpose — Returns a record count. The two parameters for the function are the related collection identifier and an optional filter. You use this primarily with activity and baseline activity tables.

Data type — String

Syntax — GET_RELATED_COUNT(RELATED_COLLECTION_IDENTIFIER, <FILTER_ON_RELATED_COLLECTION>)

Where — You can use the function with one of the following related collection identifiers:

Main Collection = "Activity"

- “ASG” (assignment collection)
- “CHILD” (child collection)
- “CST” (cost collection)
- “PRD” (predecessor collection)
- “RSK” (risk collection)
- “STP” (step collection)
- “SUB” (subproject collection)
- “SUC” (successor collection)

- "USE" (usage collection)

Main Collection = "Baseline Activity"

- "CHILD" (child collection)
- "USE" (usage collection)

Example — You might want a count of all children for a subproject activity (GET_RELATED_COUNT("CHILD")), or you may want to get a count of all predecessors for a particular activity with a relationship lag of 0 (GET_RELATED_COUNT(PRD", "REL_LAG = |0|").

GET_RISKS()

Purpose — Returns risk data for an activity.

Data type — Character

Syntax — GET_RISKS(<fieldname1>[| <fieldname2>…]

Where — <fieldname1>, <fieldname2> are names of fields in the Risk table

> **Note:** The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks.
>
> The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example — The statement:

GET_RISKS("ESDATE1")

returns each early start date calculated during risk analysis.

GET_SUCCS()

Purpose — Returns successor data for an activity.

Data type — Character

Syntax — GET_SUCCS(<fieldname1>[|<fieldname2>…])

Where — <fieldname1>, <fieldname2> are names of fields in the Relationship table

> **Note:** Because the GET_SUCCS() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example — Assuming that the successor to an activity is A200 and that the relationship type is Finish to Start, the statement:

GET_SUCCS("SUCC_ACT_ID|REL_TYPE")

It then returns **A200,Finish to Start**.

GET_USAGES()

Purpose	This function returns use records for an activity.
Data type	Character
Syntax	GET_USAGES(<fieldname1>[| < fieldname2>…]
Where	<fieldname1>, <fieldname2> are names of fields in the Use table

> **Note:** Because the GET_USAGES() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example: Assuming that an activity having a single use record shows that 24 units of the resource ENG have been used, the statement:

GET_USAGES(“RES_ID|RES_USED”)

It then returns **ENG,24.**

GO_MONTH()

Purpose	Returns a date that is the specified number of months before or after a specified date.
Data type	Date
Syntax	GO_MONTH(<date>,<integer>)
Where	<date> is a date variable or user-entered date <integer> is a positive or negative integer
Example	Assuming that ESDATE is 04OCT04, the statement: GO_MONTH(ESDATE,-2) Returns **04AUG04**.

HAS_NOTE()

Purpose	Returns a logical value indicating whether or not a note is attached to an object.
Data type	Logical
Syntax	HAS_NOTE(<string>) The string is optional.
Example	Assuming that an activity has a note attached, the statement: HAS_NOTE( ) returns **True**

Example	Assuming that an activity has a note in the category “Scope,” the statement: HAS_NOTE(“Scope”) returns **True**

IIF()

Purpose	Provides a means of defining conditional processing in a report.
Data type	Any
Syntax	IIF(<logicexp>,<iftrue>,<iffalse>)
Where	<logicexp> is a logical expression <iftrue> is any valid expression <iffalse> is any valid expression
Operation	If <logicexp> evaluates to true, the calculated field or global edit is set to the value of <iftrue>; otherwise, the calculated field or global edit is set to the value of <iffalse>.
Example	Assuming that ESDATE is 19JUN01, the statement: IIF(ESDATE>{01JUL01}, “Underway”, “Planned”) returns **Planned**.

INLIST()

Purpose	Returns a logical value indicating whether or not a value is included in a list.
Data type	Logical
Syntax	INLIST(<search>,<value1>, <value2>, ...)
Where	<search> is a variable or constant for which the function searches in the list <value1> is a variable or constant <value2> is a variable or constant **Note:** Open Plan interprets a character argument as a comma-delimited list of items.
Example	Assuming that the early start date month is June, the statement: INLIST(MONTH(ESDATE),1,4,7,10) returns **False**.
Example	Assuming that “CHRIS” is the local portion of a resource code, the statement: INLIST(“CHRIS," GET_ASSGNS(“LOCAL(RES_ID)”)) returns **True** for the activities to which CHRIS is assigned.

INSTR()

Purpose	Returns the position of the first occurrence of one string from within another string. If the string that is sought is not found, this function returns 0.

Data type Integer

Syntax INSTR(<start>,<string1>,<string2>)

Where <start> is an integer that sets the starting position for the search

<string1> is the string expression being searched

<string2> is the string expression being sought

> **Note:** The string expressions are case sensitive.

Example Assuming the following string expression to be searched:

SITE COORDINATION AND DESIGN

And the following string being sought:

COORDINATION

The statement:

INSTR(1,"SITE COORDINATION AND DESIGN","COORDINATION")

returns **6**.

LEFT()

Purpose Returns a specified number of leftmost characters in a string.

Data type Character

Syntax LEFT(<string>,<int>)

Where <string> is the string expression being searched

<int> is the number of characters to be returned

> **Note:** The string expressions are case sensitive.

Example The statement:

LEFT("SITE COORDINATION AND DESIGN", 4)

returns **SITE**.

LEN()

Purpose Returns a numeric value that is the length of a variable or constant.

Data type Integer

Syntax LEN(<data>)

Where <data> is a character, date, or numeric variable, or a character or numeric constant

Example The statement:

LEN("Dig Hole")

returns **8**.

Example	Assuming the DESCRIPTION field is 30 characters long and contains the string “EXCAVATION,” then the statement: LEN(DESCRIPTION) returns **10**. Note that it is irrelevant that the field width is 30 characters.

LEVEL()

Purpose	Returns the hierarchical level of an ID or code.
Data type	Integer
Syntax	LEVEL(<ID>)
Where	<ID> is a character variable or constant
Example	Assuming that a code file is assigned to the project in position C1, then the statement: LEVEL(C1) returns the level of C1 in the code structure.

LOCAL()

Purpose	Returns the local portion of an ID or code (that is, the portion of the ID or code that is not shared by other children of the same parent).
Data type	Character
Syntax	LOCAL(<ID>, <level>)
Where	<ID> is a character variable or constant <level> is optional, specifying the level at which you want the local portion
Example	Assuming that a code file is assigned to the project in position C1, the statement: LOCAL(C1) returns the local (rightmost) portion of C1.

LOWER()

Purpose	Converts alphabetic characters from uppercase to lowercase.
Data type	Character
Syntax	LOWER(<string>)
Where	<string> is a character variable or constant
Example	The statement: LOWER(“Dig Hole”) returns **dig hole**.

LTRIM()

Purpose	Trims leading spaces from a string.
Data type	Character

Syntax LTRIM(<string>)

Where String is a character string

Example Assuming that the activity description is “^^^^^^Administration,” the statement:

LTRIM(DESCRIPTION)

returns Administration.

> **Note:** In this example, each caret symbol (^) is used to represent a space.

MAX()

Purpose Returns the maximum value from a list of variables.

Data type Decimal, integer, character, date, or duration

Syntax MAX(<value1> [,<value 2>, ...])

Where <value1>, <value2> are decimals, integers, characters, or dates

> **Note:** Open Plan interprets a character argument as a comma-delimited list of items.

Example Assuming that value1 is 4, value2 is 25, and value3 is 66, the statement:

MAX(4, 25, 66)

returns **66**.

Example Assuming that USER_CHR01 contains a comma-delimited list of values 5, 1, 8, and 3, the statement:

MAX(USER_CHR01)

returns **8**

MID()

Purpose Returns a specified number of characters in a string. If the number of characters to extract is not specified, this function returns all remaining characters in the string.

Data type Character

Syntax MID(<string>,<start>,<int>)

Where <string> is the string expression being searched

<start> is the beginning location of the characters to be returned

<int> is optional; specifying the number of characters to be returned; if omitted, all remaining characters are returned.

> **Note:** The string expression is case sensitive.

Example The statement:

MID(“SITE COORDINATION AND DESIGN,"5, 12)

returns **COORDINATION**

MIN()

Purpose: Returns the minimum value from a list of variables.

Data type: Decimal, integer, character, date, or duration

Syntax: MIN(<value1>,<value 2>, ...)

Where: <value1>, <value2> are decimals, integers, characters, or dates

> **Note:** Open Plan interprets a character argument as a comma-delimited list of items.

Example: Assuming that value1 is 4, value2 is 25, and value3 is 66, the statement:

MIN(4, 25, 66)

returns **4**.

Example: Assuming that USER_CHR01 contains a comma-delimited list of the values 5, 1, 2, and 3, the statement:

MIN(USER_CHR01)

returns **1**.

MONTH()

Purpose: Returns a 2-digit numeric value for the month.

Data type: Integer

Syntax: MONTH(<date>)

Where: <date> is a date variable or user-entered date

Example: Assuming ESDATE is 04JUL04, the statement:

MONTH(ESDATE)

returns **7**.

NEWLINE()

Purpose: Inserts one or more lines in the calculated field expression.

> **Note:** While this function is available for both calculated fields and global edits, it is used in only calculated fields. The effect of the NEWLINE() function is displayed only in spreadsheet columns where wrapping is enabled.

Data type: Integer

Syntax: NEWLINE(<value>)

Where <value> is an integer representing the number of lines to be added to the expression.

> **Note:** <value> cannot be blank

Example The expression:

“Early:”+DATEFORMAT(ESDATE,”%D%A%Y”)+NEWLINE(1)+”Late: “+DATEFORMAT(LSDATE,”%D%A%Y”)+NEWLINE(1)+”Sched:”+ DATEFORMAT(SSDATE,”%D%A%Y”)

returns a string similar to the following:

Early:12Apr04

Late:16Apr04

Sched:14Apr04

Notice that the NEWLINE() function has forced the Late and Scheduled dates to be moved to the next line.

NUMBER_FORMAT()

Purpose Returns a formatted string from a list of parameters.

Data type Integer

Syntax NUMBER_FORMAT(<integer>, <string>, <integer>)

Where <integer> is the number to be formatted

<string> is a set of characters defining how the integer should be formatted. The following characters can be used in combination with one another (except where restrictions apply):

- “$” - displays the native currency symbol. (This character should not be used in conjunction with “%”)
- .”” – displays a decimal point within the integer. The number of decimal places can be indicated in the optional third argument <integer>.
- ,” ” – displays “thousands” separators.
- “%” – displays the resulting integer as a percentage. (This character should not be used in conjunction with “$”)

<int> is optional; specifying the number of decimal places to be returned. Up to 20 decimal places are allowed. If the number is set to 0, the decimal display is overridden.

> **Note:** If there are any unrecognized characters in the format string or if the format string contains both “$” and “%,"the format string will be returned instead of a value.

A blank <string> is interpreted as a DEFAULT format, but the argument cannot be omitted.

Example The expression:

NUMBER_FORMAT(1000, “$", 2)

Returns **$1,000.00**

OCCURS()

Purpose	Returns the number of times a string occurs within another string.
Data type	Integer
Syntax	OCCURS(<string1>,<string2>)
Where	<string1> is the string expression to be searched <string2> is the substring to be searched

> **Note:** The string expressions are case sensitive.

Example	The statement: OCCURS(“Bright light circuits”,"light”) returns **2**.

PARENT()

Purpose	Returns the parent ID of a specified level for an ID or code.
Data type	Character
Syntax	PARENT(<ID>, <level>)
Where	<ID> is a character variable or constant <level> is optional; specifying the level at which you want the local portion
Example	Assuming that a code file is assigned to the project in position C1, then the statement: PARENT(C1) returns the ID of the immediate parent of C1.

RECORD_NUMBER()

Purpose	Provides a unique number that can be used to identify a particular record.
Data type	Integer
Syntax	RECORD_NUMBER()
Where	<No arguments>
Example	This function is useful in general export scripts. For example: EXPORT csv Sample Export Script TABLE ACT FIELD RECORD_NUMBER() FIELD ACT_ID

RIGHT()

Purpose	Returns a specified number of rightmost characters in a string.

Data type	Character
Syntax	RIGHT(<string>,<int>)
Where	<string> is the string expression to be searched
	<int> is the number of characters to be returned
Example	The statement:
	RIGHT("SITE COORDINATION AND DESIGN," 6)
	returns **DESIGN**.

ROUND()

Purpose	Returns a numeric or duration value rounded to a specified precision.
Data type	Decimal, integer, or duration
Syntax	ROUND(<value>, <precision>, <type>)
Where	<value> is a numeric variable or constant
	<precision> is a positive or negative number. If positive, <precision> represents the number of decimal places. A negative precision represents rounding that occurs to the left of the decimal point, i.e., integer rounding (with the decimal point being at precision 0). A –1 represents units of 10, -2 represents units of 100, and –3 represents units of 1000 (10 to the first, second, and third powers, respectively). With a precision of –1, the last two digits of a given number will be rounded to the nearest multiple of 10.
	You can also modify the precision of a duration using one of the following codes:

- t (minutes)
- h (hours)
- d (days)
- w (weeks)
- m (months)

> **Note:** The code indicating the duration precision is a string and must be placed within quotation marks.

	<type> indicates how Open Plan should round. Valid values are "up" (rounds all decimals up), "down" (rounds all decimals down), or the field can be empty (rounds up values =>.5, and rounds down values <.5) This parameter is optional.
Example	The statement:
	ROUND(458.9738, 2, "up")
	returns **458.98**.
Example	The statement:
	ROUND(orig_dur / 3.0, "d")
	returns the original duration divided by 3.0 and rounded up to the nearest day.
Example	The statement:

ROUND(4589, -2)

returns **4600**.

SPACE()

Purpose	Returns a specified number of space characters.
Data type	Character
Syntax	SPACE(<int>)
Where	<int> is the number of space characters to be returned
Example	The statement
	SPACE(6)
	Returns six space characters.

SQRT()

Purpose	Returns the square root of the absolute value of a numeric value.
Data type	Decimal or integer
Syntax	SQRT(<value>)
Where	<value> is a numeric variable or constant
Example	The statement:
	SQRT(4)
	returns **2**.

STR()

Purpose	Converts a numeric to a character variable.
Data type	Character
Syntax1	STR(<value>,<length>,<decimal>)
Where	<value> is a decimal, integer, or duration
	<length> is a numeric or constant representing the length of the resultant character string including the decimal point and decimal places. This parameter is optional.
	<decimal> is a numeric variable or constant representing the number of decimal places. This parameter is optional.
Syntax2	STR(<date field>)
Where	<date field> is a date field
Example	The statement:
	STR(1.2345,3,1)
	returns **1.2**.

Example — Assuming that ESDATE is October 5, 2004, the statement:

STR(ESDATE)

Returns **05OCT04**

> **Note:** The string returned when using STR() with a date field will use the project's current date format.

STRTRAN()

Purpose — Replaces one substring with another substring and returns the revised string.

Data type — Character

Syntax — STRTRAN(<string1>,<string2>,<string3>)

Where — <string1> is the string expression to be searched

<string2> is the substring to be searched

<string3> is optional; is the replacement string; if omitted, a null string is assumed and <string2> is simply deleted

> **Note:** The string expressions are case sensitive.

Example — The statement:

STRTRAN("SITE MANAGEMENT","MANAGEMENT","SPECIFICATION")

returns **SITE SPECIFICATION**.

STUFF()

Purpose — Replaces a specified number of characters with another substring and returns the revised string.

Data type — Character

Syntax — STUFF(<string1>,<start>,<int>,<string2>

Where — <string1> is the string expression being searched

<start> is the position in the string expression where the replacement should begin

<int> is the number of characters to be removed before a replacement string is inserted; if 0, no characters are removed

<string2> is optional; specifies the replacement string; if omitted, a null string is assumed and the characters specified by <int1> and <int2> are simply deleted

Example — The statement:

STUFF("SITE MANAGEMENT," 6,10, "SPECIFICATION")

returns **SITE SPECIFICATION**.

SUBSTR()

Purpose	Copies a specified part of a character string.
Data type	Character
Syntax	SUBSTR(<string>, <start>, <length>)
Where	<string> is a character string
	<start> is a numeric variable or constant representing the first character to be copied
	<length> is a numeric variable or constant representing the length of the character string to be copied. The parameter is optional.
Example	The statement:
	SUBSTR(“ABCDEF,"3, 2)
	returns **CD**.
Example	Assuming the ID = “P100,"the statement:
	SUBSTR(ID, 1, 2)
	returns **P1**.

TIMENOW()

Purpose	Returns the currently set Time Now date.
Data type	Date
Syntax	TIMENOW( )
Example	Assuming the currently set Time Now date is October 7, 2004, the statement:
	TIMENOW( )
	returns **07OCT04**.

TRIM()

Purpose	Removes trailing blanks from a character variable or constant.
Data type	Character
Syntax	TRIM(<string>)
Where	<string> is a character variable or constant
Example	Assuming that the activity description is “Administration^^^^^^^^^^,"the statement:
	TRIM(DESCRIPTION)+”!!!”
	returns **Administration!!!**.

> **Note:** In this example, each caret symbol (^) is used to represent a space.

UPPER()

Purpose	Converts alphabetic characters from lowercase to uppercase.

Data type	Character
Syntax	UPPER(<string>)
Where	<string> is a character variable or constant
Example	The statement: UPPER(“Dig Hole”) returns **DIG HOLE**.

USER_ID()

Purpose	Returns name of current user.
Data type	Character
Syntax	USER_ID(<string>)
Where	(No arguments)
Example	USER_ID() returns "SYSADMIN."

VAL()

Purpose	Converts a character string to a numeric value. It can also be used to convert durations to minutes.
Data type	Decimal or integer
Syntax	VAL(<string>)
Where	<string> is a character or duration
Operation	If <string> is not a valid numeric value, VAL converts the string starting at the leftmost character until an invalid numeric is encountered. Integer zero is returned if the string does not start with a valid numeric value.
Example	The statement: VAL(“1678”) returns **1678**.
Example	Assuming that the original duration of an activity is 100 days, the statement: VAL(ORIG_DUR) returns **48000** based on conversion preferences set for the project of 8 hours per work day.

YEAR()

Purpose	Returns a 4-digit numeric value for the year.
Data type	Integer
Syntax	YEAR(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming ESDATE is 04JUL04, the statement:

YEAR(ESDATE)

returns **2004**.

Standard Calculated Fields

The following calculated fields are among those supplied with Open Plan:

_Check_My_Preds

Expression	get_preds("_pred_test")
Table	Activity
Data type	Character
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_Check_My_Succs

Expression	get_succs("_succ_test")
Table	Activity
Data type	Character
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_Dimmed_Activities

Expression	(_Check_My_Preds CONTAINS('T') or _Check_My_Succs CONTAINS('T')) and not _My_Network_Filter = [BOOL.T]
Table	Activity
Data type	Logical
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_Has_Preds_In_Other_Sub

Expression	_My_Preds_Parents CONTAINS('T')
Table	Activity
Data type	Logical
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_Has_Succs_In_Other_Sub

Expression	_My_Succs_Parents CONTAINS('T')
Table	Activity
Data type	Logical
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_Is_Activity

Expression	ACT_TYPE <> [ACTT.E] and ACT_TYPE <> [ACTT.P]
Table	Activity
Data type	Logical
Purpose	Returns true if an activity is neither an internal nor an external subproject.

_My_Network_Filter

Expression	Critical_Activity = [BOOL.T]
Table	Activity
Data type	Logical
Purpose	Returns True if an activity's critical field has been set to Controlling Critical, Critical, or Most Critical.

_My_Parent

Expression	str(parent(ACT_ID))
Table	Activity
Data type	Character
Purpose	Returns parent ID of current activity.

_My_Preds_Parents

Expression	GET_PREDS('IIF(SUBSTR((str(PRED_ACT_ID._My_Parent) <> str(SUCC_ACT_ID._My_Parent)),1,1) = "T","T","")')
Table	Activity
Data type	Character
Purpose	Returns parent predecessor ID of current activity.

_My_Succs_Parents

Expression	GET_SUCCS('IIF(SUBSTR((str(PRED_ACT_ID._My_Parent) <> str(SUCC_ACT_ID._My_Parent)),1,1) = "T","T","")')
Table	Activity
Data type	Character
Purpose	Returns parent succesor ID of current activity.

_Non_Critical_Activities_Only

Expression	(Critical_Activity = [BOOL.F] or _Option_Highlight_Critical_Path = 0) and _Is_Activity = [BOOL.T]
Table	Activity

Data type	Logical
Purpose	Returns true if an activity is not of type “External Subproject” or “Subproject” AND one of the following conditions is true: ▪ The CRITICAL field is not set to “Not Critical.” • The _Option_Highlight_Critical_Path is set to 0.

_Not_Planned_Activities

Expression	COMPSTAT <> [ACTS.0] and _Is_Activity = [BOOL.T]
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for an activity is not set to “Planned” and the activity is neither of type “External Subproject” nor “Subproject."

_Opt_Highlight_Critical

Expression	CRITICAL <> [CRIT.0] and _Is_Activity = [BOOL.T] and _Option_Highlight_Critical_Path = 1
Table	Activity
Data type	Logical
Purpose	Returns true if the CRITICAL field for an activity is not set to “Not Critical,"the activity is neither of type “External Subproject” nor “Subproject,"and the _Option_Highlight_Critical_Path is set to 1.

_Opt_Milestone

Expression	(ACT_TYPE = [ACTT.S] or ACT_TYPE = [ACTT.F]) and _Option_Show_Milestone_Dates = 1
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is either of type “Start Milestone” or “Finish Milestone” and the _Option_Show_Milestone_Dates calculated field is set to 1.

_Opt_Show_Float

Expression	_Option_Show_Float = 1 and _Non_Critical_Activities_Only = [BOOL.T]
Table	Activity
Data type	Logical
Purpose	Returns true if the _Option_Show_Float calculated field is set to 1 and the _Non_Critical_Activities_Only calculated field returns a value of true.

_Option_Highlight_Critical_Path

Expression	1

Table	Activity
Data type	Integer
Purpose	Used in the supplied bar set OptionsBar (used by OptionsBarchart AddIn View).

_Option_Show_Float

Expression	1
Table	Activity
Data type	Integer
Purpose	Used in the supplied bar set OptionsBar (used by OptionsBarchart AddIn View).

_Option_Show_Milestone_Dates

Expression	1
Table	Activity
Data type	Integer
Purpose	Used in the supplied bar set OptionsBar (used by OptionsBarchart AddIn View).

_pred_test

Expression	IIF(SUBSTR(STR(PRED_ACT_ID._MY_NETWORK_FILTER), 1, 1) = 'T', 'T', '')
Table	Relationship
Data type	Character
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_succ_test

Expression	IIF(SUBSTR(STR(SUCC_ACT_ID._MY_NETWORK_FILTER), 1, 1) = 'T', 'T', '')
Table	Relationship
Data type	Character
Purpose	Used by the supplied view FILTNET (Filtered Network View).

_The_Real_Network_Filter

Expression	_My_Network_Filter = [BOOL.T] or _Check_My_Preds CONTAINS('T') or _Check_My_Succs CONTAINS('T')
Table	Activity
Data type	Logical
Purpose	Used by the supplied view FILTNET (Filtered Network View).

Activity_is_Late

Expression	_ACT.ESDATE > BSDATE or _ACT.EFDATE > BFDATE

Table	Baseline Activity
Data type	Logical
Purpose	Returns true if the activity corresponding to the baseline activity has an early start date later than its baseline start date or an early finish date later than its baseline finish date.

Activity_is_Milestone

Expression	_ACT.MILESTONE = [BOOL.T]
Table	Baseline Activity
Data type	Logical
Purpose	Returns true if the activity corresponding to the baseline activity is a milestone activity.

Activity_is_Ontime

Expression	_ACT.ESDATE <= BSDATE and _ACT.EFDATE <= BFDATE
Table	Baseline Activity
Data type	Logical
Purpose	Returns true if the activity corresponding to the baseline activity has an early start date earlier than or equal to its baseline start date or an early finish date earlier than or equal to its baseline finish date.

Accomplished_Duration

Expression	ORIG_DUR-REM_DUR
Table	Activity
Data type	Duration
Purpose	Calculates the accomplished duration of an activity.

Activity_Desc

Expression	ACT_ID.DESCRIPTION
Table	Resource Assignment
Data type	Character
Purpose	Displays the description of an activity. (The ID portion of the expression indicates that the calculated field that links to the Resource Assignment table and the Activity table where the information is stored uses the ID field.)

Activity_Dur

Expression	ACT_ID.ORIG_DUR
Table	Resource Assignment
Data type	Duration

Purpose	Displays the original duration of an activity. (The ID portion of the expression indicates that the calculated field that links to the Resource Assignment table and the Activity table where the information is stored uses the ID field.)

Activity_Status

Expression	ACT_ID.COMPSTAT
Table	Resource Assignment
Data type	Character
Purpose	Displays the computed status of an activity. (The ID portion of the expression indicates that the calculated field that links to the Resource Assignment table and the Activity table where the information is stored uses the ID field.)

ACTRES_Description

Expression	act_id.description
Table	Usage
Data type	Character
Purpose	Returns the description for the activity associated with the current usage record.

ACTRES_OrigDur

Expression	act_id.orig_dur
Table	Usage
Data type	Duration
Purpose	Returns the original duration for the activity associated with the current usage record.

ACTRES_Requested

Expression	iif(alt_res_id <> '', alt_res_id, res_id)
Table	Usage
Data type	Character
Purpose	Returns the alternate resource ID for the record if this field is not blank. Otherwise, the resource ID is returned.

ACTRES_SchedDur

Expression	act_id.sched_dur
Table	Usage
Data type	Duration
Purpose	Returns the scheduled duration for the activity associated with the current usage record.

ACTRES_SFDate

Expression	act_id.sfdate
Table	Usage
Data type	Finish Date
Purpose	Returns the scheduled finish date for the activity associated with the current usage record.

ACTRES_SSDate

Expression	act_id.ssdate
Table	Usage
Data type	Date
Purpose	Returns the scheduled start date for the activity associated with the current usage record.

ACTRES_Suggested

Expression	iif(alt_res_id <> '', res_id, '')
Table	Usage
Data type	Character
Purpose	Returns the resource ID for the record if the alternate resource ID is not blank. Otherwise, a blank string is returned.

ACWPcum

Expression	((ACWP_LAB + ACWP_MAT) + ACWP_ODC) + ACWP_SUB
Table	Activity
Data type	Decimal
Purpose	Adds the actual resource cost of an activity to the cumulative actual cost of work performed.

All_Finish_Dates

Expression	“Early:”+DATEFORMAT(EFDATE,”%D%A%Y”) +NEWLINE(1)+“Late:”+DATEFORMAT(LFDATE,”%D%A%Y”) +NEWLINE(1)+“Sched:” +DATEFORMAT(SFDATE,”%D%A%Y”)
Table	Activity
Data type	Character
Purpose	Displays early, late, and scheduled finish dates.

All_Project_Finish_Dates

Expression	((((((' Early: ' + DATEFORMAT(EFDATE, '%D%A%Y')) + NEWLINE(1)) + ' Late: ') + DATEFORMAT(LFDATE, '%D%A%Y')) + NEWLINE(1)) + 'Sched: ') + DATEFORMAT(SFDATE, '%D%A%Y')
Table	Project
Data type	Character
Purpose	Returns early finish, late finish, and scheduled finish dates for a project.

All_Res

Expression	GET_ASSGNS(“RES_ID”)
Table	Activity
Data type	Character
Purpose	Returns the resource ID for every assignment on a given activity.

All_Start_Dates

Expression	“Early:”+DATEFORMAT(ESDATE,”%D%A%Y”) +NEWLINE(1)+“Late:”+DATEFORMAT(LSDATE,”%D%A%Y”) +NEWLINE(1)+“Sched:” +DATEFORMAT(SSDATE,”%D%A%Y”)
Table	Activity
Data type	Character
Purpose	Displays early, late, and scheduled start dates.

Assignments

Expression	GET_ASSGNS(‘RES_ID|RES_LEVEL’)
Table	Activity
Data type	Character
Purpose	Returns the resource ID and level for every assignment on a given activity.

AVAIL_ResClass

Expression	str(res_id.res_class)
Table	Availability
Data type	Character
Purpose	Returns the resource class for the corresponding resource description record.

AVAIL_ResDesc

Expression	res_id.description
Table	Availability

Data type	Character
Purpose	Returns the description for the corresponding resource description record.

AVAIL_ResType

Expression	str(res_id.res_type)
Table	Availability
Data type	Character
Purpose	Returns the resource type for the corresponding resource description record.

AVAIL_ResUnits

Expression	res_id.unit
Table	Availability
Data type	Character
Purpose	Returns the value of the unit property for the corresponding resource description record.

AVAIL_UnitCost

Expression	res_id.unit_cost
Table	Availability
Data type	Decimal
Purpose	Returns the value of the unit cost property for the corresponding resource description record.

BACcum

Expression	((BAC_LAB + BAC_MAT) + BAC_ODC) + BAC_SUB
Table	Activity
Data type	Decimal
Purpose	Adds the Budgeted Actual Cost for all resource categories.

Baseline_Finish_Variance_Percent

Expression	100 * DATEDIFFERENCE(BFDATE, EFDATE, CLH_ID) / ORIG_DUR
Table	Activity
Data type	Decimal
Purpose	Calculates the percentage variance between the baseline finish date and the early finish date.

BCWPcum

Expression	((BCWP_LAB + BCWP_MAT) + BCWP_ODC) + BCWP_SUB
Table	Activity
Data type	Decimal
Purpose	Calculates the cumulative earned value of an activity.

BCWScum

Expression	((BCWS_LAB + BCWS_MAT) + BCWS_ODC) + BCWS_SUB
Table	Activity
Data type	Decimal
Purpose	Gives the cumulative total for BCWP.

BF_Label

Expression	'BF : ' + STR(DATEFORMAT(BFDATE, '%M/%D/%Y'))
Table	Activity
Data type	Character
Purpose	Displays the baseline finish date for the activity labeled as “BF :.”

BS_Label

Expression	'BS : ' + STR(DATEFORMAT(BSDATE, '%M/%D/%Y'))
Table	Activity
Data type	Character
Purpose	Displays the baseline start date for the activity labeled as “BS :.”

Budget_at_Completion

Expression	((BAC_LAB + BAC_MAT) + BAC_ODC) + BAC_SUB
Table	Activity
Data type	Decimal
Purpose	Calculates the total budget for an activity.

c1desc

Expression	C1.DESCRIPTION
Table	Activity
Data type	Character
Purpose	Displays the description for the code assigned to the current activity at index 1.

c2desc

Expression	C2.DESCRIPTION
Table	Activity
Data type	Character
Purpose	Displays the description for the code assigned to the current activity at index 2.

c3desc

Expression	C3.DESCRIPTION
Table	Activity
Data type	Character
Purpose	Displays the description for the code assigned to the current activity at index 3.

Client_Label

Expression	'Client - ' + OPCLIENT
Table	Project
Data type	Character
Purpose	Displays the OPCLIENT field of a project labeled as “Client -.”

Complete_Milestone

Expression	(ACT_TYPE = [ACTT.S] or ACT_TYPE = [ACTT.F] or ORIG_DUR = |0|) and COMPSTAT = [ACTS.2]
Table	Activity
Data type	Logical
Purpose	Returns true if an activity has been marked as completed AND at least one of the following conditions is true – ▪ The activity type is a start milestone; • The activity type is a finish milestone; or ▪ The original duration is 0.

ControllingLogic

Expression	USER_NUM01 = 1
Table	Activity
Data type	Logical
Purpose	Used in the supplied view LOGICT (Logic Trace Network).

Cost_Info

Expression	(((((((('Budget: ' + NUMBER_FORMAT(PROJECT_BAC, '$,')) + NEWLINE(1)) + 'Actual: ') + NUMBER_FORMAT(PROJECT_ACWP, '$,')) + NEWLINE(1)) + 'Earned: ') + NUMBER_FORMAT(PROJECT_BCWP, '$,')) + NEWLINE(1)) + 'Scheduled: ') + NUMBER_FORMAT(PROJECT_BCWS, '$,')
Table	Project
Data type	Character
Purpose	Returns the budget, actual, earned, and scheduled costs for a project.

CPI

Expression	IIF(ACWPcum>0,((BCWPcum * 1.0) / ACWPcum),0)
Table	Activity
Data type	Decimal
Purpose	Calculates the Cost Performance Index (CPI) value for an activity.

Critical_Activity

Expression	CRITICAL = [CRIT.3] or CRITICAL = [CRIT.1] or CRITICAL = [CRIT.2]
Table	Activity
Data type	Logical
Purpose	Returns True if an activity's critical field has been set to Controlling Critical, Critical, or Most Critical.

CV

Expression	BCWPcum-ACWPcum
Table	Activity
Data type	Decimal
Purpose	Calculates the cost variance for an activity by subtracting the cumulative ACWP from the cumulative BCWP.

CV_label

Expression	"Cost Variance: "+STR(CV)
Table	Activity
Data type	Character
Purpose	Displays the cost variance of an activity labeled as "Cost Variance:."

Decrease_All_Durations_By_20_Percent

Expression	ORIG_DUR * .8
Table	Activity

Data type	Duration
Purpose	Decreases an activity's original duration by 20%.

Detail

Expression	ACT_TYPE = [ACTT.N]
Table	Activity
Data type	Logical
Purpose	Returns true if an activity is of type "normal" (ASAP).

Duration_Label

Expression	"Dur = "+STR(ORIG_DUR)
Table	Activity
Data type	Character
Purpose	Displays the original duration of an activity labeled as "Dur =."

EAC

Expression	ACWPcum+ETC
Table	Activity
Data type	Decimal
Purpose	Calculates the estimated total cost of an activity when complete.

Early_Dates

Expression	DATEFORMAT(ESDATE,"%D%A%Y") +NEWLINE(1)+DATEFORMAT(EFDATE,"%D%A%Y")
Table	Activity
Data type	Character
Purpose	Displays early start and finish dates.

Early_Finish_5P

Expression	DATEADD(MEAN_EF,((0-2)*SDEV_EF),CLH_ID)
Table	Activity
Data type	Finish Date
Purpose	First date in range describing 90% confidence band for early finish date.

Early_Finish_95P

Expression	DATEADD(MEAN_EF,(2*SDEV_EF), CLH_ID)

Table	Activity
Data type	Finish Date
Purpose	Last date in range describing 90% confidence band for early finish date.

Early_Finish_Label

Expression	"EF:"+STR(DATEFORMAT(EFDATE,"%M/%D/%Y"))
Table	Activity
Data type	Character
Purpose	Displays the early finish date of an activity labeled as “EF:.”

Early_or_Actual_Finish

Expression	iif(AFDATE <> {}, DATEFORMAT(AFDATE, "A: %D%A%Y"), DATEFORMAT(EFDATE, "%D%A%Y"))
Table	Activity
Data type	Character
Purpose	Returns the actual finish date for an activity, if it is not blank. Otherwise, the early finish date is returned.

Early_or_Actual_Start

Expression	iif(ASDATE <> {}, DATEFORMAT(ASDATE, "A: %D%A%Y"), DATEFORMAT(ESDATE, "%D%A%Y"))
Table	Activity
Data type	Character
Purpose	Returns the actual start date for an activity, if it is not blank. Otherwise, the early start date is returned.

Early_Start_5P

Expression	DATEADD(MEAN_ES,((0-2)*SDEV_ES), CLH_ID)
Table	Activity
Data type	Date
Purpose	First date in range describing 90% confidence band for early start date.

Early_Start_95P

Expression	DATEADD(MEAN_ES,(2*SDEV_ES), CLH_ID)
Table	Activity
Data type	Date
Purpose	Last date in range describing 90% confidence band for early start date.

Early_Start_Label

Expression	"ES:"+STR(DATEFORMAT(ESDATE,"%M/%D/%Y"))
Table	Activity
Data type	Character
Purpose	Displays the early start date of an activity labeled as “ES:.”

End_Activity

Expression	LOGICFLAG = [LOGI.F] or LOGICFLAG = [LOGI.SF] or LOGICFLAG = [LOGI.I]
Table	Activity
Data type	Logical
Purpose	Returns true if the LOGICFLAG field for an activity is set to “End Activity,” “Start and Finish Activity,” or “Isolated."

ETC

Expression	((ETC_LAB + ETC_MAT) + ETC_ODC) + ETC_SUB
Table	Activity
Data type	Decimal
Purpose	Calculates the estimated remaining cost of an activity.

External_Subproject

Expression	ACT_TYPE = [ACTT.E]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “External Subproject."

Foreign_Activity

Expression	ACT_TYPE = [ACTT.G]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “Foreign Activity” (a placeholder for an activity in an unopened project).

Foreign_Subproject

Expression	ACT_TYPE = [ACTT.Z]
Table	Activity
Data type	Logical

Purpose	Returns true if the activity is of type “Foreign Subproject” (a placeholder for a subproject activity in an unopened project).

Free_Float_Label

Expression	"FF = "+STR(FREEFLOAT)
Table	Activity
Data type	Character
Purpose	Displays the FREEFLOAT field of an activity labeled as “FF:.”

Hammock

Expression	ACT_TYPE = [ACTT.H]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “Hammock.”

Has_Cost

Expression	ACWPcum > 0
Table	Activity
Data type	Logical
Purpose	Returns true if the ACWP is greater than 0.

HasRelationships

Expression	GET_PREDS('pred_act_id') + GET_SUCCS('succ_act_id') NOT_EMPTY
Table	Activity
Data type	Logical
Purpose	Returns true if the activity has one or more relationships.

In_Progress

Expression	COMPSTAT = [ACTS.1]
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for an activity is set to “In Progress."

In_Progress_Critical

Expression	COMPSTAT = [ACTS.1] and TOTALFLOAT <> |0|
Table	Activity
Data type	Logical

Purpose	Returns true if the computed status for an activity is set to “In Progress,"and the total float field has a non-zero duration.

In_Progress_Non_Critical

Expression	COMPSTAT = [ACTS.1] and TOTALFLOAT = |0|
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for an activity is set to “In Progress,” and the total float field has a duration of zero.

Incomplete_Milestone

Expression	(ACT_TYPE = [ACTT.S] or ACT_TYPE = [ACTT.F] or ORIG_DUR = |0|) and not COMPSTAT = [ACTS.2]
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for an activity is not set to complete AND at least one of the following conditions is also true: ▪ The activity is of type Start Milestone or Finish Milestone. • The original duration is zero.

Increase_All_Durations_by_50_Percent

Expression	ORIG_DUR * 1.5
Table	Activity
Data type	Duration
Purpose	Increases an activity’s original duration by 50%.

Increase_Requirement_by_a_Factor

Expression	RES_LEVEL * 1.2
Table	Assignment
Data type	Decimal
Purpose	Increases an assignment’s resource level by 20%.

Label_Early

Expression	“Early”
Table	Activity
Data type	Character
Purpose	Displays the label for early dates in crosstables.

Label_Late

Expression	“Late”
Table	Activity
Data type	Character
Purpose	Displays the label for late dates in crosstables.

Label_Scheduled

Expression	“Scheduled”
Table	Activity
Data type	Character
Purpose	Displays the label for scheduled dates in crosstables.

Late_1_to_10

Expression	Baseline_Finish_Variance_Percent > 0 and Baseline_Finish_Variance_Percent <= 10
Table	Activity
Data type	Logical
Purpose	Returns true if the baseline finish variance is greater than 0 but less than 10%.

Late_11_to_100

Expression	Baseline_Finish_Variance_Percent > 10
Table	Activity
Data type	Logical
Purpose	Returns true if the baseline finish variance is greater than 10%.

Late_Dates

Expression	DATEFORMAT(LSDATE,”%D%A%Y”) +NEWLINE(1)+DATEFORMAT(LFDATE,”%D%A%Y”)
Table	Activity
Data type	Character
Purpose	Displays late start and finish dates.

LogicStart

Expression	USER_NUM01 = 3
Table	Activity
Data type	Logical
Purpose	Used in the supplied view LOGICT (Logic Trace Network).

LogicTrace

Expression	USER_NUM01 = 2
Table	Activity
Data type	Logical
Purpose	Used in the supplied view LOGICT (Logic Trace Network).

LogicTraceOther

Expression	USER_NUM01 = 0
Table	Activity
Data type	Logical
Purpose	Used in the supplied view LOGICT (Logic Trace Network).

Milestone

Expression	ACT_TYPE = [ACTT.S] or ACT_TYPE = [ACTT.F]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “Start Milestone” or “Finish Milestone.”

NeedsAllocation

Expression	ALT_RES_ID NOT_EMPTY
Table	Usage
Data type	Logical
Purpose	Returns true if the alternate resource ID is not blank on a usage record.

Neg_Float

Expression	TOTALFLOAT < |0| and (ACT_TYPE = [ACTT.N] or ACT_TYPE = [ACTT.S] or ACT_TYPE = [ACTT.F])
Table	Activity
Data type	Logical
Purpose	Returns true if the total float of an activity has a duration less than 0 and one of the following conditions is true: ▪ Activity is of type “normal” (ASAP) ● Activity is of type “Start Milestone” ▪ Activity is of type “Finish Milestone”

Next_month

Expression	ESDATE BETWEEN(Today , Today_plus_30_days)

Table	Activity
Data type	Logical
Purpose	Returns true if the early start date for the activity falls within a range of 30 days from the current date.

Non_Critical

Expression	Critical_Activity = [BOOL.F]
Table	Activity
Data type	Logical
Purpose	Returns true if the CRITICAL field for the activity is set to “Not Critical."

Non_Hammock

Expression	ACT_TYPE <> [ACTT.H]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is not of type “Hammock."

Not_Activity

Expression	ACT_TYPE = [ACTT.E] or ACT_TYPE = [ACTT.P]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “External Subproject” or “Subproject."

Not_Completed

Expression	COMPSTAT <> [ACTS.2]
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for the activity is not set to “Complete."

Not_Planned

Expression	COMPSTAT <> [ACTS.0]
Table	Activity
Data type	Logical
Purpose	Returns true if the computed status for the activity is not set to “Planned."

Not_Pool

Expression	RES_TYPE <> [REST.P]

Table	Resource Data
Data type	Logical
Purpose	Returns true if the resource type for the resource is not set to “Resource Pool."

Not_Resource_Critical

Expression	SSDATE <= FEDATE
Table	Activity
Data type	Logical
Purpose	Returns true if the activity’s scheduled start date is less than or equal to its earliest feasible start date.

Not_Subproject

Expression	Subproject = [BOOL.F]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is not of type “Subproject."

NURELS_COMPSTATS

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('PRED_ACT_ID.COMPSTAT'), ';', NEWLINE(1))) + NEWLINE(1))) + STR(COMPSTAT)) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('SUCC_ACT_ID.COMPSTAT'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_DESCRIPTIONS

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('PRED_ACT_ID.DESCRIPTION'), ';', NEWLINE(1))) + NEWLINE(1))) + DESCRIPTION) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('SUCC_ACT_ID.DESCRIPTION'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_EFDATES

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS("IIF((PRED_ACT_ID.AFDATE <> {}),

DATEFORMAT(PRED_ACT_ID.AFDATE, 'A: %D%A%Y'),
DATEFORMAT(PRED_ACT_ID.EFDATE, '%D%A%Y'))"), ';', NEWLINE(1))) +
NEWLINE(1))) + EARLY_OR_ACTUAL_FINISH) +
IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) +
STRTRAN(GET_SUCCS("IIF((SUCC_ACT_ID.AFDATE <> {}),
DATEFORMAT(SUCC_ACT_ID.AFDATE, 'A: %D%A%Y'),
DATEFORMAT(SUCC_ACT_ID.EFDATE, '%D%A%Y'))"), ';', NEWLINE(1))))

Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_ESDATES

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS("IIF((PRED_ACT_ID.ASDATE <> {}), DATEFORMAT(PRED_ACT_ID.ASDATE, 'A: %D%A%Y'), DATEFORMAT(PRED_ACT_ID.ESDATE, '%D%A%Y'))"), ';', NEWLINE(1))) + NEWLINE(1))) + EARLY_OR_ACTUAL_START) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS("IIF((SUCC_ACT_ID.ASDATE <> {}), DATEFORMAT(SUCC_ACT_ID.ASDATE, 'A: %D%A%Y'), DATEFORMAT(SUCC_ACT_ID.ESDATE, '%D%A%Y'))"), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_IDs

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', (((' Predecessor' + NEWLINE(1)) + STRTRAN(GET_PREDS("' ' + PRED_ACT_ID"), ';', NEWLINE(1))) + NEWLINE(1))) + ACT_ID) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (((NEWLINE(1) + ' Successor') + NEWLINE(1)) + STRTRAN(GET_SUCCS("' ' + SUCC_ACT_ID"), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_LFDATES

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS("DATEFORMAT(PRED_ACT_ID.LFDATE, '%D%A%Y')"), ';', NEWLINE(1))) + NEWLINE(1))) + DATEFORMAT(LFDATE, '%D%A%Y')) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS("DATEFORMAT(SUCC_ACT_ID.LFDATE, '%D%A%Y')"), ';', NEWLINE(1))))
Table	Activity
Data type	Character

Purpose	Used in the supplied project view NURELS.

NURELS_LSDATES

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS("DATEFORMAT(PRED_ACT_ID.LSDATE, '%D%A%Y')"), ';', NEWLINE(1))) + NEWLINE(1))) + DATEFORMAT(LSDATE, '%D%A%Y')) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS("DATEFORMAT(SUCC_ACT_ID.LSDATE, '%D%A%Y')"), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_ORIG_DURS

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('PRED_ACT_ID.ORIG_DUR'), ';', NEWLINE(1))) + NEWLINE(1))) + STR(ORIG_DUR)) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('SUCC_ACT_ID.ORIG_DUR'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_REL_LAGS

Expression	IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('REL_LAG'), ';', NEWLINE(1))) + NEWLINE(1))) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('REL_LAG'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_REL_TYPES

Expression	IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS("IIF((REL_TYPE = [Finish to Start]), 'FS', IIF((REL_TYPE = [Finish to Finish]), 'FF', IIF((REL_TYPE = [Start to Start]), 'SS', IIF((REL_TYPE = [Start to Finish]), 'SF', ''))))"), ';', NEWLINE(1))) + NEWLINE(1))) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS("IIF((REL_TYPE = [Finish to Start]), 'FS', IIF((REL_TYPE = [Finish to Finish]), 'FF', IIF((REL_TYPE = [Start to Start]), 'SS', IIF((REL_TYPE = [Start to Finish]), 'SF', ''))))"), ';', NEWLINE(1))))
Table	Activity
Data type	Character

Purpose	Used in the supplied project view NURELS.

NURELS_REM_DURS

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('PRED_ACT_ID.REM_DUR'), ';', NEWLINE(1))) + NEWLINE(1))) + STR(REM_DUR)) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('SUCC_ACT_ID.REM_DUR'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

NURELS_TOTALFLOATS

Expression	(IIF((GET_PREDS('PRED_ACT_ID') = ''), '', ((NEWLINE(1) + STRTRAN(GET_PREDS('PRED_ACT_ID.TOTALFLOAT'), ';', NEWLINE(1))) + NEWLINE(1))) + STR(TOTALFLOAT)) + IIF((GET_SUCCS('SUCC_ACT_ID') = ''), '', (NEWLINE(2) + STRTRAN(GET_SUCCS('SUCC_ACT_ID.TOTALFLOAT'), ';', NEWLINE(1))))
Table	Activity
Data type	Character
Purpose	Used in the supplied project view NURELS.

On_Time

Expression	ESDATE = BSDATE and EFDATE = BFDATE
Table	Activity
Data type	Logical
Purpose	Returns true if an activity's early start date equals its baseline start date and its early finish date equals its baseline finish date.

Opt_Highlight_Critical_Sub

Expression	CRITICAL <> [CRIT.0] and _Is_Activity = [BOOL.F] and _Option_Highlight_Critical_Path = 1
Table	Activity
Data type	Logical
Purpose	Returns true if the following conditions are met: ▪ The CRITICAL field for the activity is set to a value other than "Not Critical." • The activity is of type "External Subproject" or "Subproject." ▪ The _Option_Highlight_Critical_Path calculated field has a value of 1.

Pathn (where n is an integer between 1 and 20)

Expression	USER_CHR10 = 'cc' (where cc is a character string between ‘01’ and ‘20’)
Table	Activity
Data type	Logical
Purpose	Returns true if the USER_CHR10 field contains the character representation of the path number.

Path1to20

Expression	USER_CHR10 BETWEEN('01' , '20')
Table	Activity
Data type	Logical
Purpose	Returns true if the USER_CHR10 field contains a value between “01” and “20."

Path1to5

Expression	USER_CHR10 BETWEEN('01' , '05')
Table	Activity
Data type	Logical
Purpose	Returns true if the USER_CHR10 field contains a value between “01” and “05."

Planned_Baseline_Progress

Expression	IIF(BSDATE <> {}, IIF(BFDATE < TIMENOW(), 100, IIF(BSDATE >= TIMENOW(), 0, IIF(ORIG_DUR > |0|, ((DATEDIFFERENCE(BSDATE, TIMENOW(), CLH_ID) / ORIG_DUR) * 100), ((DATEDIFFERENCE(BSDATE, TIMENOW(), CLH_ID) / DATEDIFFERENCE(BSDATE, BFDATE, CLH_ID) * 100))))), -1)
Table	Activity
Data type	Decimal
Purpose	Calculates the planned completion percentage of an activity, based on Time Now and the baseline start date.

PM_Info

Expression	(OPMANAGER + NEWLINE(2)) + PM_EMAIL
Table	Project
Data type	Character
Purpose	Returns the Project Manager’s name and email address.

Pred_Desc

Expression	PRED_ACT_ID.DESCRIPTION
Table	Relationship

Data type	Character
Purpose	Displays the description of the predecessor activity. (The ID portion of the expression indicates the calculated field that links to the Relationship table and the Activity table, where the information is stored, uses the ID field.)

Pred_Status

Expression	PRED_ACT_ID.COMPSTAT
Table	Relationship
Data type	Character
Purpose	Displays the computed status of the predecessor activity. (The ID portion of the expression indicates that the calculated field that links to the Relationship table and the Activity table, where the information is stored, uses the ID field.)

Progress_Finish

Expression	IIF(VAL(AFDATE)>0, AFDATE, IIF(ESDATE> TIMENOW(x), ESDATE, TIMENOW(x)))
Table	Activity
Data type	Finish Date
Purpose	Displays the actual finish date, if present. Otherwise, displays either the early finish date or Time Now, whichever is later.

Progress_Percent

Expression	IIF(COMPSTAT=[ACTS.2],100,IIF(COMPSTAT=[ACTS.0] OR ESDATE=EFDATE,0,100*(DATEDIFFERENCE(ESDATE,EFDATE,CALENDAR)-REM_DUR)/DATEDIFFERENCE(ESDATE,EFDATE,CALENDAR)))
Table	Activity
Data type	Decimal
Purpose	Calculates the percentage of the original duration that is complete.

Note: If the actual start date is not entered, **Progress_Percent** will not return the expected result.

Progress_Start

Expression	IIF(val(ASDATE)>0,ASDATE,ESDATE)
Table	Activity
Data type	Date
Purpose	Displays the actual start date of an activity that is either in progress or complete. If the activity has a status of planned, this calculated field displays the early start date.

Project_ACWP

Expression	((ACWP_LAB + ACWP_MAT) + ACWP_ODC) + ACWP_SUB
Table	Project
Data type	Decimal
Purpose	Gives the cumulative total for ACWP.

Project_BAC

Expression	((BAC_LAB + BAC_MAT) + BAC_ODC) + BAC_SUB
Table	Project
Data type	Decimal
Purpose	Gives the cumulative total BAC.

Project_BCWP

Expression	((((BCWP_LAB + BCWP_MAT) + BCWP_ODC) + BCWP_SUB
Table	Project
Data type	Decimal
Purpose	Gives the cumulative total for BCWP.

Project_BCWS

Expression	((((((BCWS_LAB + BCWS_MAT) + BCWS_ODC) + BCWS_SUB
Table	Project
Data type	Decimal
Purpose	Gives the cumulative total for BCWS.

Project_Quick_Overview

Expression	((((((('Phase: ' + PROJSTATUS) + NEWLINE(1)) + 'Status: ') + OPSTAT) + NEWLINE(1)) + 'Time Now: ') + DATEFORMAT(STATDATE, '%D%A%Y')
Table	Project
Data type	Character
Purpose	Returns Project Status and Time Now information.

Resource_Activities

Expression	Milestone = [BOOL.F] and COMPSTAT <> [ACTS.2]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is NOT of type “Start Milestone” or “Finish Milestone” AND its computed status is NOT set to “Complete."

Resource_Critical

Expression	SSDATE > FEDATE
Table	Activity
Data type	Logical
Purpose	Returns true if the activity's scheduled start date is greater than its earliest feasible start date.

Resource_Dates

Expression	DATEFORMAT(SSDATE,"%D%A%Y") +NEWLINE(1)+DATEFORMAT(SFDATE,"%D%A%Y")
Table	Activity
Data type	Character
Purpose	Displays scheduled start and finish dates.

ResourceTypeNotEqualSkill

Expression	RES_TYPE <> [REST.S]
Table	Resource Data
Data type	Logical
Purpose	Returns true if a resource is not of type "Skill."

Risk_1_to_50_Critical

Expression	CRITINDEX > 0 and CRITINDEX <= 50
Table	Activity
Data type	Logical
Purpose	Returns true if the CRITINDEX field has a value greater than 0 and less than or equal to 50.

Risk_51_to_100_Critical

Expression	CRITINDEX > 50
Table	Activity
Data type	Logical
Purpose	Returns true if the CRITINDEX field has a value greater than 50.

Risk_Critical

Expression	CRITINDEX > 0
Table	Activity
Data type	Logical

Purpose	Returns true if the CRITINDEX field has a value greater than 0.

Risk_Not_Critical

Expression	CRITINDEX = 0
Table	Activity
Data type	Logical
Purpose	Returns true if the CRITINDEX field has a value equal to 0.

Sch_var

Expression	DATEDIFFERENCE(ASDATE, BSDATE, '')
Table	Activity
Data type	Duration
Purpose	Returns the difference between the actual start date and the baseline start date.

Schedfin_Baseline

Expression	IIF(Val(B00~AFDATE)>0,B00~AFDATE,B00~EFDATE)
Table	Activity
Data type	Date
Purpose	Returns the actual finish date on the first selected baseline if this value is not blank. Otherwise, the early finish date for the first selected baseline is returned.

Schedstart_Baseline

Expression	IIF(Val(B00~ASDATE)>0,B00~ASDATE,B00~ESDATE)
Table	Activity
Data type	Date
Purpose	Returns the actual start date on the first selected baseline if this value is not blank. Otherwise, the early start date for the first selected baseline is returned.

Scheduled_dates

Expression	SSDATE NOT_EMPTY
Table	Activity
Data type	Logical
Purpose	Returns true if the scheduled start date for the activity is not blank.

SPI

Expression	((BCWPcum*1.0)/BCWSPcum)
Table	Activity

Data type	Decimal
Purpose	Calculates the Schedule Performance Index (SPI) value for an activity.

Start_Activity

Expression	LOGICFLAG = [LOGI.S] or LOGICFLAG = [LOGI.SF] or LOGICFLAG = [LOGI.I]
Table	Activity
Data type	Logical
Purpose	Returns true if the LOGICFLAG field for the activity is set to “Start Activity”,“Start and Finish Activity”,or “Isolated”.

Sub_Early_Finish

Expression	_SUB.S_EFDATE
Table	Activity
Data type	Finish Date
Purpose	Displays the early finish date for a subproject.

Sub_Early_Start

Expression	_SUB.S_ESDATE
Table	Activity
Data type	Date
Purpose	Displays the early start date for a subproject.

Subproject

Expression	ACT_TYPE = [ACTT.P]
Table	Activity
Data type	Logical
Purpose	Returns true if the activity is of type “Subproject."

Succ_Desc

Expression	SUCC_ACT_ID.DESCRIPTION
Table	Relationship
Data type	Character
Purpose	Displays the description of the successor activity. (The ID portion of the expression indicates that the calculated field that links the Relationship table to the Activity table, where the information is stored, uses the ID field.)

Succ_Status

Expression	SUCC_ACT_ID.COMPSTAT
Table	Relationship
Data type	Character
Purpose	Displays the computed status of the successor activity. (The ID portion of the expression indicates the calculated field that links the Relationship table to the Activity table, where the information is stored, uses the ID field.)

SV

Expression	BCWPCUM - BCWSCUM
Table	Activity
Data type	Decimal
Purpose	Calculates the schedule variance for an activity by subtracting the cumulative BCWS from the cumulative BCWP.

SV_Label

Expression	'Schedule Variance: ' + str(SV)
Table	Activity
Data type	Character
Purpose	Displays the schedule variance of an activity labeled as “Schedule Variance:.”

Text_Finish

Expression	IIF(LFDATE>EFDATE,LFDATE,EFDATE)
Table	Activity
Data type	Date
Purpose	Displays either the early or the late finish date for an activity, depending on which is later.

Text_Start

Expression	IIF(val(ASDATE)>0,ASDATE,ESDATE)
Table	Activity
Data type	Date
Purpose	Displays the actual start date of an activity that is either in progress or complete. If the activity has a status of planned, this calculated field displays the early start date.

TF_Label

Expression	“TF=“+STR(TOTALFLOAT)
Table	Activity

Data type	Character
Purpose	Displays the total float of an activity labeled as “TF.”

Today

Expression	Timenow()
Table	Activity
Data type	Date
Purpose	Returns Time Now for the project.

Today_plus_30_days

Expression	DATEADD(Timenow(x),|30d|,"")
Table	Activity
Data type	Date
Purpose	Returns the date that is 30 days past the project Time Now date.

VAC

Expression	BACcum-EAC
Table	Activity
Data type	Decimal
Purpose	Calculates the variance at complete as the budgeted cost of an activity at completion minus the estimated cost of the activity when complete.

Examples of Custom Calculated Fields

The following are examples of custom calculated fields.

_COMPARE_2_DATES

Expression	SUBSTR(ESDATE, 1, 8) = SUBSTR(LSDATE, 1, 8) Or DATEFORMAT(ESDATE, “%C%M%D”) = DATEFORMAT(LSDATE,”%C%M%D”)
Table	Activity
Data type	Logical
Purpose	Compares two date fields without regard to the time portion of the date string and returns **True** if dates are the same. This example uses the early start and late start dates.

_DISPLAY_MIXED_DUR_IN_DAYS

Expression	STR((VAL(ORIG_DUR) /60.0 /8), 6, 2) + “ days”

Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to days, and displays the result labeled as **days**. The calculation is based on eight hours per day.

_DISPLAY_MIXED_DUR_IN_HOURS

Expression	STR((VAL(ORIG_DUR) /60.0), 6, 2) + " hours"
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to hours, and displays the result labeled as **hours**.

_DISPLAY_MIXED_DUR_IN_MINUTES

Expression	STR((VAL(ORIG_DUR)), 8, 0) + " minutes"
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to minutes, and displays the result labeled as **minutes**.

_DISPLAY_MIXED_DUR_IN_WEEKS

Expression	STR((VAL(ORIG_DUR) /60.0 /40), 6, 2) + " weeks"
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to weeks, and displays the result labeled as **weeks**. This calculation is based on 40 hours per week.

_DISPLAY_MIXED_DUR_IN_MONTHS

Expression	STR((VAL(ORIG_DUR) /60.0 /160), 6, 2) + " months"
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to months, and displays the result labeled as **months**. This calculation is based on 160 hours per month.

_FORMATTED_DATE

Expression	DATEFORMAT(ESDATE,"%A %D, %C")
Table	Activity
Data type	Character
Purpose	Displays the early start date in month abbreviation, day of month, four-digit year format. For example, this calculated field might return a date such as **Oct 10, 2004**.

For a complete list of standard date formats, search for DATEFORMAT() in the Open Plan help system.

_INDENT_A_FIELD

Expression	IIF(LEVEL(ACT_ID) > 1, space((LEVEL(ACT_ID) - 1) * 3) + DESCRIPTION, DESCRIPTION)
Table	Activity
Data type	Character
Purpose	Indents any field based on the level of the activity ID. This example will indent the activity description 3 spaces for each ID level after the first.

_NUMERIC_TO_DURATION

Expression	|1d| * USER_NUM01
Table	Activity
Data type	Duration
Purpose	Converts any numeric field to the duration data type. This example uses User Numeric Field 1 and days for the duration units.

_START_HOURS_AT_0000

Expression	DATEFORMAT(ESDATE, '%D%A%C 00:00')
Table	Activity
Data type	Date
Purpose	Changes the time portion of a date field to 00:00. This example uses the early start date.

_END_HOURS_AT_2400

Expression	DATEFORMAT(EFDATE, '%D%A%C 24:00')
Table	Activity
Data type	Finish Date
Purpose	Changes the time portion of a date field to 24:00. This example uses the early finish date.

_ID_ANY_LEVEL

Expression	IIF(LEVEL(ACT_ID) >= 3, LOCAL(PARENT(ACT_ID, 3)), ““)
Table	Activity (or any table with a hierarchical field)
Data type	Character
Purpose	Returns the value of the specified level of the hierarchical ID field used. This example will return the value of the third level of the activity. If the activity ID has

less than three levels, it returns nothing. If you want something other than the third level, change the numbers representing

Processes

Time Analysis Calculations

Although it is relatively easy to initiate time analysis in Open Plan, the actual calculations performed during the operation are, in fact, quite sophisticated. This is particularly the case in situations involving mixed calendars, different activity types, or subprojects.

To help users understand how the results of time analysis are produced in Open Plan, this section provides an in-depth discussion of the following topics:

- The topological sorting of project data
- The calculation of early dates
- The calculation of late dates
- Float calculations
- Criticality
- Subprojects
- Hammocks

Attention: For a description of how to perform time analysis in Open Plan, see **Project Analysis » Time Analysis** in the Open Plan online help.

Topological Sorting

The process of time-analyzing a project is performed in a forward pass and a backward pass through the network logic. Before performing these passes through the data, Open Plan must sort the activities into a suitable order (called a topological sort order) so that no activity is processed before any of its logical predecessors.

Open Plan automatically checks to see if a logical loop has been created in the project network whenever time analysis is performed. In most cases, the user will have to resolve the loop before continuing. There are two circumstances when a loop may be created and the user is able to continue working:

- **In a multi-user environment** – Two users sharing a network may be working on a project creating relationships at the same time. Between the two of them, they may have created a loop. The loop will not be detected while the project is still open. Once it is exited and reopened, one of the users will be asked to resolve the loop.
- **An excess of activities** – When an Open Plan project has an excessive amount of activities, Open Plan informs the user that it is switching off incremental loop detection.

In a logical loop, two activities become mutually dependent (that is, neither activity can start before the other).

The Calculation of Early Dates

As a general rule, Open Plan calculates the early start and finish dates for each activity during the forward pass of time analysis. In this pass, no activity is considered before all of its logical predecessors. There are a number of exceptions to this general rule. For example, the early dates of as-late-as-possible (ALAP) activities and of subprojects may be modified during the backward pass, which is done in reverse

topological sort order. The following discussion describes the initial calculation of early dates for all activities during the forward pass, while noting that in certain cases the dates calculated here might be subject to revision during the backward pass.

The early start date of an activity is determined by the following considerations:

- The status date (Time Now)
- Any target start date with one of the following types:
  - Not Earlier Than
  - On Target
  - Fixed Target
- Any project target start date
- Any predecessor relationships leading into the start of the activity
- The early start of a parent activity, if applicable

The early finish date of an activity, on the other hand, is determined by the following considerations:

- The early start date in combination with the remaining duration
- A target finish date with one of the following types:
  - Not Earlier Than
  - On Target
  - Fixed Target
- Any predecessor relationships leading into the finish of the activity
- For subprojects, the early finish dates of all child activities

In general, the date calculated by Open Plan will be the latest date resulting from any of the above considerations.

The Effect of Durations

It is possible for the calculation of early dates to be affected by the activity duration. This is due to the fact that the time interval between the early start and early finish of an activity, as measured in the calendar assigned to that activity, can never be less than the remaining duration of the activity. In addition to that restriction, the interval between an early start date and an early finish date must be exactly equal to the duration for activities that are not in-progress, discontinuous, or subprojects.

To keep this interval from exceeding the durations, Open Plan will delay the early start of these activities due to delays to the early finish. Consequently, both the early start and finish dates must be determined for such activities before any successors may be processed, even if these successors depend only upon the start of the predecessor activity.

For discontinuous activities, hammock activities, and subprojects, the effect of durations on the calculation of early dates is less restrictive. For discontinuous activities, Open Plan interprets the duration specified on the activity record as a minimum duration for the purpose of calculating early dates. This interpretation allows the start and finish dates of the activity to move relative to one another. As a result, it is possible for the interval between the early start and the early finish dates of a discontinuous activity to exceed the assigned duration.

For subprojects and hammocks, Open Plan ignores the duration specified on the activity record. Instead, Open Plan recalculates the duration as the interval between the early start and finish dates implied by the child activities (after taking the activity calendar into account).

The Effect of Relationship Information

Relationships in networks can be one of four types: finish-to-start (the most common), start-to-start, finish-to-finish, and start-to-finish. Each relationship can also include a lag and a calendar.

The lag on a relationship determines a minimum interval between the two events. For example, a finish-to-start relationship between activity A and activity B with a lag of 10 days means that activity B cannot start until 10 working days after the finish of activity A. By default this lag is interpreted using the calendar of the successor, but you can assign a different calendar to the relationship.

You can also define a lag as a percentage by following the numeric value with a percent sign (for example, 50%). Open Plan interprets this type of lag value as a percentage of the predecessor activity duration that must be complete before the successor can start.

The Effect of Calendars

All the calculations of early dates are based on the allowable working periods for the calendar of the activity or relationship in question.

There can be additional delays due to successive activities using different calendars. For example, suppose an activity using a 5-day week (Monday through Friday) has a predecessor that finishes on a Friday or Saturday. Assuming no lags, the later activity will not start until Monday. This situation can impact later activities regardless of their calendars.

A particular problem concerning calendars occurs when there is a finish-to-finish relationship between two activities on different calendars and when the predecessor has an early finish during a non-working period on the successor's calendar. For example, assume that activity A is on a 7-day per week calendar and finishes on a Saturday, while activity B is on a Monday-through-Friday calendar. In this situation, Open Plan will set the early finish date of activity B to the next valid work time plus a single time unit based on the minimum calculated duration defined for the project. Thus, if the minimum calculated duration for activity B is set to 1 hour, Open Plan will calculate the early finish date for activity B as 9:00 A.M. on Monday, assuming the workday begins at 8:00 A.M.

> **Attention:** For information on setting the minimum calculated duration for a project, see **Working with Projects » Tabs and Dialog Boxes » Project Properties Dialog Box » Preferences Tab** in the Open Plan online help.

There is an analogous situation with start-to-start relationships when the successor activity has a late start during a non-working period for the predecessor. In this case, Open Plan will force the activities to start at a time that is acceptable to the predecessor’s calendar.

The Modification of Early Dates

There are some cases in which Open Plan may modify the early dates after the forward pass is complete. (Note that this means that these modifications do not impact successor activities.) These special cases are as follows:

- Any start milestone activity will have its early finish date set equal to its early start date.
- Any finish milestone activity will have its early start date set equal to its early finish date.
- Any ALAP activity or subproject may have its early dates modified during the backward pass.

Substituting Actual Dates for Early Dates

Open Plan provides an option for replacing calculated early and late dates with actual start and finish dates for progressed and completed activities once time analysis is complete. This option has no effect on the calculations performed by time analysis. It is provided to simplify reporting when it is desirable to sort activities using either early or actual dates, depending on the status of the activity.

The use of this option may inhibit the diagnosis of certain scheduling problems. For example, when the option is disabled, the early dates calculated for progressed activities refer to only the remaining portion of the work, making it relatively easy to see where out-of-sequence progress has been recorded. By default, this option is enabled. You can disable the option by changing a setting on the **Options** tab of the **Time Analysis** dialog box.

The Calculation of Late Dates

Open Plan calculates late dates during the backward pass of time analysis. The backward pass processes the activities in reverse topological sort order so that no activity is considered before any of its successors. Working back from the project completion date, it is thus possible to calculate the latest date on which each activity needs to be finished and hence, the latest date by which the activity must start.

The project completion date used as a starting point for the backward pass depends upon a number of factors. For the project as a whole, this completion date is initially determined as the latest early finish date calculated during the forward pass.

This completion date may then be modified if the user has specified a project target finish date. The interpretation of this date also depends upon the target type as follows:

- **On Target** — The project target finish date is used instead of the calculated project finish date.
- **Not Earlier Than** — The project target finish date is used only if the calculated project finish date is earlier.
- **Not Later Than** — The project target finish date is used only if the calculated project finish date is later.

A final consideration arises if the multiple ends processing option has been specified on the **Advanced** tab of the **Time Analysis** dialog box. In that case, Open Plan treats each end activity as if the activity had a target finish equal to its early finish. This can result in a different starting point for the backward pass from each end activity.

> **Attention:** For information about the multiple ends processing option, see **Project Analysis » Time Analysis** in the Open Plan online help.

Once the project completion date has been determined, the calculation of late dates during the backward pass proceeds in a manner entirely analogous to the calculation of early dates during the forward pass.

The Calculation of ALAP Dates

During the backward pass, Open Plan determines whether each ALAP activity could be started later than its early start date without impacting the start dates of any successors. If this is the case, the early dates are made later until this is no longer the case. In practice, this operation amounts to calculating the free float and then, for ALAP activities only, changing the early dates so as to consume this float.

Because this is done during the backward pass, a series of consecutive ALAP activities may all have their dates modified. It is not until Open Plan has examined the last in the series that it knows whether the next-to-last can be made later, and so on.

The Modification of Late Dates

Cases in which the late dates calculated during the backward pass are subject to subsequent modification are as follows:

- Any start milestone activity with a zero duration will have its late finish date set equal to its late start date.
- Any finish milestone activity with a zero duration will have its late start date set equal to its late finish date.
- Any subproject may have its late dates modified.

Float Calculations

Open Plan calculates the following six float values during time analysis:

- Total float
- Free float
- Finish total float
- Finish free float
- Relationship total float
- Relationship free float

Each type of float is discussed in the following sections.

Total Float

Total float is simply the difference between the early and late start dates, measured in working periods of the activity calendar. In the absence of target dates and multiple calendars, there is at least one path through the project with exactly zero total float, with all other activities having zero or positive total float.

Imposing unattainable target finish dates or an unattainable project completion date complicates the issue by creating negative float. Negative float appears in cases where the late dates are earlier than the corresponding early dates.

Multiple calendars may also complicate the issue by generating additional float. For example, an activity using a 7-day work week and ending on a Saturday may have 1 day of total float simply because the calendar on the next activity prevents it from starting until Monday.

Free Float

Free float measures the maximum delay for an activity that does not cause the delay of any subsequent activity beyond its early dates. In the absence of finish targets, free float is always less than or equal to total float. Unlike total float, however, free float can never be negative.

Finish Float

In addition to the calculation of total and free float based on start dates, Open Plan also calculates values for finish total float and finish free float. For most activities, float calculated from either start dates or finish dates yields identical results. For discontinuous activities and subprojects, changes in the calculated duration can result in one float value being computed from the difference between early and late start dates and a different float value being computed from the difference between early and late finish dates.

Relationship Float

Open Plan also calculates both total and free float for each relationship in the network. Relationship total float is the amount by which the lag on a relationship would have to be increased to cause a delay in either the project completion or a late target date.

Relationship free float is the amount by which the lag on a relationship would have to be increased to cause the delay of a successor activity. This value can be used to determine which relationship is responsible for controlling the critical status of the successor activity.

Criticality

The concept of criticality is closely related to float. In the simplest case, an activity is usually defined as critical if its total float is zero. Real life situations can be more complicated. During time analysis, Open Plan calculates a field named **Critical Flag** that is very useful in reporting. (Of course, you can always report on criticality using the various float fields themselves if you wish.)

The **Critical Flag** field can have one of the following four values:

- **Critical** — The activity is not complete and has a total float that is zero or negative but is greater than the lowest total float in the network.
- **Most Critical** — The activity is not complete and has a total float that is zero or negative, and is also equal to the lowest total float in the network.
- **Controlling Critical** — The activity, while not critical itself, controls a successor with some type of critical status (that is, the Critical Flag field of the successor is set to either Critical, Most Critical, or Controlling Critical). This occurs, for example, if an activity with a 7-day per week calendar finishes on a Friday and its successor is constrained to work Monday through Friday only. The successor will start on the following Monday, and even if this successor has zero float, the predecessor will have two days of float.
- **Not Critical** — All other activities, including completed activities.

As a result, using the **Critical Flag** field to determine criticality differs from simply checking to see if total float is equal to or less than zero in the following ways:

- The **Critical Flag** field automatically checks both the start and finish total float values and uses the smaller of the two.
- Completed activities are not flagged as critical.
- In cases where there are many activities with negative float, it is possible to identify activities that have the minimum total float for the entire project.
- It is possible to identify activities that control the critical path in the network but which have more float purely because of the interference of calendars.

Time Analysis and Subprojects

Open Plan allows you to specify relationships between activities at any subproject level or between activities on different subproject levels. You can also specify relationships between activities that are in different subprojects. The richness inherent in this capability raises questions as to exactly what the calculated dates for subprojects might mean in certain circumstances. The following sections explain how Open Plan performs these calculations with regards to both internal and external subprojects.

The Treatment of Internal Subprojects

If an activity has been defined as an internal subproject, Open Plan treats any activities whose activity IDs start with the ID of the subproject as children of that subproject. For example, assuming the subproject has an ID of SUB1, Open Plan would treat an activity with the ID of SUB1.001 as a child of SUB1. This child activity could, in turn, be a parent of children such as SUB1.001.001. In this way, one can build a hierarchy of subprojects, the depth of which is limited only by the maximum length of the activity ID.

Because it is possible to enter relationships between different subprojects, these relationships should affect the start dates of their component activities at lower levels. In Open Plan, this is implemented during time analysis by inferring a start-to-start relationship from a parent to each of its children, and a finish-to-finish relationship from each child to its parent. Thus, no activity can start earlier than its parent, and the parent cannot finish until all its children are complete.

The flow of information during the forward pass is such that the early start dates of the children depend upon the parent while the early finish date of the parent depends upon the children. Likewise, during the backward pass, the late finish dates of the children depend upon that of the parent while the late start date of the parent depends upon the late start dates of the children.

As a result of this approach, the early and late dates calculated for subprojects may not always represent a summarization of the early and late dates of the child activities. This is due to the fact that Open Plan allows you to further constrain the children, either by means of target dates or by relationships that directly connect a child activity to an activity in another subproject. In this case, it is possible that all of the children of a particular subproject may, due to other constraints, start later than the earliest date permitted by the early start of the subproject. In such a situation, the early dates calculated for a subproject would differ from the earliest dates of the child activities belonging to that subproject. Since many planners wish to see summarized rather than calculated dates for subprojects, Open Plan has a default option to show summary dates in subprojects. You may disable this option by changing a setting on the **Options** tab of the **Time Analysis** dialog box.

The Treatment of External Subprojects

It is possible to open a master project without opening one or more of the external subprojects assigned to that project. In this situation, remember that Open Plan takes into account only the information that is available at the time that time analysis occurs. Thus, if you do not open an external subproject attached to a master project, Open Plan performs time analysis by treating the early and late dates stored for the external subproject as though they were On Target dates and does not write project dates back to the subproject when the processing is complete.

If a project contains a relationship to an activity in an unopened external subproject, Open Plan creates a foreign activity to represent the other end of the interproject relationship. A foreign activity acts as a placeholder activity representing the activity in a different project. Open Plan uses the following conventions when calculating dates for projects containing foreign activities:

- During the forward pass, an early date from a foreign activity is treated as though it were a Not Earlier Than target date.
- During the backward pass, a late date from a foreign activity is treated as though it were a Not Later Than target date.

The dates of the foreign activities themselves are not affected by time analysis.

If no dates exist for a foreign activity, Open Plan treats the activity as any other activity and stores a message to that effect in the session log.

Hammock Activities

A hammock activity is used to summarize a number of activities into one activity. A hammock is used for summary reporting to measure the time difference between the early dates of two activities and is provided in Open Plan primarily because this is a feature traditionally found in project management systems. For summarizing project progress, hammocks are not as flexible as rolling up groups of activities based on a structure implicit in the activity ID or code fields.

Hammock activities, by definition, cannot affect the scheduling of other activities. Thus, the dates calculated for hammocks can play no part in the calculation of dates in the rest of the project. The early and late dates for hammocks are calculated in exactly the same manner as dates are calculated for non-hammock activities. Hammocks can have relationships to multiple start and finish activities, and some or all of these can themselves be hammocks. It is not possible to have a hammock as the predecessor of a non-hammock activity.

A hammock is taken as running from the earliest early start to the latest early finish of the activities included in the hammock. Since hammock durations are determined by the activities defining the hammock, any duration information entered for the hammock by the user is ignored during time analysis. Instead, the duration of the hammock is calculated as the interval (in the specified calendar) between the start and the finish dates of the hammock activity, inclusively.

The ability to specify a calendar for a hammock provides a simple means of calculating the elapsed duration of a group of activities using different calendars.

Link/Unlink Activities

Open Plan allows you to add and remove multiple relationships of the same type between activities conveniently using the Link/Unlink Activities dialog box. When you click the **Link** button, Open Plan automatically creates relationships between selected activities using the selected type, lag, and calendar. It creates links in the order the activities display on the current view.

To remove relationships between linked activities in Open Plan, click **Unlink**.

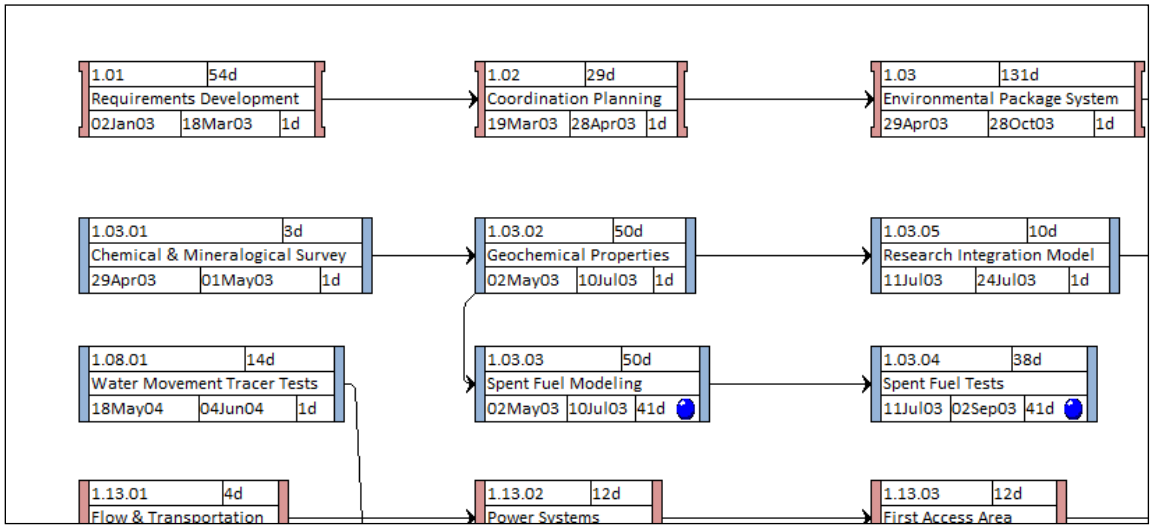
Select one of the following relationship types:

- Finish to Start
- Start to Start
- Start to Finish
- Finish to Finish

You can also use the Add Relationship dialog box to define relationship between activities. If relationships already exist between the selected activities, Open Plan would not allow you to modify the existing relationships.

Logic Trace

Open Plan allows you to outline the path of activities leading to a selected activity using the Open Plan Logic Trace Facility Dialog Box tool. It walks you through project relationships to find what activities are related to a starting act.



To view only the traced logic, from the **View** tab select **Placements** and use the TracedLogicOnly filter. Use the LogicTraceOther filter to view everything but the activities in the traced logic.

Schedule Driver Analysis

Open Plan helps you determine the most critical paths to any point in a project using the Schedule Driver Analysis tool. It allows you to specify up to 20 paths for the tool to identify. It identifies activities that are driving the dates for a target activity. It can recognize multiple paths through the activities preceding the target activity and groups the activities based on their total float. This tool does not only identify the activities directly responsible for the target activity dates but also identifies other potential controlling chains.

When you run the process, you can use the returned data to sort or filter any of the Open Plan views. In addition, Open Plan includes two views specifically designed to take advantage of the critical path data returned: a sub-sectioned barchart view and a zoned network view.

Target Dates

To understand the activities driving the dates for a particular project, Open Plan ignores the effect of target dates on other activities in the project. In some cases, however, Open Plan still needs to consider other deliverable; thus, removing other target dates is optional.

The Schedule Driver Analysis tool works by looking at the float and status of the selected activity's predecessors. The most critical predecessor is determined and the process repeats with that activity. A path is finished when there are no more activities to follow, Time Now is reached, or if there is a change in float.

Custom Reports

If you have the Milestones Professional® (MP) application installed, you can generate custom reports that take full advantage of its graphic display ability. Three report templates included with the Schedule Driver Analysis tool allow you to create the following reports:

- A Barchart report that documents all critical paths
- A separate Path Details report for each critical path you instruct the tool to identify
- A Summary Path report that summarizes the current activities on each path

The Effect of Progress Information

The presence of progress information has the effect of complicating certain calculations performed by Open Plan during time analysis, risk analysis, and resource scheduling. This section explores the effect of activity progress information on these types of calculations and concludes with a brief description of the effects of resource progress information.

The Effect of Progress on Time Analysis

A central concept to the approach Open Plan uses when considering progress information is that time analysis is concerned only with the remaining duration of each activity. In the case of completed activities, this remaining duration is always zero. For in-progress activities, the calculated early dates are always later than or equal to Time Now since these dates relate only to the remaining portion of the work.

There are a few additional ways in which progressed activities are treated differently from activities that have not been progressed:

- Targets start dates are ignored for in-progress and complete activities.
- Target finish dates are ignored for complete activities.
- Lags from the start of in-progress or complete activities are measured from either the actual start date or an implied actual start date.
- Lags from the end of completed activities are measured from either the actual finish date or an implied actual finish date.
- Lags that are expressed as a percentage completion of the predecessor activity are based on the date on which this percentage completion is expected to occur. (See below for more information on this point.)
- Completed activities are set to a non-critical status.

Activity Status and Remaining Duration

During time analysis, the main effect of progress information is to modify the duration of the activity. Normally, the remaining duration for an activity in progress will be less than its original duration. (For completed activities, the remaining duration is, by definition, zero.)

As a general rule, it is always a good practice to enter an actual start date for any activity that has been started. In addition to providing a useful historical record, the presence of an actual start date ensures that any relationship lags will be measured from that date. (In the absence of an actual start date, Open Plan uses an estimated start date, which may change from session to session as Time Now advances.) Likewise, an actual finish date should be entered when an activity is complete.

Open Plan calculates time analysis results differently depending on the options you choose on the Time Analysis Options tab:

- **Actual Date Option** — When this option is selected (the default), Open Plan displays actual dates of in-progress and complete activities (instead of calculated early and late dates) on dialog boxes and views. This does not, however, affect the time analysis calculations unless the new Observe Actual Future Dates option (see below) is also selected and actual dates occur after Time Now.

  To have Open Plan display the calculated early and late dates, regardless of activity status, clear this option.

  This allows you to see the planned restart date for in-progress out of sequesnce activities.

- **Observe Actual Future Dates** — When you select this option, Time Analysis respects all future Actual Start and Finish dates.

  This can be useful when your formal progress reporting cycle is long, say monthly, but you collect progress as it happens which may mean you have actual dates in your schedule that are beyond Time Now.

  For example, Time Now is 11/1. Task A is a 10-day task with a planned start on Time Now (11/1) and a planned finish of 11/14. On Monday 11/5, you report that you have finished Task A and you set the Actual Finish to 11/5.

  - If the Observe Actual Future Dates option is not selected, the system sees this as invalid because it is after Time Now. Therefore the activity is still planned to finish on 11/14 and successor activities can start on 11/15.
  - If the Observe Actual Future Dates option is selected, the system sees the activity as completed on 11/5, and successor activities can start on 11/6.

If the project is going according to plan, actual start and finish dates may be all the progress information you need. Schedule variances are a common occurrence. As a result, Open Plan offers a number of different ways for you to communicate a best estimate of the remaining duration for an activity that is in-progress. While it is recommended that these estimates be used to augment the actual dates, Open Plan also accepts these estimates (in the absence of actual dates) as an indication of progress.

There are a number of ways to indicate progress on an activity in Open Plan:

- Marking an activity as complete
- Entering an actual start date
- Entering an estimate of remaining duration, elapsed duration, or percentage completion
- Entering an expected finish date

A common practice for in-progress activities is to report an Actual Start date along with an Expected Finish date. Open Plan then calculates the remaining duration based on the Expected Finish – Time Now (or the Expected Finish – Actual Start if the actual start date is beyond time now and the Observe Actual Future Dates option is enabled). When the activity completes, you add the Actual Finish date

Because these different indications of progress can, in some circumstances, be inconsistent with one another, it may be useful to explain exactly how Open Plan determines the status of an activity and its remaining duration.

As an aid to diagnosis and reporting, Open Plan calculates two fields to summarize how the progress information was interpreted during time analysis — Computed Remaining Duration and Computed Status (Planned, In Progress, or Complete).

Marking an Activity as Complete

Open Plan considers an activity complete if either of the following conditions apply:

- The actual finish date prior to Time Now was entered
- The activity status was changed to Complete

If an activity is marked complete by either method, Open Plan calculates the remaining duration of the activity as zero.

If no actual finish date was entered for the activity, Open Plan assumes the activity finish date to be the period prior to Time Now. If no actual start date was entered for the activity, Open Plan estimates a start date by subtracting the original duration from the actual or assumed finish date.

Using Actual Start Dates

If an actual start date has been entered and there is no other indication of remaining duration, the elapsed duration for the activity is deduced by subtracting the actual start date from Time Now. This elapsed duration is then subtracted from the original duration to calculate the remaining duration. (Note that these calculations take the activity calendar into account.)

If the calculated remaining duration is zero, the activity is treated as complete as indicated in the time analysis session log. If the calculated remaining duration results in a negative value, the remaining duration is taken as zero and the activity is considered complete.

Entering an Estimated Duration

Open Plan allows you to enter an estimated duration for the activity using one of three methods:

- By entering a remaining duration
- By entering an elapsed duration
- By entering a completion estimate as a percentage

If you enter a remaining duration for an activity, Open Plan assumes the remaining duration to be that value. If you enter an elapsed duration for an activity, the remaining duration is calculated by subtracting this value from the original duration.

If you enter an estimate of the activity's completion status as a percentage, Open Plan calculates the remaining duration by subtracting this value from 100 and applying this remaining percentage to the original duration. If the remaining duration calculated in this manner is zero, the activity is treated as complete as indicated in the time analysis session log.

If the actual start date is not set for the activity, Open Plan will estimate a start date by subtracting the calculated elapsed duration from Time Now, taking into account the activity calendar.

Using an Expected Finish Date

The expected finish date is a convenient way to specify when you expect an activity to be completed. To obtain an estimate of the remaining duration from this type of progress information, Open Plan subtracts Time Now from the expected finish date (taking into account the activity calendar).

If the remaining duration calculated in this manner is zero or negative (which generally indicates that the expected finish date is before Time Now), the activity is treated as complete as indicated in the time analysis session log. In this case, the expected finish date is treated as if it were an actual finish date.

If the actual start date is not set, an implied start date will be calculated by subtracting the elapsed duration from Time Now, which is the same as subtracting the original duration from the expected finish date, taking into account the activity calendar.

Note, however, that the existence of the expected finish date does not, in itself, indicate that the activity is in progress. If the actual start date is not set and the implied start date calculated for the expected finish is after Time Now, the activity is not treated as in-progress. Instead, the expected finish date is treated as an On Target finish target date (but only if there is not an explicit finish target).

Special Cases of Progress Information

Three special cases of progress information require additional discussion:

- Out-of-sequence progress
- Lags based on percentages

- Progress for subprojects and hammock activities

Each of these cases is discussed in the following sections.

Out-Of-Sequence Progress

It is possible to specify that an activity has been progressed when one or more of its logical predecessors have not been completed. This is referred to as “out-of-sequence progress” in Open Plan.

Open Plan allows the user to select one of the following options for calculating dates for activities reporting out-of-sequence progress:

- **Ignore Positive Lag of Predecessor** — Ignore positive lags on relationships leading into out-of-sequence “events” (that is, the start of an activity that is marked as in-progress or the finish of an activity that is marked as complete)
- **Observe Positive Lag of Predecessor** — Observe all relationships and lags, including positive lags, leading into out-of-sequence events
- **Ignore Predecessor Relationship** — Ignore all relationships and all lags leading into out-of-sequence events (including the finish of an in-progress discontinuous activity)

If you select one of the first two options, Open Plan schedules the remaining portion of the out-of-sequence activity so as not to violate the logic of the network. For example, assume that an activity is marked as complete but has a predecessor with a remaining duration of 10 days. In this case, the early start of the out-of-sequence activity will be calculated as 10 days after Time Now. This, in turn, will affect the dates calculated for any successor of the activity.

If you enable the **Actual Date Option** for time analysis, all early and late dates are replaced in cases where actual dates are available. The use of this option does not change any calculations performed by time analysis with regards to out-of-sequence progress, nor does it alter the effects of out-of-sequence progress on unprogressed activities. As a result, the use of the **Actual Date Option** may inhibit the diagnosis of scheduling problems where there is out-of-sequence progressing.

Lags Expressed as Percent Complete

When a relationship has its lag expressed as a percentage completion of its predecessor and when that predecessor is in progress, the date on which the successor may start is calculated as described below.

If the predecessor has already achieved the required percentage completion, the successor (in the absence of any other constraints) may start on Time Now. Otherwise, Open Plan calculates how much time needs to be worked on the predecessor before that percentage completion is realized. The successor is not allowed to start until this time has elapsed from the early start date of the remainder of the predecessor activity, even if this event is delayed due to out-of-sequence progressing.

In computing the percentage completion, Open Plan takes the elapsed duration and the remaining duration into account and, of course, performs all calculations taking the activity calendar into account.

Progress with Subprojects and Hammock Activities

Both subprojects and hammock activities summarize data from lower levels. Their durations are calculated from the lower-level activities rather than being specified by the user, and by the same token, any progress information is also inferred from these lower levels. (In fact, Open Plan ignores any durations or progress information entered for parents or hammocks.)

The summarization of progress for parents and hammocks is reflected in the values of the **Computed Status** and **Computed Remaining Duration** fields as well as the actual start and finish dates on the activities.

The rules for summarizing progress information are similar for both types of activity. For subprojects:

- A subproject is considered to be complete if all of its child activities are complete.
- A subproject activity is considered to be in progress if one or more of its child activities are either in progress or complete.
- The actual start of a subproject is equal to the earliest of the actual starts of any of its children.
- The actual finish of a subproject is equal to the latest of the actual finishes of its children but only if all the children are complete.

For hammock activities:

- A hammock activity is complete if all of its end activities are complete.
- A hammock activity is in progress if one or more of its start or end activities are either in progress or complete.
- The actual start of a hammock activity is equal to the earliest of the actual starts of any of its start activities.
- The actual finish of a hammock activity is equal to the latest of the actual finishes of its end activities but only if all the end activities are complete.

Note that the case of subprojects is somewhat more complex than that of hammocks, as the actual dates may affect the time analysis. For example, if there is a lag on a start-to-start relationship between two subprojects, the date from which this lag is measured will be the earliest of any actual start dates of the children of the predecessor subproject.

The Effect of Progress on Risk Analysis

In most cases, project managers will use risk analysis processing on a project that has yet to begin in order to establish its viability. Situations may arise where it is desirable to conduct risk analysis on projects that have already recorded progress in order to accurately gauge the reliability of the remainder of the project schedule. This section provides information on how risk analysis treats the effect of project progress on duration probability.

For unprogressed activities, Open Plan applies the specified distribution probability information for activity duration (that is, the distribution shape and parameters) to the original duration. Once progress has been recorded, risk analysis processing is applied only to the remaining duration of the activity.

As in time analysis, risk analysis calculations can recognize activity progress (and thus calculate the remaining duration) in a number of different ways from activity information:

- An actual start date
- An actual finish date
- The activity is marked as complete
- An estimate of the remaining duration
- An estimate of the percent of duration complete
- A value for the elapsed duration
- An expected finish date

If an activity is complete (that is, either an actual finish date has been entered or the activity has been marked as complete), the remaining duration of the activity is considered zero. There is no need to sample from the original duration probability distribution.

If an estimate of the remaining duration has been entered, the remaining duration is fixed at the value specified by the user, and again, no sampling is necessary.

Unlike standard time analysis processing in Open Plan, the results of risk analysis processing are not used to change the completion status of an activity.

The Effect of Progress on Resource Scheduling

The existence of activity progress has a number of implications for resource scheduling. First of all, resource scheduling looks only to the future; it schedules only the remaining work and does not consider anything that occurred before the status date (Time Now). By definition, therefore, it does not schedule any resources for activities that are complete. (The network logic is retained so that if these activities have been completed out of sequence, the existence of a completed activity can affect the scheduling of subsequent activities, just as it does in time analysis.) As in time analysis, negative lags are ignored for any relationship leading into the start of an in-progress or completed activity or into the finish of a completed activity.

On the other hand, activities that are not yet started are treated in exactly the same way as they would be in an unprogressed project.

The remaining issues are, therefore, largely concerned with activities that are in progress:

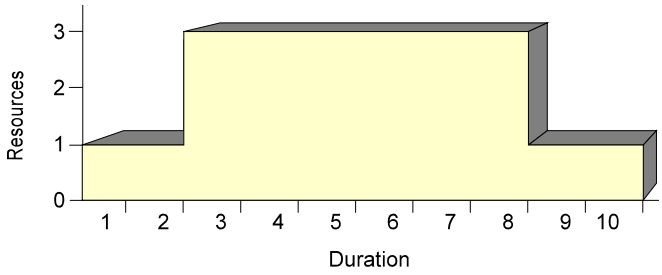
- How the system interprets a resource profile relative to the remaining duration of an activity
- The implications of progress for activity splitting

Attention: There is also an option to give priority to activities that are in progress, which is discussed in Resource Scheduling Calculations.

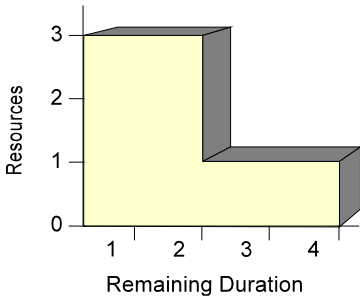
Resource Profiles for Remaining Duration

With regards to progressed activities with resource requirements, Open Plan assumes that the resource usage has been exactly as determined by the originally planned profile. In other words, Open Plan maps the original profile for each resource against the original duration, and uses the last part of that profile to match against the remaining duration of the activity.

An example will make this clearer. Assume that the activity was originally intended to take 10 days with the following resource profile:



Further assume that the remaining duration is now 4 days. The resource profile is assumed to be the same as the final 4 days of the original profile:



If the remaining duration is longer than the original duration, the original profile will be mapped onto the latter part of the remaining duration so that there will be no resource requirement for the beginning of the remaining duration of that activity. In such a case, there has obviously been a significant deviation from the plan, and one would expect to re-specify the resource requirement profile as well as redefine the activity duration.

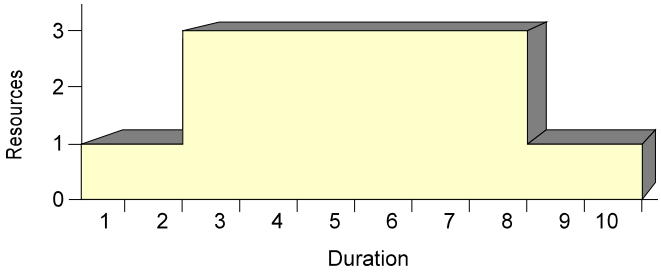
The Effect of Resource Progressing

If you have entered an estimate of the remaining usage of resources for an in-progress activity, Open Plan uses this estimate to determine the remaining requirement for the resource.

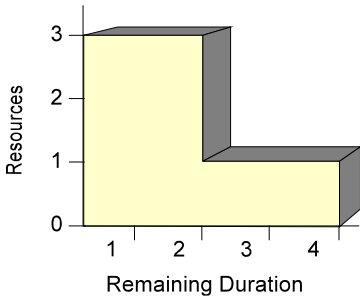
This occurs only if the activity is in progress. For activities that have not started, Open Plan uses the original assignment; for completed activities, the resource requirement is always assumed to be zero.

The profiling of this remaining requirement over time is done by first matching the last part of the original profile to the remaining duration, and then applying the remaining resource assignment to that profile on a pro-rata basis.

To illustrate how Open Plan does this, return to the example of a 10-day activity with the following resource profile:



Again assume that the remaining duration is now 4 days and that the resource profile is the same as the final 4 days of the original profile:



Now assume that you have indicated that the remaining requirement for the resource is 10 units. Since the original profile specified a requirement of 8 units for that portion of the activity duration, it is necessary to increase the requirement on each day by 20 percent. This causes rounding problems, which Open

Plan handles in the same manner as in the case of activity stretching. In the current example, this results in resource requirements of 4 units, 4 units, 1 unit, and 1 unit respectively over the 4 days.

> **Attention:** For information on activity stretching, see Resource Scheduling Calculations.

Finally, note that there is a potential problem in that the remaining profile for a resource may be zero. Suppose, for example, that the original profile called for the resource to be used in the first 3 days of a 10-day activity, and we are now in the last 4 days. Of course, if everything had gone according to plan, the remaining requirement for this resource would be zero. However, suppose that you have entered a positive amount for the remaining requirement at this point. In such a case, Open Plan assumes that the entire amount of the remaining resource is to be used at the start of the remaining duration. (If this is not what you intend, you can update the profile to reflect a modified assignment.)

The Splitting of In-Progress Activities

Activity splitting does not take into account any splits that may have occurred in the past since it has no way to know about these. The limit on the number of splits is, therefore, interpreted as if the previous work was all in one piece.

A potential problem can arise if it proves impossible to start the remaining part of the activity on the first working period after Time Now (which could be either because of the activity having been progressed out of sequence or because of the unavailability of resources). In this case, Open Plan splits the activity, and this split is counted toward the total number of splits allowed for the activity.

This action may affect the ability of the system to split the activity further in future scheduling. For example, assume that an in-progress, splitable activity has the maximum number of splits set to 2. If the remaining part of the activity is scheduled consecutively with the work already done (that is, on the first work day after Time Now), it is permissible to split it later if necessary. However, if the activity cannot be scheduled immediately following Time Now, a split is deemed to have taken place, and subsequent splits are not allowed.

Note that such a split is inevitable and will occur regardless of the minimum split duration. In fact, Open Plan will split the activity in this circumstance even if the activity is not defined as splitable. To reduce the likelihood of this circumstance occurring, use the in-progress priority processing option when performing resource scheduling.

Resource Scheduling Calculations

The primary objective of resource scheduling is to produce a project schedule that takes into account not only the network logic, but also the availability of resources. This is potentially a very large combinatorial problem, and no commercially available project management system claims to provide optimal solutions. Rather, they employ heuristic algorithms designed to find acceptable solutions in a reasonable amount of computing time, even for very large problems.

As with the case of time analysis, the relatively straightforward interface of the resource scheduling function in Open Plan does not adequately suggest the complexity and sophistication of the resulting calculations. This section is designed to help users obtain a deeper understanding of these calculations through discussions of the following topics:

- Resource scheduling methods
- The effects of activity and resource attributes
- The interpretation of availability and assignment data
- The effect of processing options

- Resource assignment profiles
- Controlling priorities
- Resource substitution
- Reserving resources

This section concludes with a discussion of the interpretation of resource scheduling data.

Resource Scheduling Methods

There are two basic methods used in Open Plan resource scheduling calculations:

- Resource-limited scheduling attempts to schedule the project in the shortest possible time without exceeding the resource availabilities (though this can be modified selectively using thresholds).
- Time-limited scheduling (sometimes known as leveling) schedules the project within a specified time, exceeding resource availability where necessary but attempting to minimize the extent of any such overloads. (This also can be modified using the smoothing option.)

Clearly, if the system is able to schedule the project within the time constraints and without exceeding any of the resource availabilities (in other words, if there is no conflict between these objectives), both methods of scheduling will produce similar results. The methods differ in how they deal with any conflict between these constraints on time and resources and more particularly how they behave when a particular activity cannot be scheduled within its late dates without exceeding the available resources.

Support for resource scheduling with regards to subprojects and hammocks is limited. Open Plan takes note of the resource usage in these cases, but no attempt is made to alter the dates of such activities based upon resource availability. The resource usage is noted when the dates are known — that is, after dealing with children (of subprojects) or controlling activities (of hammocks) — and may result in overloads.

The Effect of Activity Attributes

Open Plan supports special activity attributes to facilitate the scheduling of particular activities. Each of these strategies (splitting, stretching, and reprofiling) can be regarded as methods of improving the results of the standard scheduling algorithm by relaxing the rules in a particular way. The fourth attribute, Immediate, restricts the operations of resource scheduling by forcing Open Plan to schedule an activity as early as possible.

Each attribute is discussed in the following sections.

Activity Splitting

If you have designated an activity as splittable, the activity can be scheduled in segments, the gaps between the segments being forced by the non-availability of a resource rather than by the calendar for that activity. This non-availability can, in turn, be due either to the originally specified availability profile (for example, using a resource calendar) or to that resource having been preempted by another activity.

An activity will be split only if it has been specified as splittable and generally only if doing so improves the scheduled finish date of that activity. (There is one exception to this, discussed at the end of this section.)

When an activity is split, it is as if the gaps between the scheduled segments simply do not exist. The total number of working periods in all the segments adds up to the specified duration, and the resource requirements for each period of its duration are unchanged. For example, if a 10-day activity is split into two 5-day segments, the resource requirement for the first day of the second segment is the same as was specified for the sixth day of the original duration.

Activity splitting is over and above any non-working periods due to the calendar of the activity. A single split segment can span such non-working periods, and the length of the segment is measured in this calendar. Obviously then, splitting an activity results in the time between the scheduled start and the scheduled finish of an activity being greater than its duration, even after taking into account the activity calendar.

The allowability of splitting is specified on an activity-by-activity basis, and the user can also control the extent of the splitting by specifying two parameters: the minimum length of each segment, and the maximum number of segments. These two parameters may seem to have similar effects. In general, it is preferable to use the first one rather than the second. The reason for this is best illustrated in another example.

Suppose an activity has a duration of 15 days, and we expect to have to split it into 3 segments. We could either limit the minimum length of a segment to 5 days or else limit the maximum number of segments to 3. With the first approach, we cannot have more than 3 segments, and we are further constrained to make them all the same length. If we use the second approach, it will generally take longer to find an acceptable solution, simply because there are more choices to consider.

In the interest of processing speed, it is generally advisable to apply a realistic limit on the minimum length of each segment of work, whether or not you also choose to limit the maximum number of segments. In the above example, you could allow a little flexibility by limiting the minimum length of a segment to 4 days rather than 5 days. In the absence of such a limit, Open Plan may make a large number of abortive attempts to schedule based on the first two segments being of length 1 day each, 1 day and 2 days, 2 days each, and so on, all of which subsequently get abandoned because the last segment is too long to be scheduled.

Note that this is especially true when the duration and the maximum number of segments get very large. Imagine an activity with a duration of 150 days that can be split into up to 30 segments. Relying solely on a limit of 30 segments may result in Open Plan doing a great deal of abortive work because of the many potential ways in which the activity can be split. Also, in each case, Open Plan cannot tell that the last segment is impossible to schedule until it has scheduled the first 29 segments. Setting a minimum length of 4 days, for example, greatly limits the number of combinations, and in particular those combinations that leave a very large residue to be scheduled as the 30th segment.

Although Open Plan will never split an activity unless the result is an earlier scheduled finish date than the date that can be accomplished without splitting, there is a possibility that an activity will not be split in such a way that it minimizes the scheduled finish date. To illustrate how this can happen, consider an activity with duration of 10 days that is allowed to be split into a maximum of 2 segments. (In accordance with our previous advice, we may also limit the minimum duration to 3 days, although with only 2 segments this limit is much less important.)

Suppose further that at the point in time when we come to schedule this activity, the resource it requires is available for 3 days, followed by a gap, then for 5 days, another gap, another 5 days, another gap, and finally for 7 days.

Open Plan will schedule the first 3 days to take advantage of the earliest availability because a 3-day split is allowed. It will then be left with a 7-day segment, which it cannot split further because it is limited to 2 segments. This second segment will have to be delayed until the resource is available for 7 consecutive days. In this case, delaying the start so that the first segment could be 5 days long would enable the second segment, also 5 days long, to be scheduled earlier. Because it does not do an exhaustive search, Open Plan will not find this better solution.

Had the split been limited only by the minimum length of each segment, the above situation could not apply. The first 3 days would be scheduled as before, then 4 days would be scheduled during the first 5-day availability period, and the final 3 days would be scheduled during the second 5-day period, resulting in a scheduled finish date 2 days earlier than the best solution possible with only 2 segments. There is no better solution, subject to the stated constraints.

In summary, it is always desirable to limit the minimum length of each segment, either in combination with a limit on the number of segments or alone. Using the length limit as the controlling limit typically improves both processing speed and schedule efficiency.

Note: To see exactly how an activity has been split, stretched, or reprofiled (that is, into how many segments the activity has been split and the length of each segment), examine a resource histogram or a crosstable view based on scheduled dates. You could also look at the barchart views directly.

Activity Stretching

Activity stretching is another strategy that can be employed to facilitate the scheduling of individual activities. If you designate an activity as stretchable, then the activity can be scheduled to use a reduced level of a resource by stretching the duration, up to a user-defined maximum.

The stretching is done on a linear basis. Stretching is only done if it will result in an earlier scheduled finish date than can otherwise be achieved. If an activity is stretched to take one extra time period, the requirements for each period are reduced by a prorated amount. This generally causes rounding problems, but these problems are resolved by Open Plan in the same way as it handles a total resource that cannot be divided exactly by the duration — namely, by keeping the cumulative usage as close as possible to the correct figure.

To see how this rounding works, suppose that an activity has a duration of 6 days and requires 5 units of a resource for a total of 30 units. If the system is unable to schedule the activity as specified, it can try to stretch the duration of the activity by 1 day (from 6 days to 7 days). The resource levels in each of the 7 days would be as follows:

- Day 1 — 4 units (30/7)
- Day 2 — 5 units ((30*2/7) – 4)
- Day 3 — 4 units ((30*3/7) – 9)
- Day 4 — 4 units ((30*4/7) – 13)
- Day 5 — 4 units ((30*5/7) – 17)
- Day 6 — 5 units ((30*6/7) – 21)
- Day 7 — 4 units ((30*7/7) – 26)

Note that the stretching has not reduced the peak requirement of 5 units, so this approach may or may not help in the scheduling in this particular case. If stretching the activity to 7 days fails, the system will try stretching the activity to 8 days, and then to 9, up to the maximum number of days specified. At 8 or 9 days, the peak requirement is reduced to 4, while at 10 days the profile becomes uniform again, with a requirement of 3 units per day.

In defining the maximum stretch duration, it is desirable to make the allowable duration large enough to make a real difference, which generally (but not necessarily) means large enough to reduce the peak requirement.

Finally, note that the maximum stretch duration is still measured in the calendar of the activity, and an activity will never be scheduled or use resources during non-working periods on that calendar.

Activity stretching can be time-consuming due to the need to consider each stretched length in conjunction with each possible start date; the reprofiling technique described in the next section is both more flexible and more efficient.

Activity Reprofiling

Reprofiling enables Open Plan to satisfy the total resource assignment for an activity in any way consistent with the availabilities. Thus a requirement of 20 man-hours of a particular resource might involve using the resource for 5 hours the first day, 8 hours the second day, and 7 hours the third day. The user does retain some control. A maximum duration can be specified.

The reprofiled requirement cannot exceed the original requirement on a cumulative basis starting from the scheduled start.

For example, suppose that a 3-day activity requires 5 units of a resource for each day of its duration for a total of 15 units. The activity cannot be assigned more than 5 units on the first day. Suppose that there are only 3 units available on the first day and they are assigned to the activity. On the second day, the activity can be assigned up to 7 units, allowing it to catch up with the cumulative total of 10 units. Suppose that there are still only 3 units available, however, and these are assigned. On the third day, the activity will be allowed to use up to the remaining total of 9 units. If 9 units are not available, then the activity will be stretched up to the maximum duration, using the resources as they become available.

Limiting the resource usage to the specified cumulative profile is designed to prevent the first activity scheduled from simply using all the resources it needs on the first day. A user who wants to allow more flexibility can override this limit to any desired extent by modifying the resource profile itself. There is also an option on the **Advanced** tab of the **Resource Scheduling** dialog box to limit reprofiling to the original level. The most extreme case would be to specify that all the resources are required on the first day. A more sensible approach would be to increase the requirement slightly on the first day and to make a corresponding reduction on the last day.

The interpretation of when a reprofiled activity is deemed to have started can be confusing. This is because it is quite possible for the resource profile to be adjusted so that the usage is zero for the first period or for the first several periods. The system assigns a scheduled start date that may seem arbitrary but is in fact consistent with the rules described above as well as with the fact that the resource must be used during the activity duration.

This is best explained with the aid of an example. Suppose that a reprofilable activity requires a single resource at a level of 1 unit per day over its entire duration of 10 days. Suppose further that it is available for scheduling on January 1 but that the resource is not available until February 1, when it becomes available with a level of 3 units. The activity will use 3 units of the resource on each of the first 3 days of February, and 1 unit on February 4, for a total of 10 units. The scheduled finish date will be February 4, and the scheduled start date will be January 26, 9 days earlier.

Note that there is a period of 4 days during which the activity has started but no resources are used. This is more than merely a convention since the scheduled dates are the dates used to compute the start dates for the successors of the activity. The details of when an activity actually uses its resources are displayed in a histogram based on scheduled dates. As a convenience for certain reporting requirements, Open Plan also stores the date on which the first resource was used in the First Usage field on the activity table.

> **Note:** Because resource requirements can be profiled so as not necessarily to cover the start of an activity, it is possible for the first usage date to be later than the scheduled start date, even without reprofiling. Also, note that in the special case of an activity with no resource assignments at all, the first usage date is always set to the scheduled start date.

The Immediate Attribute

Activities assigned the Immediate attribute are treated first among the list of activities that are available for scheduling at any particular point in time. If there is more than one such activity in the list, the order in

which these activities are scheduled is not important, because in fact, all such activities will get scheduled on the earliest date possible regardless of whether this creates a resource overload.

In general, the Immediate attribute should be used sparingly. Its main purpose is to deal with situations where a sequence of activities needs to be scheduled without breaks between the activities. In such cases, the first activity in the string does not need to be classified as Immediate, but the remaining activities should be. As with normal activities, they cannot be scheduled before all of their predecessors, and they cannot be scheduled on non-work days. Therefore, there should typically be no other logic constraining these subsequent activities, and they would normally be on the same calendar as the first activity.

Note that although Immediate activities are scheduled without regard to resource overload, giving these activities the highest priority maximizes the probability that such an overload will be avoided, if necessary, by delaying other, lower-priority activities.

The Effect of Resource Attributes

Three attributes assigned to resources have a potential effect on resource scheduling calculations:

- Resource thresholds
- Consumable resources
- Perishable resources

Each attribute is discussed in the following sections.

Resource Thresholds

It is possible to combine the characteristics of resource-limited and time-limited scheduling by applying thresholds to one or more resource and then specifying a resource-limited schedule. Thresholds have no effect on time-limited scheduling as such but impart some of the characteristics of time-limited scheduling to resource-limited scheduling. In fact, it is possible to think of time-limited scheduling as entirely equivalent to resource-limited scheduling where all the thresholds are set to very high values.

A threshold is an amount by which a particular resource may be exceeded if necessary to stay within the late dates of the project, even though the resource-limited method of resource scheduling is used. When an activity cannot be scheduled within the time constraint, Open Plan first attempts to schedule it by exceeding one or more of the specified resource availabilities but only those for which a threshold has been specified and only to the extent of this threshold. If this is not possible, the activity is scheduled later, as it would be in the usual resource-limited mode.

Considerable control can be exercised over the scheduling algorithm through the use of thresholds. The user can control the extent to which resource availabilities are exceeded on the basis of each individual resource.

Consumable Resources

Regular resources generally represent personnel or pieces of equipment that are available for duty for certain specific periods of time. If the project is unable to use all the resources available on a particular day, this usage is lost forever. For example, if there are three electricians available for a week and you employ only two of them on Monday, this does not enable you to use four on Tuesday.

Open Plan supports a special type of resource called a consumable resource. Once a consumable resource becomes available, it remains available until it is used. Consumable resources are used to model resources like materials whose unused availability can be accumulated.

The data defining the availability of a consumable resource is also interpreted slightly differently than for regular resources. The **From Date** is the date from which the resource is available. The number of

resource units defined for the availability become available on that date and can be used at any time from that date onwards. The **To Date** of the availability is significant because it indicates the date after which the resource is assumed to be unlimited for the purposes of scheduling. For this reason, it is required. (If the resource availability is defined by multiple entries, the latest date is used by the system for this purpose just as with non-consumable resources.)

Consumable resources can also be created by activities. This is achieved by specifying a negative resource requirement. For example, assume that an activity has an assignment for a consumable resource of -2 units per day. The result would be that for every day of the duration of that activity, the number of available units for that resource would increase by 2.

Open Plan cannot correctly support the splitting of activities that use consumable resources. Resource may overload due to using the same availability for more than one split since Open Plan does not update the availability until the operation is complete. (For regular resources, of course, there is no conflict between different splits.) Furthermore, on a progressed project, it is important to modify the availabilities of consumable resources to indicate the true amount available at Time Now. This is because Open Plan makes no attempt to adjust these for any actual usage. Unlike regular resources, availabilities prior to Time Now are available for use on the remaining part of the project, so these should be adjusted to reflect the current situation.

Perishable Resources

Perishable resources are very similar to consumable resources, except that the **To Date** on each availability record indicates the date after which the particular batch of resources cannot be used.

Typically, you could specify the availabilities of perishable resources as a series of non-overlapping contiguous periods. One example of the use of perishable resources is to model funding of projects. It may be that in each financial year, a certain amount of funding is available. If it is not used during the year, it is forfeited. This would be represented by a different availability definition for each year.

However, there is no requirement for each unit of the resource to last for the same length of time nor for the availability records to be contiguous or not to overlap. Each record can be thought of as defining a batch of the resource, which becomes available on a certain date and which expires on a certain date. Open Plan will attempt to match each resource requirement with a batch in the most efficient manner. For example, if an activity requires 10 units of a particular perishable resource on June 1, Open Plan considers only the remaining availabilities of those resources available on or before June 1 and do not expire until after June 1. Assuming that the total is at least 10 units, it will first take the earliest expiring resources from the batch and so on. If two batches expire on the same date, Open Plan will use the one that became available later.

Open Plan cannot correctly support the splitting of activities that use perishable resources. Resource may overload due to using the same availability for more than one split since Open Plan does not update the availability until the operation is complete. (For regular resources, of course, there is no conflict between different splits.) Furthermore, on a progressed project it is important to modify the availabilities of perishable resources to indicate the true amount available at Time Now. This is because Open Plan makes no attempt to adjust these for any actual usage. Unlike regular resources, availabilities of perishable resources prior to Time Now are available for use on the remaining part of the project and so should be adjusted to reflect the current situation.

Scheduling Intervals and Mixed Time Units

In Open Plan, it is possible to define the durations of activities using time units ranging from minutes to months. This feature contrasts with many traditional scheduling systems that rely on the concept of a base time unit in which all durations are expressed.

The question of mixed time units has a direct bearing on resource scheduling calculations. In Open Plan, the availability of each resource is stored in an array, each element of which represents a uniform interval

of time. During scheduling, the assignment profile for each activity is compared with the data in this array. For the activity to be scheduled, Open Plan selects a trial start date and then compares the requirements with the availabilities in each time interval. When a suitable scheduled start date is selected, the availability of the activity is reduced.

To remove the ambiguity that could arise from the use of mixed time units, Open Plan allows you to specify a scheduling interval each time you perform resource scheduling. This scheduling interval determines the granularity to which the availability will be tracked. For example, if you select days as the scheduling interval, Open Plan checks only that the resource is available within a particular day before scheduling the activity. This is true even if the activity duration is measured in minutes. A 15-minute activity requiring one person for its entire duration requires 15 man-minutes during the day (or possibly even spanning 2 days). As long as that resource is available some time during that day (or days), the activity can be scheduled. When the activity is scheduled, the resource availability for that day is decremented by 15 man-minutes, but no information is retained about any particular 15-minute interval during the day.

The Interpretation of Availability Data

When you define the availability of a resource in Open Plan, you specify an availability level, a start and finish date for the availability, and the name of a calendar. For non-consumable resources, Open Plan interprets the data as follows: the resource is available at this level on every period between the two dates that is a working period in the specified calendar.

This sounds simple, but the lack of a natural base interval creates a situation that may require further explanation. Consider the example of a 15-minute activity discussed above. From that discussion, it should be clear that multiple 15-minute activities can be scheduled concurrently even though they all need one person and there is only one person available, provided only that the total availability during the day is not exceeded. But what exactly is this total availability? Put another way, exactly how many of these activities can be scheduled in a day?

The answer is that it depends upon the calendar specified for the availability record. If the calendar indicates an 8-hour day, then a level of 1 results in 480 man-minutes of that resource being available on each working day. (Of course, some days could have different numbers of working minutes in them, in which case the resource level on those days would differ.)

Resource availability records are additive, and of course there could be different calendars on different records.

The lack of a natural base interval creates a more serious problem with regard to thresholds since there is no calendar associated with this feature. Thresholds are therefore interpreted based upon the project default for the number of minutes in the scheduling interval. So, if you choose to schedule in days and if the number of hours per day defined for the project is 8, Open Plan translates an availability of 1 into 480 man-minutes.

For consumable resources, the level defines the number of units that become available on the start date. The finish date and the calendar are ignored, except that the finish date is used for the sole purpose of determining how far into the future to maintain usage information. The threshold represents the additional number of units that can be used in total throughout the project (that is, as if they were available at the start of the project).

The Interpretation of Assignment Data

When you assign a resource as a level (rather than a total) requirement, this assignment is interpreted in much the same way as availability levels, using the activity calendar. Thus, a level requirement of 1 person on an 8-hour-per-day calendar translates into 480 man-minutes per day.

The difficulty with requirements comes when they are specified as totals. What does a total of, say, 5 actually mean? This could be interpreted, for example, as either 5 man-hours or 5 man-days.

To remove this ambiguity, Open Plan uses the project defaults for the duration unit and for the number of minutes in that unit. Thus, if the default duration unit is days, then a total of 5 is equal to 5 man-days. If a day is defined as 8 hours, this is interpreted as a total requirement of 2400 man-minutes.

This means that the interpretation of a total requirement will not change if the duration of a particular activity changes (for example, from 5d to 40h) but will change if the project conversion settings are altered.

For consumable resources, the problem is reversed — levels expressed as total requirements are not ambiguous, but those expressed as level requirements are. As in the case of non-consumable resources, however, Open Plan refers to the project defaults to interpret the entry. Thus, for a consumable resource, a requirement expressed as a level of 5 will be interpreted as a requirement for 5 per minute, hour, day, week, or month, depending upon the project default settings.

Resource Assignment Profiles

It is often appropriate to assume that any resources required by an activity are required at an even rate throughout the duration of that activity. In such cases, it is necessary to indicate only the activity ID, the resource code, and the amount of the resource required each period.

In more complex situations, particular resources may be required for only a part of the duration of the activity, or the assignment may be unevenly distributed over the duration. Open Plan provides two ways of specifying such profiled assignments, and these can be used separately or in combination:

- You can specify a profiled assignment by entering multiple levels, offsets, and periods for the same resource.
- You can specify a predefined profile curve that Open Plan will use to spread the amount of the assignment.

In the first case, each assignment is interpreted as specifying a particular requirement of units for a time span starting at the start of the activity plus the offset and continuing for a specified duration. These calculations take into account the working and non-working periods defined in the activity calendar.

> **Note:** Open Plan does not permit an activity to use a resource outside the span of its duration. As a result, the offset duration of an assignment cannot be less than zero, and the combined duration of the offset and period cannot exceed the activity duration.

By combining assignments of the same resource, a profile of any desired complexity can be created. These assignments are treated cumulatively, so they can be defined to overlap if desired.

The second method of specifying a profile uses the predefined spread curves. Spread curves are defined in terms of percentages of the total to be distributed to each 10% of the duration of an activity.

The advantage of using spread curves is that it saves time, particularly if you are likely to alter the activity durations. (As offsets and periods are absolute, they generally have to be altered every time the duration changes.) On the other hand, as the curves are always defined on a 10-point scale and as activity durations will not normally be multiples of 10, the use of spread curves may produce rounding effects unacceptable to some users.

As stated above, the two methods can be used in combination, either on the same or on different requirement records. When a spread curve is specified in conjunction with an offset and period on the same record, the amount of the assignment is spread between offset and offset + period according to the indicated curve.

The Effect of Processing Options

The potential effects of three of the resource scheduling processing options require further explanation:

- Hard zeros
- Smoothing
- Scheduling interval

Each processing option is discussed in the following sections.

Hard Zeros

The user can specify the availability of any particular resource by using one or more settings on the **Availability** tab of the **Resource Details** dialog box. Each availability entry includes the following information:

- A start date (From Date)
- An end date (To Date)
- An availability level (Availability)
- A calendar name (Calendar)

Any number of these settings can be used to build a profile of availability. They are treated cumulatively. For example, the following two entries:

Availability	From Date	To Date	Calendar
5	01Jan04	31Dec04	<default>
2	01Jul04	31Jul04	1

would signify that 5 units were available throughout 2004 using the default calendar and that these would be augmented during July by 2 more units working to calendar 1. Depending upon the definition of the calendars, this might mean that during July there would be a total of 7 units available on weekdays and 5 on weekends.

Clearly, there will be periods of time during which there is no availability defined, and it is important to understand the conventions used by Open Plan to interpret the availability during such periods.

As we will see, there are various ways in which these periods may arise, but for each resource there must always be two such periods:

- A period of time prior to the earliest start date specified on any of the resource availability records for that resource. Under no circumstances will Open Plan schedule an activity to use resources during this period. This may mean that an activity cannot be scheduled prior to its late date and will result in a fatal processing error. The solution is either to modify the availability data so as to start earlier or else modify the project start date and redo time analysis on the project.
- A period of time after the latest end date specified on any of the resource availability records for that resource. Open Plan treats the resource during this period as if its availability is unlimited. This ensures that the resource-limited scheduling will always be able to schedule a project in a finite amount of time. If Open Plan is forced to schedule activities beyond the date range over which availabilities are defined, a warning message is issued.

If there are no availability records at all for a particular resource or if the resource is not selected for scheduling during that session, the availability of that resource is regarded as unconstrained, and a warning message is issued.

It is also possible for there to be additional periods within the overall date range covered by the availability definitions during which the availability of a particular resource is undefined or zero. In fact, there are several ways in which this can occur:

- There can be gaps in dates covered by the availability definitions.
- There can be gaps due to the use of resource calendars (for example, each weekend may constitute such a gap).
- There can be availability records that explicitly define a date range and an availability level of zero. (The reason for doing this will become apparent below.)

In resource-limited scheduling and in the absence of thresholds on the resource in question, each of these situations is treated the same way. The availability is treated as zero, and activities will be delayed if necessary so as to avoid using resources during these periods.

In time-limited scheduling and in resource-limited scheduling with a threshold, it is generally permissible to exceed the availability if necessary in order to comply with the date constraints. This may or may not be appropriate when the availability is zero. For example, you may be able to vary the number of engineers assigned to a job but you may not be able to alter their Monday-to-Friday workweek. In this case, one would want to allow the availability to be exceeded during the workweek but would not want the zero availability at the weekend to be violated.

The question therefore arises as to how and when to permit a zero availability to be exceeded. Open Plan offers considerable flexibility in this, as follows:

- By default, periods of zero availability within the overall date range are treated the same as any other resource availability. In other words, they may be exceeded in time-limited scheduling; in resource-limited scheduling they may be exceeded up to the limit imposed by the threshold.
- The hard zeros processing option inhibits the use of a resource when its availability is zero, either due to a gap in the dates covered by availability definitions or due to the use of a resource calendar.
- The hard zeros option can be overridden in specific periods by explicitly defining a period with zero availability.

Care should be taken when the hard zeros option is used, particularly in conjunction with resource calendars, as its use may make it impossible to schedule an activity within an acceptable time frame. For example, if a resource is available only on weekdays and an activity requires that resource for 6 consecutive days (which means that the activity is not on a 5-day-per-week calendar), it will be impossible to schedule that activity until after the end of the availability data. In resource-limited scheduling, the activity will simply slip, giving rise to a warning message, while in time-limited scheduling this probably will not be possible and the result will be a fatal processing error.

> **Attention:** The interpretation of availabilities, and hence of zero availabilities, is somewhat different for consumable resources. For more information, see Consumable Resources.

Smoothing

Resource smoothing is a processing option that modifies the effect of time-limited scheduling and of resource-limited scheduling when thresholds are used. To understand the effects of this option, we first need to consider the objective of time-limited scheduling and how it attempts to achieve this objective.

Everything in this section applies equally to resource-limited scheduling with thresholds, but for clarity, we will refer only to time-limited scheduling.

The stated objective of time-limited scheduling is to minimize the sum of the maximum amounts by which each resource availability is exceeded, without delaying the completion of the project. For example, by this standard it is preferable to exceed a resource by 1 unit for 2 periods rather than by 2 units for 1 period. Similarly, it is preferable to exceed one resource by 2 units than to exceed three resources each by 1 unit.

If, during the course of resource scheduling, a particular resource availability has already been exceeded by 2 units at a particular date, then by the standards of this objective there is no cost associated with exceeding it by a similar or lesser amount at some other time. Indeed, there may be a benefit in so doing, either directly or in terms of the scheduling of subsequent activities.

The standard time-limited algorithm takes note of the extent to which availabilities have already been exceeded and makes use of these excess amounts just as if they were freely available. As already stated, to do so incurs no cost. Moreover, since this approach generally results in earlier scheduling of the activity being considered, the flexibility of the system to schedule the remaining work without further exceeding availabilities is maximized.

This algorithm sometimes gives results that do not seem “right.” In particular, there is a tendency to exceed resource availabilities during periods when it appears that it was not necessary to do so.

The smoothing option avoids this problem by not exceeding any resource availability for any activity unless it is strictly necessary in order to schedule that activity. Note that this reduces the flexibility for subsequent activities and so the algorithm may sometimes perform worse by the standards of the strict objective function defined above.

The advantages and disadvantages of the smoothing option can be further illustrated by the use of two simple examples.

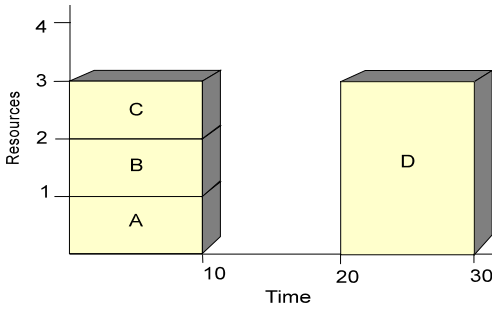
Consider four independent activities, all of which require the same resource:

- Activity A has a duration of 10 days, 20 days of total float, and requires 1 unit per day.
- Activity B has a duration of 10 days, 20 days of total float, and requires 1 unit per day.
- Activity C has a duration of 10 days, 20 days of total float, and requires 1 unit per day.
- Activity D has a duration of 30 days, zero total float, and requires 3 units per day for the last 10 days only.

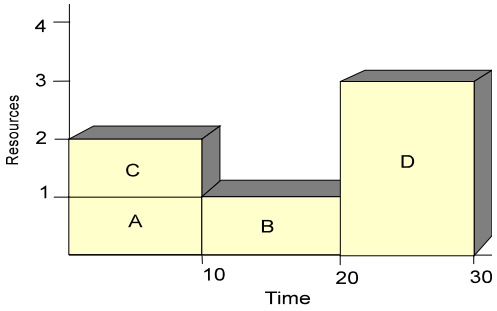
Assume further that there is only 1 unit of the resource available.

Because there is no logic connecting these activities, any one of them could be considered first. We will assume the use of the default scheduling priorities. In other words, priority is given to the smallest total float. Hence, as activity D has no float, it will be considered first. In time-limited scheduling, Open Plan has no option but to schedule it on its late dates (which correspond to its early dates of course), thereby exceeding the available resource by 2 units.

The standard algorithm (without the smoothing option) subsequently behaves exactly as if these extra 2 units were actually available and will therefore schedule activities A, B, and C all on their early dates:



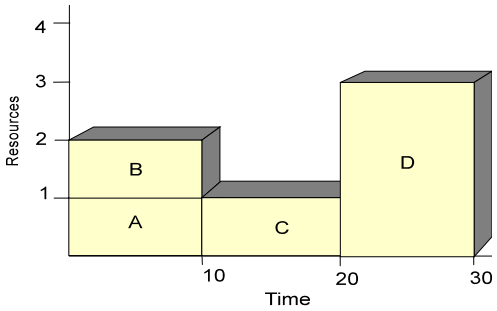
If the smoothing option is specified, however, the treatment of activities A, B, and C is different. The first to be considered (which could be any of them but assume it is activity A) is scheduled on its early date, within the original availability of 1 unit. The second activity, say B, is delayed by 10 days in order not to exceed the availability; this is acceptable, as it has 20 days float. Finally, activity C cannot be scheduled without exceeding the resource again, and this is done:



Most planners would probably prefer the “smoothed” result, even though by the standards of the stated objective, they are both equal — both solutions result in the resource being exceeded by 2 units.

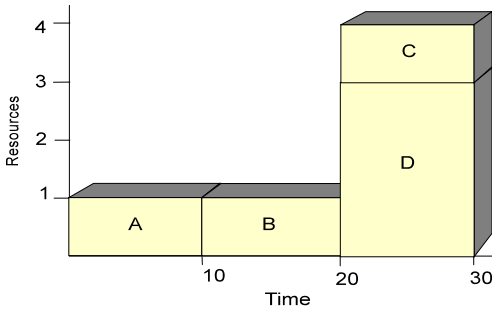
The problem with the smoothing option can be seen if the example becomes slightly more complicated by assuming that there are finish-to-start relationships between activities A and C and between activities B and C — activity C cannot start until both activity A and activity B have been completed. Activity D is scheduled as before. Because of the added logic, it is no longer possible to choose freely between the other three activities; Open Plan must schedule either activity A or activity B first. Both activity A and activity B now have a float of 10 days, so the choice is again arbitrary — assume that Open Plan selects activity A and schedules it on its early date. Note that this does not free up activity C, so the next activity to be scheduled must be activity B. The strict time-limited method (without smoothing) schedules this as before on its early date.

Finally, it schedules activity C after activity A and activity B, on its early date:



When the smoothing option is in effect, the scheduling of activity D and activity A goes as before. When activity B is considered, it will be delayed by 10 days in order not to exceed the resource “unnecessarily.” Note that this is permissible, as the activity has 10 days float.

Finally, when Open Plan comes to consider activity C, it will have no choice but to schedule the activity on its late date, a total delay of 20 days compared to its early date, since the scheduling of activity B has used the entire float. This requires Open Plan to exceed the resource availability by 1 more unit than would have been necessary had the smoothing option not been in effect:



The effect of the smoothing option is clearly detrimental in this case. Strict time-limited scheduling produces a lower maximum requirement, which is the prime purpose of the exercise. It is interesting to note, moreover, that this example does not depend upon the priority rule, or even upon Open Plan’s choice of the strictly serial scheduling algorithm, for its effect. No algorithm that schedules activities one at a time in accordance with the network logic and that does not reconsider earlier decisions can do better with the smoothing option, though it is quite easy to make the strict time-limited option do worse (by choosing an inappropriate priority rule that would consider activity D last, for example).

Scheduling Interval

By specifying the scheduling interval for a resource scheduling session, it is possible to define the granularity of the scheduling process over time. In cases of very large projects using large numbers of resources spread over long time periods, there may be a significant performance benefit in not scheduling to the smallest detail. On the other hand, a larger scheduling interval may introduce approximations that may or may not be acceptable.

Note that the performance benefit comes in part simply from not having to do so many comparisons in order to test whether an activity can be scheduled on a particular date. Also, because such a test is typically more likely to succeed, fewer start dates need to be tried.

For example, assume that you have defined the scheduling interval as weekly. If the resource availability definitions indicate that a particular resource is available at a level of 2 units per day on a 5-day workweek, all that is stored is that there are 10 units available during the week. During subsequent scheduling, an activity may be scheduled to use that resource during the week provided only that the total usage does not exceed 10 units, but without regard to the specific days on which the resource is required. For example, it may be that to execute the schedule actually requires 3 units on Monday and only 1 unit on Tuesday and so on. In this case, resource scheduling will produce rough estimates that may not be completely accurate in every detail.

Of course, to some extent, increasing the size of the interval used in scheduling tends to give a more optimistic answer than can actually be achieved. In the above example, for instance, there is nothing to stop an activity from using all 10 units on Monday, or even on Sunday if the activity calendar permits, even though in reality there are only 2 units available per day and they are not available on Sundays. To guard against the worst instances of this, use calendars on your activities rather than relying on resource calendars and avoid specifying requirement levels on individual activities that even in isolation exceed the availability.

If you want Open Plan to avoid approximations in its calculations, but you do not want to specify an unnecessarily small scheduling interval, it is important to determine the largest scheduling unit that does not cause approximations. In the simplest case, this is the lowest common denominator of all durations. For example, if all durations are known to be multiples of two hours, then a 2-hour interval is appropriate. It is also important that the project calendars conform to these intervals as well. For example, if you choose a 2-hour scheduling interval, Open Plan performs resource scheduling based on elapsed 2-hour intervals starting at midnight. For working days defined as starting at 8 A.M. and finishing at 4 P.M., this interval is suitable. For working days defined as starting at 9 A.M. and finishing at 5 P.M., however, a 1-hour scheduling interval would be required.

Priorities in Resource Scheduling

Resource scheduling in Open Plan employs the strictly serial method of determining the sequence of calculations to be performed when scheduling the activities in a project. This approach is designed to avoid the combinatorial nature of the problem by considering one activity at a time in a predefined order.

Clearly, the effectiveness of this approach depends on the way in which this order is determined, though that order is, in practice, constrained to a greater or lesser extent by network logic. The order of processing is always such that no activity is considered before its predecessors (that is, in topological sort order). There is generally more than one such order, however — the exception being when all the activities are strung end-to-end in a single chain. As a result, how ties between competing activities are resolved becomes a key concern.

Open Plan employs various rules for breaking ties. Some of these rules are always beneficial (or at least never detrimental) and are consequently employed automatically.

The strictly serial method selects activities to be scheduled by maintaining a list of those activities that are “available for scheduling” (that is, those for which all predecessors have already been selected). At the outset, this list consists of the set of start activities. From this list, the activity that has the highest priority (according to the priority rules defined below) is selected and removed from the “available” list. As a consequence of this selection, one or more of the selected activities successors may now be “available,” in which case they are added to the list.

Depending upon the priority rules, the next activity to be selected is as likely to be one of the newly available activities as any of the previous candidates. This is the meaning of the term “strictly serial.”

This method has been selected for Open Plan because it has been found to be the most effective in practice and also because it has certain features that seem to be worthwhile.

- In conjunction with the default priority rule based on total float, this method tends to favor the scheduling of the entire critical path before returning to schedule the next most critical activity.
- It is the method that gives the most effect to the priority rules, and therefore gives the most flexibility to users who want to determine their own rules.

User-Defined Priorities

As explained in the previous section, Open Plan determines the processing order primarily based on network logic. There will generally be ties encountered during this process (that is, there will be points at which more than one activity is available for scheduling). Open Plan first applies certain internal rules to establish a unique choice for the next activity to be scheduled. If there is still a tie, Open Plan then applies rules controlled by the user. These user-defined rules are based on the following features:

- You can assign the Immediate attribute to an activity.
- You can give priority to any activity that is already in progress.
- You can specify up to three activity fields to be used in sequence to decide ties.

There is no particular reason why giving priority to in-progress activities should improve the quality of the schedule as such, but it may be highly undesirable to reassign a resource that is already working on an activity. If present, this option takes precedence over the other user-defined priorities.

Specifying priority fields for the session works as follows: Priority is given to the activity with the lowest value in the first priority field. If there is still a tie, the second priority field is checked, and followed, if necessary, by the third. If a tie still remains after checking all the fields referenced, Open Plan uses the activity ID as a final tie-decider in order to ensure repeatability of the results.

By default, Open Plan uses Total Float as the first priority field while leaving the other two fields blank. This default means that Open Plan gives priority to the activity with the lowest total float. This has been found to produce the best results more reliably than any other general rule, but there may be specific cases where other rules will produce a better result.

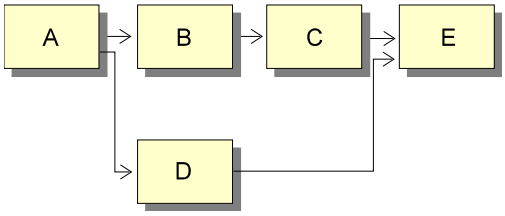
One common way of using the user-defined priorities is to allow the user to enter priority values directly on the activity record. (The **Advanced** tab of the **Activity Details** dialog box includes a field for this purpose.) It is recommended that such a user-defined priority field be used as a secondary priority field in combination with total float.

Open Plan also provides access to two special-purpose “fields” that you can select when performing resource scheduling: Hierarchical Priority and Remaining Float:

- **Hierarchical Priority** — Open Plan refers to the priority field of the parent activities when attempting to break a tie between two activities. This enables the earlier ones to increase the priority of a whole subproject.
- **Remaining Float** — Open Plan uses the activity’s remaining float (that is, the difference between the late start and the earliest feasible start — this value is calculated during the course of resource scheduling) when determining priorities.

Limitations on User-Defined Priorities

It is important to understand exactly what user-defined priorities can and cannot do. In particular, it must be noted that even in the presence of user-defined priorities, the network logic is the first consideration. Generally speaking, an activity will not be considered before its predecessors. Consider, for example, the following network:



In this example, assume that activity C is regarded as having a higher priority than the other activities. High priority is indicated by a low numeric value, so assume the priority of C is set to zero and that of the other activities to 1. Despite these priorities, Open Plan schedules activity A first, because this is the only activity without predecessors. Open Plan then has the choice of scheduling either B or D. Both have a priority of 1, resulting in a tie. This tie will be decided as follows:

- On the basis of a secondary priority field, if present.
- If no secondary priority field is specified, then the tie is broken arbitrarily.

Open Plan does not look ahead at this stage and decide to schedule B first so it can get to the high priority activity, C.

If it happens that the choice is D, B will follow, then C, and finally E. So in this case, C did not benefit at all from its high priority; in such a case, the user must specify the high priority on B as well as C.

Using the default priority rule based on total float, the above situation tends not to occur because, in the absence of target activities, B and C will have the same total float. The scheduling order tends to go as far as it can down the path with shortest float, then backtrack to the next shortest, and so on.

Another point to note is that the priority rules relate to the order in which activities are considered during the scheduling process, not the order of the schedule dates that get assigned to them. Even if Open Plan schedules activity C before activity D (that is, the scheduled date of C is determined before the scheduled date of D), it does not necessarily follow that the schedule date of C is earlier. In the above example, activity D may get scheduled at an earlier date than C, even though C gets considered first for a number of reasons, among them being:

- Activity D is available earlier. If it does not compete with B for resources, or because it uses a different resource, or because there is plenty of availability, it may start at the same time as B, which is clearly impossible for C, however high its priority.
- Activity C may get delayed because of a lack of resources while activity D may subsequently be accommodated at an earlier date because it uses a different resource, uses less of the resource, is shorter, or is available to start sooner.

Because resource scheduling in Open Plan obeys the same constraints of network logic as does time analysis, it can never produce a schedule in which the project completion date is earlier than the date obtained by time analysis. In fact, time analysis can be regarded as a special case of resource scheduling in which resources are not a constraint (that is, in which the availability of all the resources is infinite).

Resource Substitution

Several powerful features enable you to give Open Plan selective control over the specific resource that is assigned to an activity. These features include:

- Alternate resources that enable you to specify a resource that can be used as an alternate to the requested resource.
- Resource pools that enable you to specify that the resource assignment can be satisfied by any one of a number of individual resources belonging to a specified “resource pool.”
- Skills that enable you to assign a skill rather than a specific resource.

Alternate resources can be defined for each assignment. The other two features are attributes of a particular resource. Thus, alternate resources can be used in combination with the other two features, which are mutually exclusive.

Also associated with the issue of resource substitution is the ability to roll up resources to a higher level in the resource hierarchy. You can use this feature to schedule resources at the pool level, whether you have defined the assignments at that level or at a lower level.

Alternate Resources

To indicate that an alternate resource can be used to satisfy a particular assignment, enter the name of the alternate resource when you define the assignment. During scheduling, Open Plan will initially try to use the originally requested resource. If this is impossible without either delaying the activity beyond its late date or overloading the resource, Open Plan will then try using the alternate instead. Open Plan will not use both resources in combination to satisfy the requirement. In attempting to schedule using the requested resource, Open Plan will, if necessary, try to split, stretch, or reprofile the resource as may be permitted for that activity before resorting to the alternate.

For example, suppose that an activity requires resource JOHN for 10 days, with an alternate, MARY. Open Plan will use JOHN if it can do so without delaying the project or overloading that resource. It may then decide to use MARY instead, but it will not, for example, use JOHN for the first 5 days and MARY for the last 5 days.

If it is not possible to schedule the activity using the requested resource without delaying the project or overloading the resource, Open Plan attempts to use the alternate resource instead. It is possible to set an option on the **Advanced** tab of the **Resource Scheduling** dialog box in which Open Plan will always attempt to use the alternate resource where the use of the original resource will delay the activity beyond its earliest feasible date.

It is not possible to specify more than one explicit alternate resource for any one assignment. If multiple assignments request the same resource and if two or more of these assignments refer to different alternate resources, the second and subsequent alternate resources encountered are ignored. Multiple alternates can, however, be specified if they constitute the members of a pool, as described in the next section.

Note: In the time-limited resource scheduling mode, the use of alternate resources has no effect when the resource is overloaded unless you select the smoothing option.

Resource Pools

It is possible to define a hierarchical set of resources in which certain resource codes are defined as the children of other resource codes. The parent/child relationship is implied by the resource code so that, for example, the resource PROGRAMMERS might have children PROGRAMMERS.JOHN, PROGRAMMERS.MARY, and so on.

A resource pool normally groups together similar resources, which can be regarded as interchangeable to some extent. Resource pools have no availabilities directly assigned to them; availabilities are specified only at the lowest level in the hierarchy, which represents individual resources.

It is possible to assign a resource pool to an activity, either as the requested resource or as the alternate. When a pool is encountered in an assignment, Open Plan attempts to satisfy the requirement using one of the individual resources in the pool. It tries them in an arbitrary order (actually, the order in which they appear in the resource window) in just the same way as alternate resources, using the first one that enables it to schedule the activity without overloading the resource or delaying the project completion. If this approach fails, Open Plan attempts to determine the resource that causes the least delay or least amount of overload and schedules the activity using that resource.

Open Plan can also consider members of a pool as separate resources during resource scheduling. In this case, Open Plan treats assignments separately rather than combining them. In order for Open Plan to separate the resources, the option on the **Advanced** tab of the **Activity Details** dialog box must be checked. If this option is not selected, Open Plan will combine the assignments and attempt to schedule a single resource.

Note: You can exercise some control over the order in which Open Plan considers members of a pool by rearranging the members of the pool when displaying the resource file. (Resources appearing on the left or above are considered first.)

It is possible to use pools in conjunction with alternate resources. As a result, it is possible to specify a particular resource as the requested resource and a pool as the alternate. For example, if you would like the programmer MARY to do the work if possible, but otherwise would accept anyone else from the PROGRAMMERS pool, you would specify PROGRAMMERS.MARY as the requested resource and PROGRAMMERS as the alternate.

Skills

A skill is an attribute of an individual resource, each of which may have any number of skills. In general, the skill sets of different individual resources will be different but overlapping. Open Plan allows you to

designate a skill as a requested resource, and this request may be satisfied by using any individual resource that possesses that skill. As with assignments involving pools, Open Plan considers each such resource with the requested skill until it finds one that can satisfy the requirement without overloading the resource or delaying the project. If this is impossible, Open Plan reverts to the resource that causes the least delay or least amount of overload.

The difference between skills and pools derives from the overlapping nature of the skill sets. A resource that possesses skills can also belong to a pool. For example, the resource PROGRAMMERS.MARY could be used to satisfy requests for PROGRAMMERS.MARY, or for PROGRAMMERS or for any of the skills MARY possesses.

As with pools, it is possible to combine skills with alternate resources, and in particular to specify a particular resource in combination with the skill required. For example, if you need a SQL programmer (someone with the skill SQL) but would choose Mary for preference, you can specify PROGRAMMERS.MARY as the requested resource, and SQL as the alternate.

Resource Roll-up

Resource roll-up is an attribute that can be applied to individual resources or pools. If a resource is rolled up to a higher level, any request for it is treated as if the request is for its parent. The availability of that pool resource is the sum of all the rolled-up resources belonging to it.

The purpose of resource roll-up is to allow a more generic resource schedule to be produced, without detailed consideration of which individual resource will do each job. For example, if the pool PROGRAMMERS contains 12 individuals, the pool will be scheduled just as if PROGRAMMERS was an individual resource with an availability level of 12.

Although it is permitted for a pool to contain resources of different types (consumable and non-consumable resources, for example), this approach would not normally make much sense. In the case of a roll-up, mixing different resource types is not allowed. If an attempt is made to roll up resources of different types to the same pool, a fatal error occurs during resource scheduling.

As currently implemented, roll-up scheduling does not work in conjunction with alternates or skills. The existence of a rolled-up resource disables these features.

Warning: You must check all members of a resource pool for resource roll-up. If not, you will get a warning message, and your results will be incorrect.

The Effect of Reservations on Resource Scheduling

Resource scheduling will not include resource availability reserved for another project. When scheduling using time limited mode, or resource limited mode with a threshold, a resource, which is totally reserved for another project during a particular period, might still be used if necessary, but not if the **Hard Zero** option is set.

When an activity belongs to an external subproject, the availability considered for scheduling that particular activity includes that reserved for the subproject, any higher level external subprojects to which it belongs, and the main project.

Resource Usage by Other Projects

Project Scheduling Priority

Summary resource usage allows different projects to be scheduled successively using only a single resource file. In order to realize the functionality of summary resource usage, it is necessary to define a priority for each project.

Storing Resource Usage Summary

In order to take advantage of the resource usage summary, the resources used by a project must be saved in summary form with the resource file. This makes the data available to other projects using the same resource file. Each time the project is saved, the resource usage summary is also saved, provided that the appropriate selections have been made on the **Scheduling** tab of the **Project Properties** dialog box.

Storing the resource usage summary when you **Save** is only possible when both the project and the resource file are opened in exclusive mode.

Using the Resource Usage Summary

When resource scheduling a project, on the **Resource Scheduling** dialog box there is an option to **Consider Usage On Other Projects** that have a higher-level priority. (A higher priority is indicated by a lower numeric value.)

A project has no access to the other projects directly. The priority Open Plan uses for a subsequent project is the priority at the time the summary usage data was last stored.

A resource is not double-counted if it is used by the project to which it was reserved and is not available to the current project. For example, suppose that 75% of a resource is reserved for a particular project, then this leaves 25% not reserved available for the current project.

- If the project uses only 60% of the resource instead of 75%, there is still only 25% available for the current project. (Open Plan does not recognize the difference between what was used and what was reserved if the usage is lower than quantity reserved.)
- If, on the other hand, the project uses 80% of the resource, the difference between what is used and what is reserved is subtracted from the 25% not reserved. This leaves only 20% available for the current project.

When scheduling a number of projects together (using external subprojects), the scheduling priority is that for the master project. (Of course, this does not prevent you from using the individual project priorities as one of the priority fields used to break ties between individual activities.)

> **Note:** It is essential that the summary resource usage data for higher priority projects be re-saved after any changes are made to the reservations.

Maintaining Summary Resource Usage Data

Summary resource usage data needs to be deleted when it is no longer appropriate. There are two ways to delete these records:

- Perhaps the easiest way to delete the data is to use the **Delete Summary Usage For This Project** button on the Scheduling tab of the Project Properties dialog box. This will delete the records not only for the current project but also for any open external subprojects.

- The second way to delete the summary usage file is by selecting **Summary Usage** from the Edit tab on the Resource View. This will allow summary resource usage to be deleted for one or more projects selected from the list box.

The Interpretation of Resource Scheduling Results

The main result of a resource schedule is the calculation of schedule dates for each activity. Sometimes, it may not be easy to understand how a particular set of dates has been determined, and Open Plan provides a set of diagnostic tools to assist in this. These tools include:

- The earliest feasible date
- The delaying resource
- The resource scheduling session log

The Earliest Feasible Date

The earliest feasible date of an activity represents the earliest date on which the scheduling system was able to consider scheduling the activity. It might be the same as the early start date (it should never be earlier), or it might be later due to delays in previous activities.

If an activity has been delayed due to one or more previous activities having been delayed, the earliest feasible date will be later than the early start date. If the activity in question is delayed due to the unavailability of any of the resources it needs, the scheduled start date will be later than the earliest feasible start date. These two simple tests, therefore, establish whether an activity was delayed due to constraints on its own resources or due to previous delays:

- If the earliest feasible date is later than the early start date, there was a delay due to previous activities.
- If the scheduled start date is later than the earliest feasible date, there was a delay due to the resource assignments of that activity.

Of course, it is possible for both kinds of delay to apply to a single activity.

In addition to a delay in the start date, there can be a further delay in the finish date due to the use of splitting, stretching, or reprofiling. This is indicated by the scheduled duration being larger than the computed remaining duration.

The Delaying Resource

It is also quite useful to know which resource caused an activity to be delayed and/or split. Open Plan attempts to indicate the delaying resource for an activity; although, it is only a partial answer because there may be several resources involved and Open Plan indicates only one of them. Any activity that has been delayed or split due to its own resource assignments should have a delaying resource indicated.

When there are alternate resources involved (including both pools and skills), the **Delaying Resource** field reports the first alternate resource tried, even though Open Plan may use a different alternate.

The Resource Scheduling Session Log

Information can also be gleaned from the resource scheduling session log, which stores the messages issued by Open Plan during scheduling. This log contains a record of the decisions that Open Plan made with respect to the scheduling of each activity. The session log is interesting because it appears in the order in which the decisions were made and because it details each decision made with respect to an activity.

There are a number of diagnostic messages that may appear in the session log:

- Delaying activity <activity> due to resource <resource>
- Splitting activity <activity> due to resource <resource>
- Stretching activity <activity> due to resource <resource>
- Reprofiling activity <activity> due to resource <resource>
- Activity <activity> exceeds resource <resource> by <amount>
- Resource <resource> unlimited beyond availability
- Alternate <resource> unsuccessful — returning to <resource>
- Alternate <resource> unsuccessful — alternating to <resource>

The first four of these messages simply indicate that the specified activity has been delayed or somehow modified due to lack of availability of the specified resource.

The fifth message indicates that the specified activity could not be scheduled without exceeding the stated resource. This message appears only with time-limited scheduling or with resource-limited scheduling with thresholds. If the smoothing option is in effect, the message appears for every activity that exceeds a resource availability. If this option is not in effect, the message appears only when the amount that the resource is exceeded is greater than any previously incurred for that resource.

The sixth message is more of a warning that the system has had to schedule beyond the time horizon specified in the availability file, where the resource is considered to be unlimited. This may constitute an error on the part of the user, or it may have been deliberate. This message appears only once for each resource affected and is not related to a particular activity.

The final two messages in the list refer to situations in which Open Plan is unable to schedule an alternate resource. In the first case, Open Plan is returning to the originally requested resource. In the second case, Open Plan is continuing to the next alternate available from the requested pool or skill.

Risk Analysis Calculations

Management decisions are concerned with the future. While we hope to influence the outcome of events by the decisions we make, everything about the future is, to a greater or lesser extent, uncertain. The techniques of risk analysis were developed to provide a consistent methodology for quantifying this uncertainty with the ultimate goal of improving the quality of decisions.

Nowhere is the need for this type of approach more evident than in project management. By definition, project management involves the planning and coordination of large numbers of activities for which there are often no precise precedents. In many cases, the data defining the durations and costs of the individual activities making up the project cannot be known exactly before the project begins. Clearly a project plan is, by its very nature, a forecast and thus contains elements of uncertainty.

Using the tools of modern risk analysis, Open Plan allows you to quantify the uncertainties associated with project schedules. By precisely defining the probable range of the durations of each activity, Open Plan provides you with the means to model the impact of these uncertainties on such critical areas as the project completion date.

The risk analysis feature in Open Plan provides one of the most sophisticated project management systems available today. This section is designed to introduce you to some of the concepts behind this powerful feature. It begins with a general discussion of the concept of probability, followed by a discussion of the following topics:

- Interpreting risk analysis data

- The operating characteristics of risk analysis in Open Plan
- The beta distribution, as used in Open Plan

Attention: For a discussion of risk analysis operations in Open Plan, see Project Analysis » Risk Analysis in the Open Plan online help.

The Concept of Probability

On the surface, the goal of risk analysis seems impossible. After all, how can something as elusive as uncertainty be quantified and measured? To tackle this problem, mathematicians developed the concept of *probability*. In this context, the term probability abandons its everyday meaning of a simple likelihood or chance and takes on a mathematical precision.

Perhaps the simplest introduction to the concept of mathematical probability is to consider the tossing of a coin. Most people are familiar with the idea that the probability (or "odds") of the outcome of a single toss of a coin producing a head is 50%. This estimate is based on the fact that there are just two possible outcomes to the process of tossing a coin (it is traditional to ignore the possibility of the coin landing on its edge) and that these two outcomes are equally likely.

The concept of mathematical probability asserts that if the coin tossing is repeated frequently, there should be a roughly equal number of outcomes for either side of the coin. The more often the coin is tossed, the better the chances that the proportion of heads to total coin tosses will approach 50%. In short, we can define the probability of a particular outcome occurring as equal to the number of ways of getting that outcome divided by the total number of possible outcomes of a given random process.

This notion that a definite and quantifiable relationship exists between the number of possible outcomes and the likelihood of any single outcome occurring lies at the heart of the concept of probability.

Probability Distributions

The concept of probability provides the means to analyze the possible outcomes of a given process. Specifically, probability is concerned with defining the *frequency* of each possible outcome in relation to the range and distribution of all possible outcomes. The range and distribution of all possible outcomes for a process is referred to as the *probability distribution* of that process. This distribution can be represented in a number of ways, including tables and graphs.

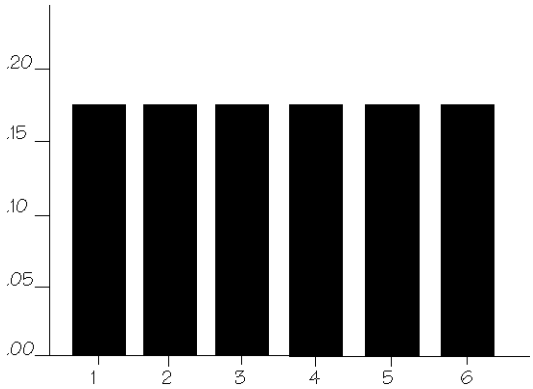
To take a simple example, consider the roll of a single die as a process. Each possible outcome – 1, 2, 3, 4, 5, or 6 – is equally likely. As a result, we can say that each outcome has one chance in six of occurring. Another way of saying this is that each result has a probability of .167, and that when added together, all the possible outcomes have a probability of 1.0.

If we construct a table to represent all the possible outcomes and their probabilities, we come up with the following:

Outcome	Probability
1	0.167
2	0.167
3	0.167
4	0.167

Outcome	Probability
5	0.167
6	0.167

We can also chart this distribution of probability using *x* and *y* coordinates where *x* represents the range of outcomes and *y* represents the probability of each outcome:



Because the probabilities for all the possible outcomes are the same in this case, this type of distribution is called a *uniform distribution*.

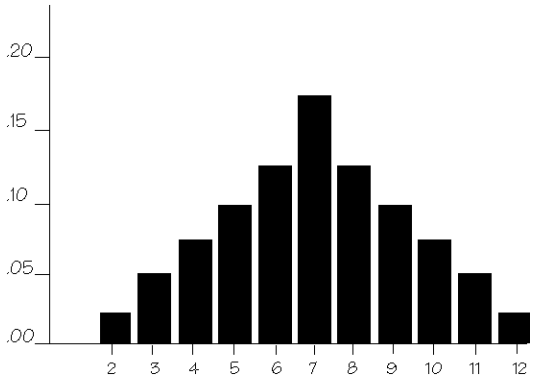
Of course, not all probability distributions are uniform. Consider, for example, the combined score from rolling a pair of dice. The range of possible outcomes is from two to twelve. Yet as any gambler knows, these outcomes are not equally likely. Rolling a twelve is much less likely than rolling a seven because there are many ways of achieving a seven but only one way to roll a twelve.

If we represent this distribution as a table we get the following:

Outcome	Probability
2	0.028
3	0.056
4	0.085
5	0.111
6	0.139
7	0.167
8	0.139
9	0.111
10	0.085
11	0.056

Outcome	Probability
12	0.028

Representing these probabilities as a graph makes it clear why this type of distribution is referred to as a *triangular distribution*:

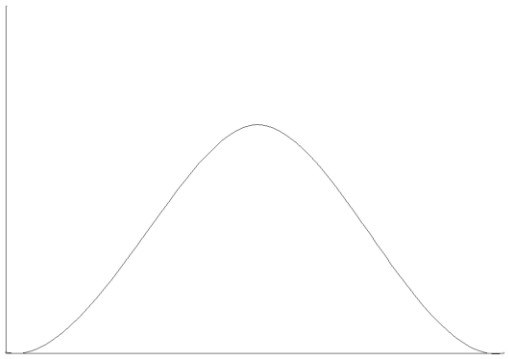


Since the height of the triangle above at any point along the x-axis represents the probability of the outcome represented by that point, the illustration makes it clear that the probability of rolling a seven is much higher than the probability of any other outcome.

Consider also the symmetry of the shape of the triangle. No matter which direction one moves away from the triangle's apex, the probability of occurrence declines at the same rate.

All triangular distributions are not necessarily symmetrical. Later in this section, we discuss the example of a triangular distribution where the most probable outcome is heavily weighted or *skewed* towards one end of the distribution.

Obviously, rolling a pair of dice adds complexity to the possible outcomes of the process when compared to rolling a single die. But suppose we add more and more dice, creating more and more possible outcomes. Eventually, if we reach a level of sufficient complexity, the probability distribution looks something like this:



This distribution is called a *normal distribution*, and it turns out that the sum of the outcomes of multiple tries of any random process tends towards this curve. In fact, it appears as though a great many processes in nature ranging from the IQs of randomly selected people to the velocity of molecules in a gas can all be effectively analyzed using normal distributions.

The normal distribution is an example of a *continuous distribution*. Unlike our earlier examples in which each possible outcome was discrete, the curved shape of the normal distribution now depicts a range of continuous rather than discrete outcomes. As a result, it is the area under the curve between any two *x*

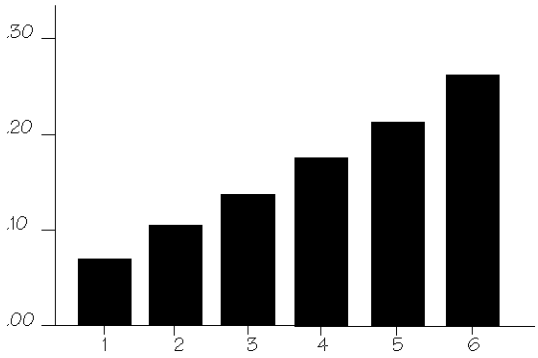
values that represents the probability of the outcome falling between these two values. (Incidentally, both uniform and triangular distributions can also represent continuous outcome.)

As useful as the normal distribution is, it still cannot represent situations where, for one reason or another, a particular outcome is more probable than others and yet does not fall in the exact center of the range of all possible outcomes. In this situation, some type of *asymmetrical probability distribution* is called for.

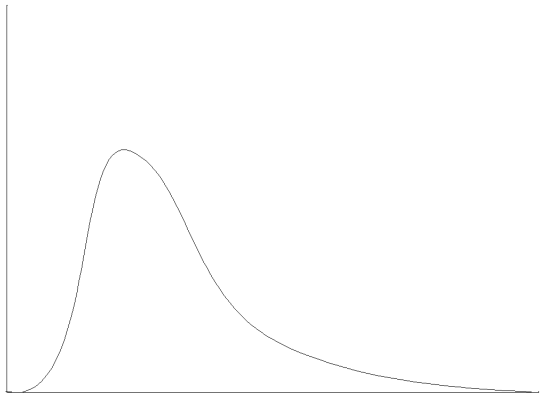
Let us re-examine the process of rolling two dice but this time analyze the probability distribution of the greater of the two numbers that appears. Clearly, the greater of two values is more likely to be toward the top end of the range of each — in fact, six turns out to be the most likely value. Why? Because out of 36 possible outcomes, only one will result in the maximum number being one (that is, both dice would have to be one), while there are a total of eleven ways to get six as the maximum:

Outcome	Probability
1	0.028
2	0.083
3	0.139
4	0.194
5	0.250
6	0.305

This is an example of an asymmetrical or *skewed distribution* and can be represented as a triangular shape:



As a process increases in complexity, we eventually come up with a continuous curved shape for the distribution, somewhat similar to a normal distribution but skewed in one direction:



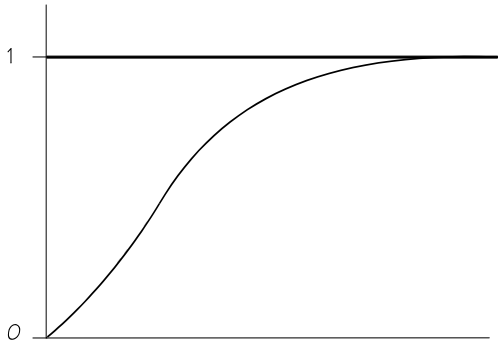
Depending on how the distribution is defined, there are a number of ways of representing this type of probability. However, for the purposes of Risk Analysis, a family of distributions known as *beta distributions* is used.

The beta distribution, like the triangular distribution, is primarily used to describe asymmetrical probability. However, a beta distribution differs from a triangular distribution since its shape effectively precludes outcomes at the extremes of the distribution from having any significant probability. By contrast, a triangular distribution allows for a significant probability for outcomes at the extreme limits of the range, no matter how much the distribution is skewed.

Note: It is generally recommended that asymmetrical probability be represented by triangular rather than beta distributions in Open Plan.

Cumulative Distributions

While representations of distributions that focus on the frequencies of possible outcomes are very common, there are other approaches as well. Another way to represent a probability distribution is to use a *cumulative distribution* of probabilities. The result is an s-shaped curve representing for each x value the probability that the outcome is less than or equal to that value. The cumulative curve always starts at 0 and ends at 1:



Summarizing Probability Distributions

Representing a probability distribution as a table or graphically on an x/y axis tells us everything we need to know about a process whose outcomes can be described theoretically. But, this is not always possible in the real world. Consider the problem of representing data arrived at empirically, for example, by sampling the outcomes of an experiment.

For this and other reasons, it is often convenient to use some *statistical representation* or summary to describe a sampled distribution. The most common of these summary statistics are:

- The *mean value* of the distribution
- The *standard deviation* of the distribution

The mean value (traditionally represented by the symbol μ) is a measure of the central tendency of the distribution and is more or less synonymous with the more colloquial terms average and expected value. The mean is calculated as follows:

$$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

where μ is the mean, x_1, x_2, and x_n are outcomes, and n is the total number of outcomes.

For probability distributions based on sampling, the mean is considered to be the most reliable statistical indicator of the *central tendency* since it takes into account all the outcomes sampled.

The other important statistical measure of a distribution is the standard deviation (represented by the symbol σ). The standard deviation measures the amount of variation within the distribution around the mean value and is calculated by first estimating the *variance* of the distribution according to the following formula:

$$\sigma = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n}$$

where x_1, x_2, and x_n are outcomes, $\bar{x}$ is the sampled mean value, and n is the total number of outcomes.

The square root of the variance is then taken to find the standard deviation for the distribution.

For example, assume that ten students taking a math test receive the following scores: 64, 69, 71, 73, 73, 75, 82, 82, 82, 89.

The mean score is 76.

$$\frac{(64 + 69 + 71 + 73 + 73 + 75 + 82 + 82 + 82 + 89)}{10} = 76$$

The variance is 54.1.

$$\frac{(144 + 49 + 25 + 9 + 9 + 1 + 36 + 36 + 36 + 169)}{10} = 54.1$$

The standard deviation is 7.1.

$\sqrt{54.1}$ = 7.1

As it turns out, if we know the mean and the standard deviation, we can completely describe a normal distribution represented by the equation:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-1/2\left[\frac{x - \mu}{\sigma}\right]^2}$$

where f is the frequency, x is the outcome, μ is the mean value, and σ is the standard deviation.

Note that the value of f peaks when x = μ and declines symmetrically as x moves away from μ in either direction. The rate at which f declines depends on the parameter σ. Note also, that the value of f never quite reaches zero. This means that no value of x is an absolutely impossible outcome for any true

normal distribution. In practical work, however, it is common practice to truncate the tails of a normal distribution.

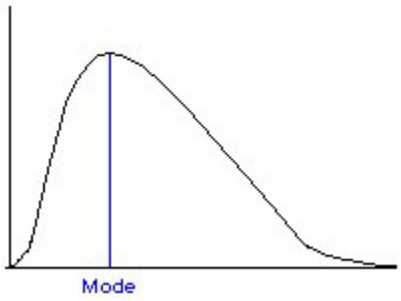
Also, note that since not all distributions are symmetrical, there may also be a need to measure the extent to which variations from the mean in one direction are more likely than the other in order to describe the distribution mathematically.

Other Measures of Central Tendency

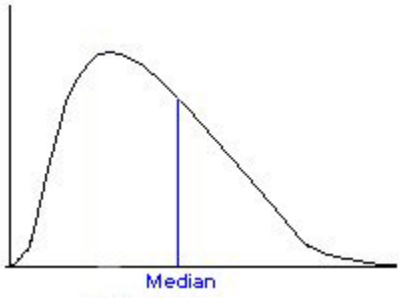
Although the mean is perhaps the most common measure of central tendency of a distribution by virtue of its stability, it is not the only one. Other measures of central tendency can also be useful, particularly in cases of asymmetrical distributions. Two of the most common of these are:

- The mode value
- The median value

The mode or “most likely” value of a distribution is represented by the highest point on the frequency distribution curve:



The median value of a distribution divides the area under the curve into two equal parts:



For example, assume once again that ten students taking a math test receive the following scores: 64, 69, 71, 73, 73, 75, 82, 82, 82, 89.

In this case, the mode score is 82 (the most common score), while the median score is 74 (five students scored below, five students scored above).

For symmetrical probability distributions, the mean, mode, and median values are the same.

More About Standard Deviation

As we have seen, the standard deviation is the square root of the average of the squares of the individual deviations from the mean value. For this reason, the standard deviation is sometimes referred to as the root mean square (RMS) deviation. It should be noted that one result of this squaring is that the standard deviation measures the degree of variation without regard for the direction in which the variation occurs.

Still, the standard deviation of a probability distribution is a very useful value to know. As a very rough guide, it turns out that there is about a two-thirds chance of any particular outcome falling within one standard deviation of the mean value and about a 99% chance of the outcome falling within three standard deviations of the mean. This variation can occur in either direction from the mean and is usually referred to as a 99% *confidence interval*.

For example, the mean temperature for Houston, Texas, during the month of August at 5 P.M. is 93 with a standard deviation of 3.5. Based on three standard deviations (that is, ±10.5), then it is almost (99%) certain, that the 5 P.M. temperature in Houston on any given day in August will fall between 82.5 and 103.5.

Confidence Intervals

As noted previously, the standard deviation measures the degree of variation without regard for the direction. This is not a problem in symmetrical distributions, but if the distribution is not symmetrical then there may be a significantly greater chance of a variation in one direction rather than the other.

Another way of establishing confidence for a probability distribution is to estimate confidence intervals directly, using statistical sampling. Establishing a confidence interval directly means determining a pair of values within which we have some confidence that the outcome will fall. The interval needs to be qualified by the degree of confidence required —a 90% confidence interval would be a pair of values between which the outcome in question has a 90% chance of falling. Thus, for example, in the random selection of any number between 1 and 100, there is a 90% confidence interval that the number chosen will be greater than 5 and less than 96.

It is normally implicit that the interval should be symmetrical, in the sense that there would be an equal probability that the outcome would fall outside the confidence interval in either direction. Note that because the distribution itself may be skewed, this means that the confidence interval is not in general symmetrically placed with respect to the mean value.

In general, confidence intervals calculated in this manner are to be preferred to those based on standard deviations since the skew in the underlying distribution is indicated.

In some cases, however, standard deviations are more reliable indicators of confidence. This is particularly true for relatively small samples since standard deviations are based on all outcomes instead of a few extreme values.

Combining Probabilities

So far we have just considered the probability in relation to a single process. Yet project management poses the case where many activities, each with separate factors affecting duration and cost, can nevertheless affect each other as well.

Fortunately, the probability theory provides us with the analytical tools
to combine probability distributions in ways appropriate to specific circumstances. For example, consider the following project consisting of a pair of activities:

[A] ⟶ [B]

Suppose that activity A and activity B are both expected to have a duration between one and six days, and that any number of days between one and six is equally likely.

The mean or “expected” duration of each activity is 3.5 days:

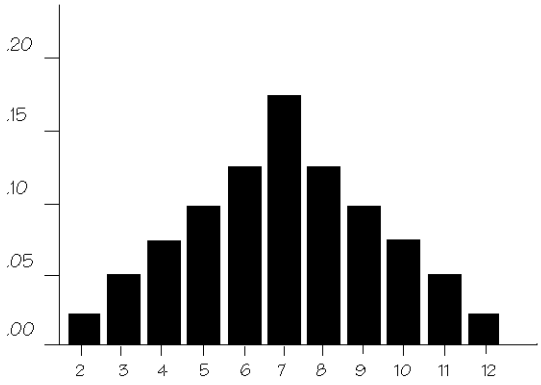
$$\left[\frac{(1+2+3+4+5+6)}{6}\right]$$

Further, we can calculate that the mean duration for the whole project is seven days by adding the mean durations of the individual activities together. Mean durations are additive in this case.

It also turns out that variances are additive in this case. The variance of the distribution of each duration is approximately 2.9, and so the variance of the total project is 5.8. By taking the square root of the total project variance, we arrive at the standard deviation, about 2.4 in this case.

It is common shorthand to state this estimate of the overall project duration as "7 ± 2.4." Note that this does *not* mean that the duration is sure to fall within these limits. (For a normal distribution, the probability of an outcome falling within 1 standard deviation of the mean is only about 68%.)

What is the actual shape of the distribution of the overall project duration in this case? If we graph the duration probability of the project, we realize we have the same triangular distribution shape we get when we roll a pair of dice and add the two numbers:



Unfortunately, we cannot always combine probabilities by adding means and variances. In fact it is only appropriate to do so when the outcomes themselves are being added, as in the case of the durations of two activities to be performed in series.

Suppose that the same two activities had to be done in parallel. The overall project completion is now dependent on both activities being completed. In other words, we are looking for the distribution of the greater of two uniformly distributed durations. This is analogous to the problem discussed earlier of finding the distribution of the greater of a pair of randomly thrown dice.

Since project networks generally comprise a multitude of activities in combinations of the above two configurations (that is, partly in series and partly in parallel), the analytical approach to calculating probability introduced here quickly becomes impractical. The only practical approach is to use a technique known as *Monte Carlo simulation*.

The Monte Carlo Simulation Technique

The Monte Carlo simulation technique gets its name from the casinos that make that European city famous. Conceptually, Monte Carlo simulation is very simple. In order to find out the distribution of possible outcomes, we merely perform the process a large number of times. With the results, we build up a histogram to represent the probability distribution from the individual outcomes of each trial.

Of course, we cannot really perform a project many times. Indeed, we cannot even perform it once without defeating a principal purpose of risk analysis (namely, to decide whether or not the project is feasible in the first place).

We can, however, *pretend* to do so, and this is exactly what Monte Carlo simulation does. Instead of doing time analysis once based on fixed duration lengths for each activity, we simulate the project several hundred or several thousand times. Each time, we use a different set of data for durations and costs sampled from the probability distributions of these uncertain values.

The sampling is done so that the probability of selecting a particular duration in the simulation is the same as our subjective estimate of the probability of that value actually occurring. With definite values for

activity duration and costs thus supplied, we can calculate such things as project completion date and total cost for each trial.

As enough simulations are performed, a definite picture of the risks inherent in the project begins to emerge.

Note: Do Monte Carlo simulations work? If you have a pair of dice, you can try simulating the simple two-activity project described in the previous section. Each simulation of the project is conducted by rolling the two dice, the total project duration for that trial being the sum of the values thrown. Keep a note of the number of times each possible value (from two to twelve) occurs. After a while, the expected triangular distribution should emerge, and the more simulations you do the better the results.

The Question of Subjectivity

Estimates of probability are always subjective to some degree. Consider your estimate of the outcome of the next toss of a coin which has just turned up heads 20 times in succession. If we assume that the coin is unbiased, it follows by definition that the next toss is just as likely to be heads as tails. But some people might start to question this assumption after 20 consecutive heads.

How soon and to what extent this consideration is allowed to modify your estimate of the probability of the next outcome depends upon such factors as how well you trusts the coin-thrower, whether you have examined the coin, etc. In this way, a subjective viewpoint would play a definite role in the expectation of probability.

In fact, *all* probabilities are to some extent subjective. That is to say, they reflect the state of knowledge of the estimator.

In project management, the subjective knowledge of the project planners plays an important role in estimating uncertainties. Unlike the simple example of coin tossing, it is not usually possible to perform experiments about the outcome of processes of interest to project managers. Nor do these events lend themselves to the calculation of a probability based on arguments of symmetry. Generally, we are concerned with quantifying the uncertainty associated with a forecast of the cost or duration of a process which (a) has yet to occur and (b) is expected to occur exactly once.

Clearly, no two individuals are likely to agree on these definitions nor is there any way to prove who was right even after the process. Whether tossing a single coin or analyzing the risks of a project combining thousands of activities, the estimate of the uncertainty is, in the end, a statement more about the state of knowledge of the estimator than about the real world.

The concept of probability and risk analysis in this environment is not easy to grasp. Indeed, some project managers may feel that it is without foundation. We will, nevertheless, maintain that it is possible to make a subjective estimate of the probability distribution associated with our forecasts about the future. Perhaps more importantly, it is *necessary* to make such an estimate since the failure to do so merely asserts that there is no uncertainty associated with these forecasts.

Interpreting Risk Analysis Data

The standard views supplied with Open Plan enable you to quickly access important information about projects you wish to subject to risk analysis. However, since the results produced by risk analysis can be more complex than those produced by time analysis or resource scheduling, you may require more detailed information about how Open Plan handles certain reporting functions. Among the basic concepts of reporting risk analysis results covered in this section are:

- The significance of early and late start dates

- The concept of sampling error
- The calculation of finish dates

Early and Late Start Dates

For the most part, risk analysis is concerned with the earliest dates possible for activities in general and for the project completion in particular. On large projects, it may also be important to track the probable early start dates for multiple project milestones.

In general, the early start dates calculated by risk analysis will be later than those in a project using standard time analysis. This is because in simulating a project with parallel paths of activities, it is always the latest of the early start dates that controls the scheduling of the remainder of the project.

This tendency of the latest early dates controlling the subsequent early dates also tends to skew the shape of the probability distribution of all early start dates to the right. This skewedness undercuts the reliability of using the calculated mean date and standard deviation to estimate confidence intervals. In many cases, for example, it may turn out that adding three standard deviations to the right of the mean early start date results in a value outside the range of the curve, thus rendering confidence levels based on standard deviations inaccurate.

Unlike early start dates, the latest start dates of activities may be of little practical significance to many users. Nevertheless, some project managers may be concerned with late start dates and use this information to avoid starting expensive activities with lots of float much earlier than absolutely necessary by assigning that activity a target start date. Assuming that you wish these target start dates to pose a minimal risk for delaying the project completion date, you would want to select a date at a very low confidence level (for example, 1% to 5%). Selecting a later start date (in other words, a date with a higher probability of actually occurring) increases the likelihood of causing further delays in the completion of the project.

Note: An inevitable result of supplying an effective target start date to an activity is the alteration of the early date's distribution curve and an increase in the probability of the activity being critical and thus affecting subsequent activities. Specifying a date with a very low probability to a small number of activities is likely to have a minimal effect on the project. However, the more activities supplied with target start dates, the greater the likelihood that the final outcome of the project will be altered from the original projections.

The Concept of Sampling Error

One key to interpreting risk analysis data is understanding the significance of the values reported for the mean and standard deviation of a probability distribution. It is important to keep in mind that these values, based on the sampling of Monte Carlo simulations, are estimates. Obviously, the more simulations Open Plan performs, the more likely the estimated mean and standard deviation would approach the theoretically correct values for that distribution.

However, it turns out that the likely error in the estimate of a mean can itself be estimated. To estimate the sampling error of a mean value arrived at by multiple simulations, it is necessary only to divide the standard deviation by the square root of the number of simulations performed to calculate the standard error of the mean. Just as one standard deviation on either side of a mean defines a 68% probability of occurrence of an outcome within that range, one standard error on either side of the mean defines a 68% probability that the theoretical value of the mean will fall in this range. Similar confidence levels of 95% based on two standard errors and 99% based on three standard errors can also be calculated in this way.

For example, assume that the probability distribution of the early start date of an activity has a standard deviation of three days. At the 99% confidence level, the sampling error of our estimated mean is as follows:

Number of Simulations	Estimated Sampling Error (99% Confidence)
10	± 3 days
100	± 0.9 days
1000	± 0.3 days

The capability to estimate a sampling error is one reason why it is important to display standard deviations alongside mean values when reporting on the results of risk analysis.

Finish Dates

Open Plan does not generate the mean and standard deviation of the finish dates for each activity directly during the risk analysis. Instead, these dates are calculated analytically, using the sampled mean and standard deviation of the start dates and the theoretical mean and standard deviation of the specified duration distribution.

Finish dates are calculated in this manner to save both space and processing time. An unintended result, however, is the possibility of slight inconsistencies between the results reported for the finish of one activity and the start of its successor. These differences are not significant relative to the sampling error implicit in the Monte Carlo approach and should disappear if a large enough number of simulations are performed.

Operating Characteristics of Risk Analysis

This section contains detailed information on the following areas of the operation of risk analysis in Open Plan:

- Sampling methods
- A comparison of symmetrical distributions used by Open Plan
- A comparison of asymmetrical distributions used by Open Plan
- Analytical calculations in risk analysis

Sampling Methods in Risk Analysis

All sampling performed by Open Plan is based on pseudo-random numbers generated in the range 0 to 32767.

Uniform and triangular distributions are sampled by the inversion method. In the case of uniform distributions, this method amounts to a simple linear transformation of the random number to place it in the range specified by the user.

Open Plan samples normal distributions by adding 12 independent pseudo-random numbers. The input minimum and maximum values specifying the distribution are taken to represent a range of six standard deviations while the mean is calculated by averaging the two values.

This method of sampling is approximate since Open Plan deliberately ignores the approximately 1% chance of data falling outside of that range in a true normal distribution. However, the accuracy of this approximation can be gauged from the following comparison of theoretical and sampled percentile points for a normal distribution ranging from 0 to 100:

Percentile Point	Theoretical Value	Sample Value Based on 10,000 Simulations
1	11	12
5	23	23
10	29	29
25	39	39
50	50	50
75	61	61
90	71	72
95	77	78
99	89	89

For example, in this case the theoretical expectation that 5% of the outcomes will not be greater than 23 is exactly mirrored by the sampled values.

Attention: For information about sampling a beta distribution, see The Beta Distribution.

In all four cases, the underlying distribution is taken to represent a continuous outcome. In reality, however, the actual durations and costs are measured in discrete units such as days, dollars, etc. To translate continuous values to discrete values, the two specified end-points are treated inclusively and given full weight. For example, the user has specified a particular range of possible outcomes as from 0 to 10. This indicates that a total of 11 possible outcomes exist, and (in the case of uniform distributions) the values 0 and 10 are just as likely to occur as any intervening values.

This translation of continuous to discrete values involves the possibility of rounding errors in cases where the number of possible outcomes is relatively small. These errors may affect the standard deviation of the distribution.

Comparing Symmetrical Distributions

All four distributions used by risk analysis in Open Plan can be directly compared only in the case when the triangular and beta distributions are specified to be symmetrical. In the following table, each distribution was assumed to have a minimum value of 0, a maximum value of 100, and a mean value of 50:

Percentile Point	Uniform (SD=29.2)	Triangular (SD=20.6)	Normal (SD=16.8)	Beta (SD=16.8)
1	1	6	11	14
5	5	15	23	22

Percentile Point	Uniform (SD=29.2)	Triangular (SD=20.6)	Normal (SD=16.8)	Beta (SD=16.8)
10	10	22	29	27
25	25	35	39	37
50	50	50	50	50
75	75	65	61	62
90	90	78	71	72
95	95	84	77	78
99	99	93	89	86

The table shows the cumulative probability distribution for each of the four distribution types. The **Percentile Point** column indicates the height of this cumulative curve. Each of the subsequent columns shows the corresponding value for a particular distribution. For example, the second line of the table indicates that 5% of the values sampled from the normal distribution will be equal to or less than 23.

As might be expected, a uniform distribution represents a considerably wider spread of values than the others, indicated both by the standard deviation (SD) and by the extreme percentile points. The triangular distribution follows in terms of spread. The normal and beta distributions both have finer tails than either the uniform or triangular distributions and, in this symmetrical case, resemble each other quite closely.

Comparing Asymmetrical Distributions

It is also possible to see how the two asymmetrical, or skewed, distributions behave when varying degrees of skew are introduced. One interesting point is that the standard deviation of the triangular distribution increases with skew. In a sense, this tendency of triangular distributions compensates for the skew by ensuring that there is still a significant probability of data falling throughout the range:

Mode Value	Mean Value	Median Value	SD	Percentile Points							
				1	5	10	25	75	90	95	99
0	33	30	24	0	2	4	13	50	68	78	90
5	35	31	23	2	4	7	15	51	69	78	90
10	37	33	23	2	6	9	17	52	70	79	90
15	38	35	22	3	8	11	20	54	72	80	91
20	40	37	22	3	9	13	22	55	72	80	91
25	42	39	21	4	10	15	25	57	72	81	91
30	43	41	21	5	11	17	26	58	73	81	91

Mode Value	Mean Value	Median Value	SD	Percentile Points							
				1	5	10	25	75	90	95	99
35	45	43	21	5	12	18	29	60	74	82	92
40	47	45	21	5	13	19	31	61	75	83	93
45	48	48	21	6	14	21	33	63	77	84	92
50	50	50	21	6	15	22	35	65	78	84	93
55	52	53	21	6	16	23	36	66	78	85	93
60	53	55	21	7	17	24	38	68	80	86	94
65	55	57	21	8	17	25	40	70	81	87	94
70	57	59	21	8	18	26	41	72	82	87	94
75	58	61	21	7	19	27	43	75	84	89	95
80	60	63	22	8	20	28	45	77	85	90	95
85	62	65	22	9	20	28	45	79	87	91	96
90	63	67	23	9	20	30	47	82	90	93	97
95	65	69	23	9	22	31	48	84	92	95	97
100	67	70	24	9	21	31	49	86	95	97	99

Note how both the mean and the median of the distribution are much closer to the central point (50) than is the mode. (In fact, calculating the mean of a triangular distribution is analogous to calculating the center of gravity of a triangle by adding the minimum, mode, and maximum values and dividing by three.)

Also, note that even with extreme skew in either direction, there remains a significant chance of obtaining data close to the opposite end of the range. For example, when the mode is equal to the maximum value of 100, there is a 1% chance of obtaining values below 10.

By contrast, the standard deviation of the beta distribution decreases with skew, exacerbating the tendency for a very long, thin tail to the distribution:

Mode Value	Mean Value	Median Value	SD	Percentile Points							
				1	5	10	25	75	90	95	99
0	12	9	11	0	0	1	3	18	28	35	47
5	16	13	12	0	2	4	7	21	31	37	51
10	20	17	12	1	3	5	10	27	36	42	54

Mode Value	Mean Value	Median Value	SD	Percentile Points							
				1	5	10	25	75	90	95	99
15	23	22	13	1	4	7	13	33	43	49	63
20	27	25	14	3	7	10	16	36	47	53	65
25	31	29	15	4	9	13	19	40	51	57	69
30	35	34	16	6	11	15	23	45	56	62	71
35	38	37	16	8	14	18	26	49	60	66	77
40	42	42	16	11	17	22	30	53	63	69	78
45	46	45	17	13	20	25	34	56	66	72	80
50	50	50	17	14	22	27	37	62	72	78	86
55	54	55	17	18	26	32	42	65	74	79	87
60	58	58	16	20	30	36	46	69	77	82	88
65	62	63	16	22	32	39	50	73	81	85	91
70	65	66	16	26	37	43	54	76	84	88	93
75	69	71	15	31	42	48	59	79	86	90	95
80	73	75	14	32	45	52	63	83	89	92	96
85	77	78	13	38	50	56	67	87	93	95	98
90	80	83	12	44	56	62	72	89	94	96	98
95	84	87	12	49	61	68	77	92	95	97	99
100	88	91	11	51	64	71	82	96	98	99	99

Both the mean and the median in a beta distribution are closer to the specified mode. As the amount of skew increases in either direction, the probability of values toward the opposite extreme becomes insignificant. Thus, when the mode in a beta distribution is equal to the maximum value of 100, there is only a 1% chance of obtaining data in the bottom half of the range. This particular beta distribution does not yield significantly different results from a triangular distribution with a minimum value of 50.

Analytical Calculations in Risk Analysis

Since some results produced by risk analysis can be calculated analytically, Open Plan takes advantage of this fact to speed up processing and to minimize data storage requirements (thereby decreasing the need for disk access).

In particular, the mean and standard deviation of early and late finish dates for an activity are, in most circumstances, calculated from the mean and standard deviation of the early start dates, the late start dates, and the input duration distribution.

One result of this approach is that there may be minor discrepancies between the mean and standard deviation reported for a finish date of one activity and those of the corresponding start date of a successor activity. This is due to the sampled mean and standard deviation of the first activity's duration in the particular simulation run not being equal to their theoretical values. This discrepancy should not be significant unless the number of simulations is very small and can be reduced or even eliminated by increasing the number of simulations.

The Beta Distribution

Strictly speaking, the beta distribution is a two-parameter distribution generating a random variable in the range {0,1}. In fact, this type of distribution originally arose out of the study of order statistics: if the two parameters are *n* and *m*, the distribution represents the probability distribution of the (*n*+1)th lowest value out of (*m*+1) independent random numbers. Note that this means that the values of *n* and *m* must be integers.

In order to make the beta distribution capable of representing data in a range other than {0,1}, as is certainly required for the purposes of risk analysis, a simple linear transformation is required. This, however, implies the specification of an additional two parameters to define the minimum and maximum of the distribution.

Thus, in order to have a sufficiently generalized beta distribution capable of representing an arbitrary range of data, a total of four parameters is required: two parameters to specify the range of data and two parameters (which must be integers) to define the shape of the distribution.

Typically, a user specifying a skewed distribution expects to specify only three parameters – the minimum, the maximum, and the mode. For the beta distribution, however, the position of the mode relative to the range is based on the ratio of the two integer parameters *n* and *m*. In effect, it is impossible to define a beta distribution with only the minimum, maximum, and mode without encountering the following problems:

- The position of the mode does not in itself uniquely define the shape of parameters n and m. A beta distribution where n equals 2 and m equals 3 is vastly different from one where n equals 200 and m equals 300. Although both distributions could share the same mode value, the latter distribution would be much more highly peaked, providing a greatly reduced probability of an occurrence at the extremes of the range.
- The position of the mode may not be represented by any rational fraction, so that no integer values of n and m exactly fit. Of course, one can get arbitrarily close by choosing a very large value for m. This approach, however, would result in a very peaked distribution.
- When the mode is close to either end of the range, the shape of the resulting distribution enables only an extremely small probability of realizing a value toward the opposite end of the range. (Just how small is shown in the table at the end of the previous section.)

These problems highlight the limitations of using a beta distribution as opposed to a triangular distribution for representing probabilities in project management. Nevertheless, some managers continue to favor the beta distribution since the curved shape would seem more “natural.”

Open Plan attempts to compensate for the inherent shortcomings of beta distributions by fixing the value of parameter *m* at 6, making the standard deviation of the beta approximate to that of a normal distribution with the same range. This results in restricting the *n* parameter to an integer value between 0 and 6, and it is out of this limited group of possible beta distributions that Open Plan conducts its sampling. To overcome the rounding error resulting from this small number of discrete allowable values

for *n*, linear interpolation is used between the two *n* values which bracket the required (generally fractional) value of the mode.

This approach has a number of ramifications. For example, as noted in the previous section, Open Plan uses the theoretical mean and standard deviations of the duration distributions in order to calculate certain results analytically whenever possible. However, Open Plan only approximates the theoretical values of beta distributions using the following empirically derived formulae:

$$mean\ value = \frac{minimum + 6\ (mode) + maximum}{8}$$

$$standard\ deviation = \frac{(maximum - minimum)}{6\ (1.25 + x' - x)'}$$

$$x = \frac{(mode - minimum)}{(maximum - minimum)}$$

The quality of these approximations is adequate for most purposes, as indicated in the following table, based on a series of beta distributions in the range 0 to 100, with various modes. The table compares the mean and standard deviations, as estimated by the above formulae, with sampling estimates based on 10,000 simulations:

Formula Mode	Sampled Mean	Formula Mean	Sampled SD	SD
0	12	13	11	11
5	16	16	12	11
10	20	20	12	12
15	23	24	13	14
20	27	27	14	14
25	31	31	15	15
30	35	35	16	15
35	38	39	16	16
40	42	42	16	16
45	46	46	17	16
50	50	50	17	17
55	54	54	17	16
60	58	57	16	16

Formula Mode	Sampled Mean	Formula Mean	Sampled SD	SD
65	62	61	16	16
70	65	65	16	15
75	69	69	15	14
80	73	73	14	14
85	77	76	13	14
90	80	80	12	12
95	84	84	12	11
100	88	87	11	11

Cost Calculations

Costs calculations take place at all levels within a project and are among the most feature-rich aspects of Open Plan. Using cost calculations, you can calculate current values for both budget and actual costs, compare these values to baseline budget and actual costs, and calculate earned value.

- **Budgeted costs** — Open Plan allows you to enter and store time-phased budgets by any or all of the four categories.
- **Actual costs** — When entering resource progress, you can enter actual costs both as quantities of resource units and as costs. You can also enter actual cost directly against an activity if you do not track actuals back to the resource.
- **Baseline dates** — You can store either early, late, or scheduled dates for activities in a project baseline. You can create multiple baselines for a project to allow you to base earned value calculations on different stages of project development. You can also create a performance measurement baseline (PMB) that is used for cost calculations.
- **Physical percent complete** — When entering information about the progress of an activity, you can enter a separate estimate for the physical completion of the activity. Physical percent complete can also be entered for resources assigned to an activity.

In Open Plan, all costs come from the cost of the resources assigned to an activity in a project. This means that unless a project uses resources, you cannot use Open Plan to calculate or display costs.

Cost Data

Cost calculations take place at all levels within a project. The following sections discuss the levels at which cost calculations are performed:

- Resource
- Activity
- Subproject

- Project

Resource

The manner in which resources are assigned has ramifications for the way the resource cost is calculated. In Open Plan, resources can be assigned in either of the following two ways:

- **Total** — When resources are assigned using any of the available spread curves, Open Plan interprets the value as a specified total of resource units required by an activity regardless of the activity duration. Resources that are assigned in this way are multiplied by a unit rate to yield a cost.
- **Level** — When resources are assigned without using a spread curve, Open Plan interprets the value as a specified level of resource units required by the activity for each time unit. Resource units that are assigned in this way are multiplied by a duration before multiplying by a unit rate to yield a cost.

The duration used in this calculation can be either the resource duration (called a period) or the duration of the activity to which the resource is assigned. If it is entered, Open Plan uses the resource period instead of the activity duration.

Depending on how the resource is defined in the **Resource Details** dialog box, the rate used in calculating the resource cost can be either a flat rate or one that can escalate over time.

All values are expressed as currency and are separated into the four resource categories:

- Labor
- Material
- Other Direct Costs
- Subcontract

Each category is rolled up to fields in the activity table.

For **Labor** resources, Open Plan rolls up the budget, earned value, scheduled value, actual quantities, and remaining quantities to the activity and subproject levels of the project provided that:

- Budget is stored in the BAC_QTY field of the activity.
- Earned value is stored in the BCWP_QTY field of the activity.
- Scheduled value is stored in the BCWS_QTY field of the activity.
- Actual quantities are stored in the ACWP_QTY field of the activity.
- Remaining quantities are stored in the ETC_QTY field of the activity.

In the case of **Other Direct Costs** resources, Open Plan stores the budget values in the activity budget cost field of the activity. In addition, any actual costs for these resources are stored in the activity actual cost field of the activity.

For all resource classes, the resource definition table contains a field called the **Roll Cost** flag. The value of this field is set to **True** by default. If you change this value to **False** for a specific resource, Open Plan does not use that resource during cost calculations. To set the value to **False**, clear the **Consider in Cost Calculations** option on the **General** tab of the **Resource Details** dialog box for that resource.

Attention: For more information on using the Resource Details dialog box, see **Working with Resources » Tabs and Dialog Boxes » Resource Details Dialog Box** in the Open Plan online help.

Activity

Open Plan stores the rolled-up values of the budget and actual cost of resources in the activities to which the resources are assigned.

Activity Fields	Budget Cost (BAC)	Earned Value (BCWP)	Scheduled Value (BCWS)	Actual Cost (ACWP)	Remaining Cost (EAC)
Labor Quantity	BAC_QTY	BCWP_QTY	BCWS_QTY	ACWP_QTY	ETC_QTY
Labor Cost	BAC_LAB	BCWP_LAB	BCWS_LAB	ACWP_LAB	ETC_LAB
Material Cost	BAC_MAT	BCWP_MAT	BCWS_MAT	ACWP_MAT	ETC_MAT
ODC (Other Direct Costs)	BAC_ODC	BCWP_ODC	BCWS_ODC	ACWP_ODC	ETC_ODC
Subcontract Cost	BAC_SUB	BCWP_SUB	BCWS_SUB	ACWP_SUB	ETC_SUB

Attention: For information on how each cost is calculated, see Cost Calculations.

In addition, if a baseline is applied to the project, the cost calculations process can also calculate the budgeted cost of the work scheduled for each activity. Open Plan performs this calculation by looking at what baseline budget value should have been accomplished by the current Time Now date.

Open Plan uses the Project Measurement Baseline (PMB) for this calculation if it exists. If the PMB does not exist, Open Plan uses the currently selected baseline, or, in the case of multiple baselines, the first selected baseline.

Subproject

Subprojects may have their own resources assigned to them. Typically, resources that are assigned at the subproject level are those that work as long as the subproject takes. These are known as **Level of Effort** resources. The cost calculations that are performed at the subproject level are the same as those performed at the activity level.

In addition, the **Include Child Activity Values in Subproject Costs** option on the **Cost Calculations** dialog box has ramifications for cost calculations:

- If this option is selected, Open Plan rolls up budget and actual values from child activities to their parent subprojects.
- If this option is cleared, a subproject calculates budget values from its own resources but not from child activities.

If you report on a project using grouped views, you may wish to turn this feature off in order to avoid double counting the activity cost by including it in both the activity and its parent subproject.

The subproject physical percent complete (PPC) is calculated by summing the activity BCWP values of the activity and dividing by the summed baseline BAC cost fields for all categories:

$$\text{Subproject PPC} = \frac{\text{Sum(BCWP\_CST)}}{\text{Sum(BAC\_LAB + BAC\_MAT + BAC\_ODC + BAC\_SUB)}}$$

Project

Projects inherit their rolled up physical percent complete, budget costs, and actual costs from these values in their activities.

Resource Cost Table (CST) Records

CST calculations are done cumulatively. The new record represents the difference between the cum-to-date and the existing CST records. The following list details how CST table record values are calculated:

- ACWP_QTY — User entered or calculated through auto progress
- ACWP_CST — User entered or the individual record's ACWP_QTY multiplied by the unit rate
- ACWP_ESC — User entered or the spread of ACWP _QTY over the From Date to the To Date multiplied by the escalated rates
- BCWP_QTY — Take the sum of the resources used stored in the RES_USED field in the performance measurement baseline in the Baseline Usage table multiplied by the resource PPC and then subtract the values of the existing BCWP_QTY records
- BCWP_CST — Take the sum of the resource costs stored in the RES_CST field in the performance measurement baseline in the Baseline Usage table multiplied by the resource PPC and then subtract the values of the existing BCWP_CST records
- BCWP_ESC — Read the records in the performance measurement baseline in the Baseline Usage table in date order until reaching the cumulative BCWP_QTY. Add the escalated resources stored in the RES_ESC field in the performance measurement baseline for the records and ratio the last values as needed, and then subtract the existing values for BCWP_ESC records.

For example, assume that the cumulative BCWP_QTY = 4, and the resource has the following 3 PMB.BSU records:

Quantity	1	4	6
RES_ESC Cost	$10	$50	$80

To calculate the BCWP_ESC, Open Plan reads the records in the performance measurement baseline in the Baseline Usage table in order until it reaches 4. This returns the first record and three-fourths of the second record. Open Plan then adds the RES_ESC values: 10 + ¾(50). This gives you a BCWP_ESC of 37.5.

Note: When ACWP is being calculated, the fields ACWP_CST and ACWP_ESC are set on the CST records based upon the current rates and escalations. These values are locked in. If ACWP_CST and ACWP_ESC are already non-zero (indicating that they have been calculated previously), the CST record is NOT recalculated.

Cost Calculations

This section discusses the cost calculations Open Plan uses to determine the BAC, BCWS, BCWP, ACWP, ETC, and EAC for a project.

Budget at Completion

Labor Quantity

This value represents the total number of labor resource units budgeted for the project. The Labor Quantity BAC for the project is stored in the BAC_QTY field in the Project table. The value stored in the project field is calculated at or rolled up from the activity level.

The calculations that are performed at the activity level depend on whether the **Use Version 2 Cost Calculation Method** option is selected on the **Cost** tab of the **Project Properties** dialog box.

By default, the **Use Version 2 Cost Calculation Method** is cleared for new projects. In this mode, Open Plan calculates the Labor Quantity BAC by summing the values in the RES_USED field for labor resources assigned to this activity for the Performance Measurement Baseline (PMB) in the Baseline Usage table. By always using the PMB when it is available, Open Plan provides consistency and reliability.

If there is no baseline named PMB, Open Plan uses the currently selected baseline, or, in the case of multiple selected baselines, the baseline identified as **Baseline 1**. If the project contains baselines, but none is selected or named PMB, Open Plan prompts you to select a baseline. If the project has no baselines, Open Plan uses the **Version 2 Cost Calculation Method** automatically.

If the **Use Version 2 Cost Calculation Method** is selected, Open Plan calculates this value by summing the values in the Level field of the Assignment table adjusted for any curve, offset, or period that may be defined.

Cost by Category

The BAC value represent the total cost for the resource budgeted for the project. This value is stored in the appropriate Project field as shown in the following table:

Resource Category	Project Field
Labor Cost	BAC_LAB
Material Cost	BAC_MAT
ODC (Other Direct Costs)	BAC_ODC
Subcontract Cost	BAC_SUB

The BAC value stored in the project field is calculated by summing the values in the BAC field on the Activity table.

The calculations that are performed at the activity level depend on whether the **Use Version 2 Cost Calculation Method** option is selected on the **Cost** tab of the **Project Properties** dialog box.

By default, the **Use Version 2 Cost Calculation Method** is cleared for new projects. In this mode, Open Plan calculates BAC for each category by summing the values in either the unescalated cost field (RES_CST) or the escalated cost field (RES_ESC) in the Performance Measurement Baseline (PMB) on the Baseline Usage table. The value that Open Plan uses in this calculation is determined by a setting on the **Cost Calculations** dialog box:

- If the **Use Resource Cost Escalation** option is cleared on the Cost Calculations dialog box, Open Plan uses the **RES_CST** field.
- If the **Use Resource Cost Escalation** option is selected on the Cost Calculations dialog box, Open Plan uses the **RES_ESC** field.

By always using the PMB when it is available, Open Plan provides consistency and reliability. If there is no baseline named PMB, Open Plan uses the currently selected baseline, or, in the case of multiple selected baselines, the baseline identified as **Baseline 1**. If the project contains baselines, but none is selected or named PMB, Open Plan prompts you to select a baseline. If the project has no baselines, Open Plan uses the **Version 2 Cost Calculation Method** automatically.

If the **Use Version 2 Cost Calculation Method** option is selected, Open Plan calculates the BAC value by summing the values in the Level field of the Assignment table adjusted for any curve, offset, and period that may be defined and multiplying this value by the appropriate category cost rate. The rate that Open Plan uses is determined by a setting on the **Cost Calculations** dialog box:

- If the **Use Resource Cost Escalation** option is cleared on the Cost Calculations dialog box, Open plan uses the unit rate for all entries.
- If the **Use Resource Cost Escalation** option is selected on the Cost Calculations dialog box, Open Plan uses the relevant escalated rates for each entry.

Budgeted Cost of Work Scheduled

Labor Quantity

This value represents the budgeted quantity of labor for the work scheduled for the project from the baseline start date of the project until Time Now. The Labor Quantity BCWS for the project is stored in the BCWS_QTY field on the Project table. This value is calculated by summing the values from the Activity table.

The values in the Activity table are calculated during cost calculations by summing the original (budgeted) labor resource quantity (RES_USED) value in the Performance Measurement Baseline (PMB) in the Baseline Usage table from the baseline start date until Time Now. By always using the PMB when it is available, Open Plan provides consistency and reliability.

If there is no baseline named PMB, Open Plan uses the currently selected baseline, or, in the case of multiple selected baselines, the baseline identified as **Baseline 1**. If the project contains baselines, but none is selected or named PMB, Open Plan prompts you to select a baseline during cost calculations. If the project has no baselines, Open Plan uses the **Version 2 Cost Calculation Method** automatically. A baseline must be selected in order for Open Plan to calculate earned value.

Cost by Category

The BCWS value represent the total budgeted cost of the work scheduled for the project from the baseline start date of the project until Time Now. This value is stored in the appropriate Project field as shown in the following table:

Resource Category	Project Field
Labor Cost	BCWS_LAB
Material Cost	BCWS_MAT
ODC (Other Direct Costs)	BCWS_ODC

Resource Category	Project Field
Subcontract Cost	BCWS_SUB

The BCWS value stored in the project field is calculated by summing the values in the BCWS field on the Activity table.

The values in the Activity table are calculated during cost calculations by summing the values in either the unescalated cost field (RES_CST) or the escalated cost field (RES_ESC) for the Performance Measurement Baseline (PMB) in the Baseline Usage table from the baseline start date until Time Now. The field that Open Plan sums in this calculation is determined by a setting on the **Cost Calculations** dialog box:

- If the **Use Resource Cost Escalation** option is cleared on the Cost Calculations dialog box, Open Plan uses the **RES_CST** field.
- If the **Use Resource Cost Escalation** option is selected on the Cost Calculations dialog box, Open Plan uses the **RES_ESC** field.

By always using the PMB when it is available, Open Plan provides consistency and reliability. If there is no baseline named PMB, Open Plan uses the currently selected baseline, or, in the case of multiple selected baselines, the baseline identified as **Baseline 1**. If the project contains baselines, but none is selected or named PMB, Open Plan prompts you to select a baseline. If the project has no baselines, Open Plan uses the **Version 2 Cost Calculation Method**. A baseline must be selected in order for Open Plan to calculate earned value.

Budgeted Cost of Work Performed

Labor Quantity

This value represents the earned value of labor resource units budgeted for the project. The Labor Quantity BCWP for the project is stored in the BCWP_QTY field on the Project table. This value is calculated by summing the values in the Activity table.

The values in the Activity table are calculated during cost calculations by summing the earned value of resource labor units (BCWP_QTY) in the Resource Cost table.

The earned value of resource labor units (BCWP_QTY) in the Resource Cost Table is calculated during cost calculations, by multiplying the physical percent complete of the assignment by the original (budgeted) labor resource quantity (RES_USED) value in the Performance Measurement Baseline (PMB) on the Baseline Usage table. By always using the PMB when it is available, Open Plan provides consistency and reliability.

If there is no baseline named PMB, Open Plan uses the currently selected baseline, or, in the case of multiple selected baselines, the baseline identified as **Baseline 1**. If the project contains baselines, but none is selected or named PMB, Open Plan prompts you to select a baseline during cost calculations. If the project has no baselines, Open Plan uses the **Version 2 Cost Calculation Method** automatically. A baseline must be selected in order for Open Plan to calculate earned value.

Each time you calculate costs, an entry is added to the resource cost table that represents current period BCWP. For a new period, this value is calculated by subtracting the sum of all previous values from the new total calculated BCWP.

Cost by Category

The BCWP value represents the earned value of costs for the project. This value is stored in the appropriate Project field as shown in the following table:

Resource Category	Project Field
Labor Cost	BCWP_LAB
Material Cost	BCWP_MAT
ODC (Other Direct Costs)	BCWP_ODC
Subcontract Cost	BCWP_SUB

The BCWP value stored in the project field is calculated by summing the values in the BCWP field on the Activity table.

The values in the Activity table are calculated during cost calculations by summing the values in either the unescalated cost field (BCWP_CST) or the escalated cost field (BCWP_ESC) in the Resource Cost table. The field that Open Plan uses in this calculation is determined by a setting on the **Cost Calculations** dialog box:

- If the **Use Resource Cost Escalation** option is cleared on the Cost Calculations dialog box, Open Plan uses the **BCWP_CST** field.
- If the **Use Resource Cost Escalation** option is selected on the Cost Calculations dialog box, Open Plan uses the **BCWP_ESC** field.

Actual Cost of Work Performed

Labor Quantity

This value represents the actual number of labor resource units that have been used by the project. The Labor Quantity ACWP for the project is stored in the ACWP_QTY field on the Project table. This value is calculated by summing the values in the ACWP_QTY on the Activity table.

The ACWP_QTY values in the activity table are calculated during cost calculations based on one of the following conditions:

- If the **Progress Based on Activity Progress** option is selected on the Resource Details dialog box, Open Plan calculates this value **based on the progress reported on the activity.**
- If the **Progress Based on Activity Progress** option is cleared on the Resource Details dialog box, Open Plan calculates this value based on the values that have been entered on the Resources tab of the Activity Progress dialog box. These values are stored in the **ACWP_QTY** (and **ACWP_CST** and **ACWP_ESC**) fields of the Resource Cost table.

Cost by Category

The ACWP value represents the actual costs for the project. This value is stored in the appropriate Project field as shown in the following table:

Resource Category	Project Field
Labor Cost	ACWP_LAB
Material Cost	ACWP_MAT
ODC (Other Direct Costs)	ACWP_ODC

Resource Category	Project Field
Subcontract Cost	ACWP_SUB

The ACWP value stored in the project field is calculated by summing the values in the ACWP field on the Activity table.

The ACWP values in the Activity table are calculated during cost calculations based on one of the following conditions:

- If the **Actual Cost Based on Progressed Quantity** option is selected on the **Resource Details** dialog box, Open Plan calculates this value based on the progressed quantity of the resource and stores it in the ACWP_CST and ACWP_ESC fields on the Resource Cost table. Then Open Plan rolls it up to the activity level.
- If the **Actual Cost Based on Progressed Quantity** option is cleared on the **Resource Details** dialog box, Open Plan calculates this value based on the values that have been entered on the **Resources** tab of the **Activity Progress** dialog box and which are stored in the Resource Cost table. Open Plan summarizes these values to the activity level.

Estimate to Complete

Labor Quantity

This value represents the estimated remaining labor quantity for the project. The Labor Quantity ETC for the project is calculated summing the activity ETC values.

The ETC_QTY values in the activity table are calculated during cost calculations based on one or both of the following conditions:

- If the **Progress Based on Activity Progress** option is selected on the **Resource Details** dialog box, Open Plan calculates remaining values automatically during cost calculations.
- If the **Progress Based on Activity Progress** option is cleared on the **Resource Details** dialog box, ETC_QTY is calculated at the activity level by summing the Remaining values on the Assignment table for Labor resources.

It is possible for some resources to be calculated automatically and some to be calculated manually.

Cost by Category

The ETC value represents the estimate of the remaining costs for the project. The ETC value at the project level is calculated by summing up the activity values stored in the appropriate Project field as shown in the following table:

Resource Category	Project Field
Labor Cost	ETC_LAB
Material Cost	ETC_MAT
ODC (Other Direct Costs)	ETC_ODC
Subcontract Cost	ETC_SUB

The ETC field in the Activity table is calculated by summing the remaining values on the Assignment table for each category. If the **Progress Based on Activity Progress** option is selected on the **Resource Details** dialog box, Open Plan calculates Remaining values during cost calculations.

Estimate at Complete

Labor Quantity

This value represents the total of both the estimated and the completed labor quantities for the project. The Labor Quantity EAC for the project is calculated by first totaling the values in the ETC_QTY and the ACWP_QTY fields in the Activity table and then adding the resultant totals.

Cost by Category

The EAC value represents the estimate of the total costs for the project. The EAC value displayed on the **Cost** tab of the **Project Properties** dialog box is calculated by adding the values in the ETC and the ACWP fields for the appropriate categories..

The EAC field displayed on the Cost tab of the Activity Information dialog box is calculated by adding the ETC and ACWP values from the appropriate categories.

Physical Percent Complete (PPC)

Physical percent complete is percentage of the work of an activity that has been achieved or completed. On the **Resources** tab of the **Activity Progress** dialog box, you can enter the PPC for an activity in two ways:

- At the resource level by entering the PPC in the resource's **Phy. %** field in the grid
- At the activity level by entering the PPC in the **Activity Physical % Complete** field at the top of the tab

The PPC fields at the resource level and at the activity level are interdependent and changing one value affects the other. In other words, when you enter a value in one field, Open Plan automatically calculates and enters the value in the other field in order to keep the two fields synchronized.

The following examples illustrate the two methods of entering PPC.

Example One: Entering PPC at the Resource Level

Assume that the activity has two resource assignments (with linear requirements) as follows:

Resource	Budget (qty)	Unit Cost	Budget Cost	PPC
Res1	10	60	600	50%
Res2	10	50	500	25%

By entering the PPC at the resource level, Open Plan calculates the activity PPC using the following formula:

$$\frac{(BudgetCost_1 \times PPC_1)+(BC_2 \times PPC_2)+(BC_n \times PPC_n)}{BC_1 + BC_2 + BC_n}$$

$$\frac{(600 \times .5)+(500 \times .25)}{600+500} = \frac{300+125}{1100} = 38.6\%$$

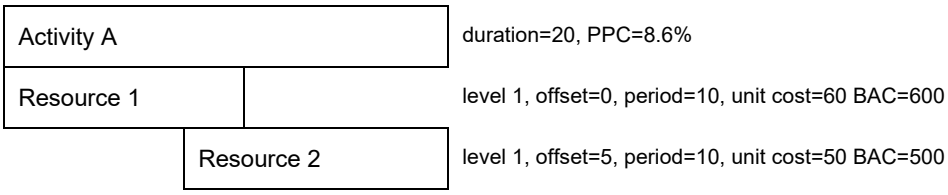
Open Plan enters 38.6 in the **Activity Physical % Complete** field.

> **Note:** This method does not take into account duration, curve, offsets, and periods. It simply reflects what you enter.

Example Two: Entering PPC at the Activity Level

By entering the activity PPC, Open Plan would calculate the resource PPCs with regard to duration, curve, offsets, and periods.

Here is the same information as in the first example but presented in a different format:



Because of the activity duration and the offset of the second resource, the two resources overlap. When you enter the activity PPC of 38.6%, Open Plan makes the following calculations:

First, Open Plan determines the earned value of the activity by multiplying the sum of the resource assignment BACs with the activity PPC you entered:

Earned value for the activity = (activity PPC * (sum of the res. assignment BACs))

= (38.6% * (600+500))

= 424.6

Next, Open Plan looks at the activity profile and determines in which period the earned value falls in relation to the resources:

Profile

R1	60	60	60	60	60	60	60	60	60	60				
R2						50	50	50	50	50	50	50	50 …	

Since only a fraction of the last period is used, Open Plan evenly distributes the earned unit amount between the two resources:

Earned Profile

R1	60	60	60	60	60	60	7.3	(EV = 367.3)
R2						50	7.3	(EV = 57.3)

Cumulative Earned Value

60	120	180	240	300	410	424.6	

Then, Open Plan adds the earned value for each resource and divides by the resource's BAC to determine the PPC for the resource:

PPC

R1 PPC = (367.3 / 600) = 61.2%

R2 PPC = (57.3 / 500) = 11.4%

Calculated Fields

To facilitate the reporting of cost information, Open Plan provides a number of standard calculated fields related to costs at the activity level. These fields are available for display in any view and are defined as follows.

ACWPcum (Actual Cost of Work Performed)

This value represents the actual cost of the work performed on the activity or project from its inception to Time Now.

The ACWPcum is calculated according to the following formula:

```
ACWP_LAB + ACWP_MAT + ACWP_ODC + ACWP_SUB
```

Open Plan has two methods for acquiring actual quantity:

- You may enter the actual quantity directly into the Resources tab of the Activity Progress dialog box.
- You may allow Open Plan to calculate the actual quantity based on the activity status.

The method used depends on settings made on the **General** tab of the **Resource Details** dialog box.

BACcum (Budget at Complete)

This value represents the total budgeted cost of the activity or project and is calculated from the costs of the Project Measurement Baseline (PMB) using the following formula:

```
BAC_LAB + BAC_MAT + BAC_ODC + BAC_SUB
```

If the PMB does not exist, Open Plan uses the currently selected baseline, or, in the case of multiple baselines, the first selected baseline.

BCWPcum (Budgeted Cost of Work Performed)

This value represents the sum of the budgets for the completed work PMB. The cumulative budgeted cost of work performed (BCWP) is frequently referred to as earned value (EV). This value is calculated according to the following formula:

```
BCWP_LAB + BCWP_MAT + BCWP_ODC + BCWP_SUB
```

BCWScum (Budgeted Cost of Work Scheduled)

Activity BCWS (BCWScum) is a calculated field on the activity table that represents the cumulative to-date budgeted cost of work scheduled for the activity and is the sum of the BCWS for each resource category:

```
BCWS_LAB + BCWS_MAT + BCWS_ODC + BCWS_SUB
```

> **Note:** A baseline must be selected in order for Open Plan to calculate the Activity BCWS.

CPI (Cost Performance Index)

This value represents the cost efficiency factor representing the relationship between the actual costs expended and the value of the physical work performed. This value is calculated according to the following formula:

```
BCWPcum / ACWPcum
```

CV (Cost Variance)

This value represents the numerical difference between the earned value (BCWP) and the actual cost (ACWP) and is calculated according to the following formula:

```
BCWPcum – ACWPcum
```

ETC (Estimate to Complete)

This calculated field shows the total ETC for an activity by combining the ETC for each resource category:

```
ETC_LAB + ETC_MAT + ETC_ODC + ETC_SUB
```

EAC (Estimate at Complete)

This value represents the final costs of the activity or project when completed and is calculated from the current resource assignments. This value is calculated according to the following formula:

```
ACWPcum + ETC
```

SPI (Schedule Performance Index)

This value is the planned schedule efficiency factor representing the relationship between the value of the initial planned schedule and the value of the physical work performed, earned value. This value is calculated according to the following formula:

```
BCWPcum / BCWScum
```

SV (Schedule Variance)

This value is used as an indicator of how much a program is ahead or behind schedule and is calculated according to the following formula:

```
BCWPcum – BCWScum
```

VAC (Variance at Complete)

This value represents the algebraic difference between Budget at Complete and Estimate at Complete and is calculated according to the following formula:

```
BACcum – EAC
```

Multi-Project Operations

The extensive set of multi-project features in Open Plan allows you to break large projects down into smaller subprojects, thus providing for high-level summarizations of lower-level detail. The multi-project capabilities can also provide you with a method of rolling up smaller projects into a single master project in order to perform scheduling and resource planning functions at a more comprehensive level.

The multi-project capabilities of Open Plan serve two main purposes. The first is to allow cross project planning and scheduling for departments or smaller organizations. In this scenario, there are a number of individual projects sharing the same pool of resources.

A second use of multi-projects is to provide an integrated program management environment for large complex projects that have been broken down into subprojects. In this scenario, a master scheduler defines a master project as well as the external subproject components and then ownership of each component is given to individual team members.

This topic provides discussions of the following issues related to the multi-project capabilities of Open Plan:

- External subprojects
- The effect of the **Save As** command
- Relationships between projects
- Project target dates
- Data merging
- Baselines
- Cost calculations
- Data structures

External Subprojects

External subprojects are complete projects represented as a single activity in a higher-level (master) project.

When you open an external subproject attached to a master project, the external subprojects will be opened in the same mode as the master project. If it is not possible to open the master project and the external subproject in the same mode, both the external subprojects as well as the master project are opened in the read-only mode.

In cases where you first open a master project and then open an external subproject during a subsequent operation, Open Plan may switch the access mode for the master project if the subproject cannot be opened in the same mode. If you have made changes to the master project, Open Plan allows you to save any changes before switching to the more restrictive mode.

When you elect to save a master project, all open external subprojects are saved back to their original files as well. If you attempt to save data to an external subproject that Open Plan was unable to open with the appropriate access mode (or for which you have insufficient rights), a message box is displayed.

The Effect of the Save As Command

When you are in a master project, it is important to remember that the **Save As** command treats open external subprojects differently than it treats closed external subprojects. When you issue the **Save As** command, Open Plan merges open external subprojects into the master project as internal subprojects. On the other hand, closed external subprojects remain external to the master project, and links to these subprojects are maintained.

> **Note:** You cannot perform **Save As** on a file if there is another file open which refers to it.

Interproject Relationships

Relationships may be entered as usual between any subproject or activity that can be viewed while updating the master project. To add relationships to activities that are part of external subprojects, the external subprojects must be open.

To enforce these constraints when external subprojects are accessed directly (that is, when not accessed as part of the master project), Open Plan inserts foreign activities as predecessors and/or successors to the relevant activities in the external subproject. Such activities have Foreign Activity as their activity type, and their dates are treated as On Target dates. Likewise, resource scheduling schedules these activities using their scheduled dates and assumes the Immediate option. This has the effect of enforcing dependencies created as part of the multi-project environment.

When Open Plan inserts a foreign activity in an external subproject, it also adds the necessary parent activities to make the ID of the foreign activity unique within the external subproject. These foreign parents also serve as a means of tracking where the external link comes from (that is, the project to which they refer). It is important to note that no operation should be performed on foreign activities or their parents.

Note: The **RetainIRLsOnSaveAs** system preference allows you to set Open Plan to either retain or delete the IRLs when you are performing a project **Save As** on a project. Setting it to **1** retains the IRLs in the saved project. By default, it is set to **0**.

Project Target Dates

When saving external subprojects from within a master project, Open Plan checks for the existence of a foreign subproject or foreign activity. If none exists, Open Plan sets a target project finish date for the external subproject. If necessary, Open Plan also sets a project target start to create a relationship with the external subproject activity.

The project start date that is created could overwrite a target start date previously set for the master project.

Data Merging Issues

When external subprojects are opened from within a master project, the data from the subproject is effectively merged into the current project. To perform this operation, Open Plan validates that the master project and all external subprojects use the same ancillary files.

When you open a hierarchical project in Open Plan that contains external subprojects, the program checks that the master project has the same ancillary data files attached to the various external subprojects. For example, if you have attached two code files to an external subproject, the master project must have the same two code files attached to it as well.

Open Plan can make these ancillary file assignments only when either the master project or the external subproject does not have an ancillary file of that type assigned. If, for example, the master file uses Resource File A and the external subproject uses Resource File B, Open Plan will not be able to merge the data and the external subproject project will not open.

For the purposes of merging data, Open Plan ignores the Default calendar file. This means that if, for example, an external subproject uses the Default calendar file and the master project uses another calendar file, the calendar file that is assigned to the master project will also be assigned to the external subproject.

In the case of code files, Open Plan also validates that the master project and all external subprojects attach the code file to the same code field and makes the appropriate assignments where necessary. This can have the result of assigning the same code file to more than one code field.

For example, assume that the master project has a code file named Location assigned to Code Field 1 and that Code Field 2 is empty. Assume also that an external subproject has a code file named Location assigned to Code Field 2 and that Code Field 1 is empty. When the data is merged, Open Plan assigns Location to Code Field 2 in the master project and to Code Field 1 in the external subproject.

Baseline Issues

Security permitting, a baseline created at the master project level is visible in the subprojects. The owner of the baseline can grant rights to other users for viewing and/or changing the baseline. Thus, users who have authorization are able to access and update baselines from an external subproject.

When a baseline is selected, the baseline data is opened for the main project and for all open subprojects. When a subproject is opened subsequent to the selection of a baseline, the subproject baseline data needs to be read.

Cost Calculations

When you perform cost calculations for a master project that includes external subprojects, Open Plan calculates the budgeted and actual costs for any open external subprojects and then rolls up the costs to the master project level. This activity-level cost information is saved back to the external subprojects when a save is performed.

The calculation of budgeted costs is based on the information stored in the resource file assigned to the project at that time. (The same holds true for the calculation of actual costs if the **Actual Cost Based on Progress Quantity** option has been selected.) Thus, even if you have previously calculated the cost at the subproject level, these values are recalculated when you run cost calculations at the master project level.

Since the cost information related to activities is saved back to the external subproject level but the information from the resource file is not, it is possible that you will get different costs totals depending on whether you calculate costs in the master project or in the external subproject. To verify which unit costs will be used to calculate activity and resource costs, check the resource file assigned to the project before performing the calculation.

Reporting

Customizing Barchart Views

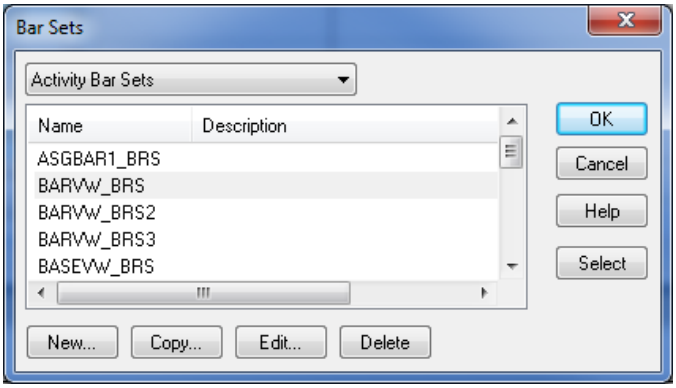
Open Plan provides you with complete control over the appearance of bars within a barchart view. This control allows you to create customized reports based on the standard barchart views by changing options such as:

- The conditions under which the bar is displayed
- The dates represented by the bar
- The appearance and characteristics of the bar, symbols, and accompanying text

Defining Bar Sets

A bar set is a group of activity or resource assignment bars that you can assign to a barchart view. For example, you might have a particular style of bar to show a critical activity and another bar to show an activity that is not critical.

You can change how Open Plan displays bars in a barchart through the **Bar Sets** dialog box. Open Plan displays this dialog box when you click **Bar Sets** on the **View** tab or when you right-click anywhere in the barchart portion of the view and click **Bar Sets** on the context menu.



The two sets of standard bar sets that are selectable in the top-most field of the **Bar Sets** dialog box are:

- Activity Bar Sets
- Multi-table Bar Sets

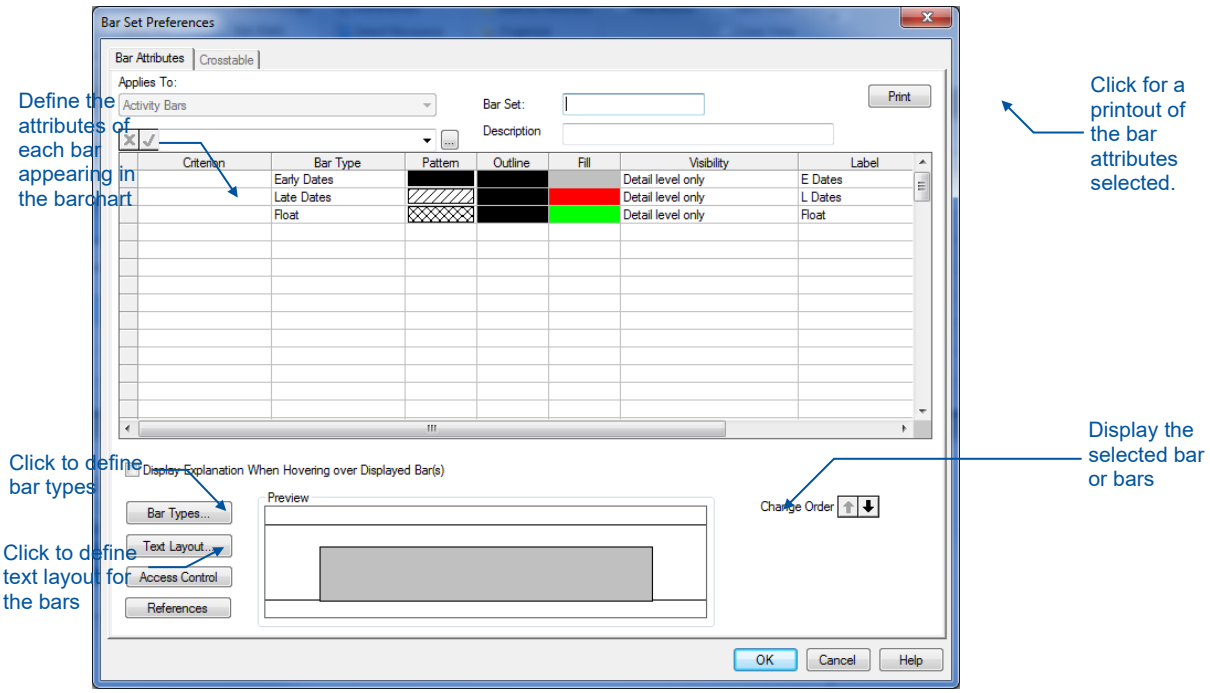
The Bar Sets dialog box allows you to perform the following functions:

- Select the bar set to which the bar views apply
- Assign a bar set to the current view simply by selecting it from the list and clicking OK
- Create a new bar set that can be used with any barchart view
- Copy an existing bar set and modify it to meet new requirements
- Edit an existing bar set
- Delete a bar set that is no longer required

On the **Bar Sets** dialog box, clicking **New** or selecting a bar set and clicking **Copy** or **Edit** displays the **Bar Attributes** tab of the **Bar Set Preferences** dialog box.

Bar Set Attributes

In Open Plan, bar attributes specify the criterion an activity must meet in order to display a particular bar. Each bar attribute consists of a bar label (for the legend), a filter expression, and a definition of the bar type and display characteristics. User-defined bar attributes are a flexible and efficient way of customizing barchart views, making it possible to have the display of the bars driven by any information stored for the activity in Open Plan. For example, you could indicate varying degrees of criticality by displaying differently colored bars for non-critical, critical, and super-critical activities. You might wish to display high-level summary bars for most activities, with a smaller number of key activities exploded to show more detail.



The preview area at the bottom of the dialog box displays the currently selected bar attribute. You can select multiple attributes and preview how different bars will look together.

The **Print** button allows you to print the attributes chosen in the **Bar Set Preferences** dialog box. This is useful when modeling other bar sets after one another.

> **Attention:** For more information and field descriptions, see **Views and Reports » Barchart View » Tabs and Dialog Boxes » Bar Set Preferences Dialog Box** in the Deltek Open Plan online help.

To customize bar attributes

1. Take one of the following actions:
   - On the **View** tab, click **Bar Sets**.
   - Right-click an empty area in a barchart view, and click **Bar Sets** on the context menu.
2. From the **Bar Sets** dialog box, select the bar set you would like to edit and click **Edit**.
3. On the **Bar Attributes** tab of the **Bar Set Preferences** dialog box, enter the information for each set of attributes that you want to customize.

- To define a new set of attributes, scroll to the empty line at the bottom of the list, and enter the information for the attributes.
- To delete a set of attributes, select the row and press the **Delete** key.

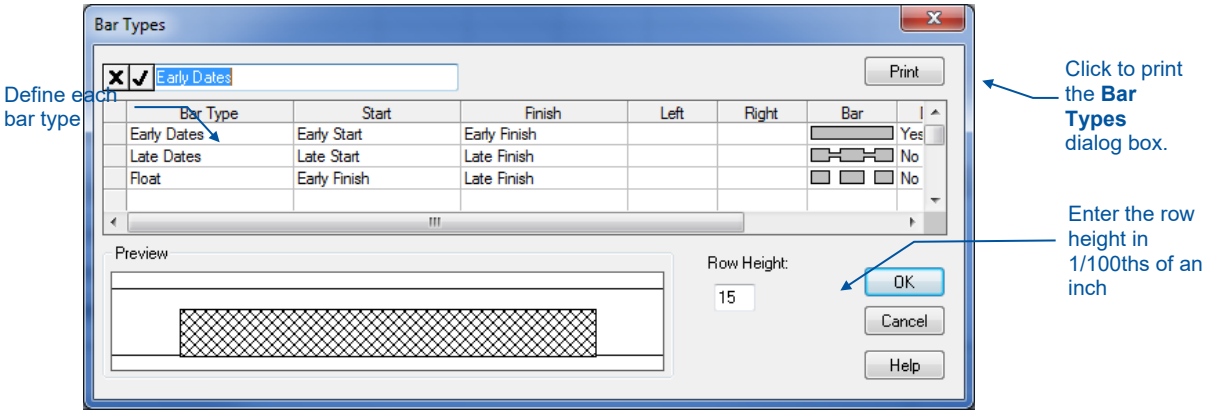
4. When your changes are complete, click **OK** to return the **Bar Sets** dialog box.

   To display the bar set you just edited, select it from the list and click either **OK** or **Select**.

Defining Custom Bar Types

Each bar type in Open Plan represents a definition of a bar: the start and finish dates used to position the bar, the symbols placed to the left and right of the bar, the shape and dimensions of the bar, and so on. Although the barchart views supplied with Open Plan define many standard bar types, it is also possible to customize these bar types or to create new types.

To define a custom bar type, click **Bar Types** on the **Bar Attributes** tab of the **Bar Set Preferences** dialog box. Open Plan responds by displaying the **Bar Types** dialog box:



Attention: For more information and field descriptions, see **Views and Reports » Barchart View » Tabs and Dialog Boxes » Bar Types Dialog Box** in the Deltek Open Plan online help.

To customize bar types, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Bar Sets**.
   - Right-click an empty area within the barchart, and click **Bar Sets** on the context menu.
2. From the **Bar Sets** dialog box, select the bar set you would like to edit and click **Edit**.
3. On the **Bar Attributes** tab of the **Bar Set Preferences** dialog box, click **Bar Types**.
4. Enter the information for each bar type you want to customize.
   - To define a new bar type, scroll to the empty line at the bottom of the list, and enter the information for the bar type.
   - To delete an existing bar type, click the bar type row and press the **Delete** key.
   - To select multiple bar types, press the **Shift** or **Ctrl** keys and click the selection buttons to the left of the desired rows.
   - To adjust the row height for the bar type, enter a value in the **Row Height** field in hundredths of an inch.

5. When your changes are complete, click **OK** to return to the **Bar Attributes** tab of the **Bar Set Preferences** dialog box.

Defining Text Layout for Bars

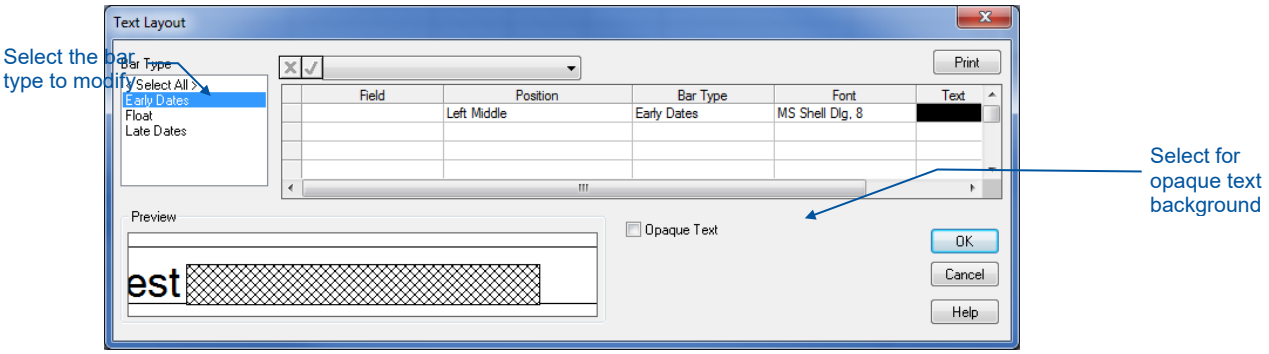
For each bar type, you can define as many as 15 different positions in which text is displayed. The following illustration and table shows the possible positions for bar text:

1	4	7	10	13
2	5	8	11	14
3	6	9	12	15

1 – Left top
2 – Left middle
3 – Left bottom
4 – Center left top
5 – Center left middle
6 – Center left bottom
7 – Center top
8 – Center middle
9 – Center bottom
10 – Center right top
11 – Center right middle
12 – Center right bottom
13 – Right top
14 – Right middle
15 – Right bottom

You can display any text, numeric, or date field for the activity using this feature.

To define the text layout of a bar, click **Text Layout** on the **Bar Attributes** tab of the **Bar Set Preferences** dialog box. Open Plan responds by displaying the following dialog box:



To use an opaque background for bar text, click **Opaque Text**. If you do not click this option, items such as relationship lines will display behind the text.

> **Note:** If you want a displayed field to include literal text (for example, “Total Float =”) as a label for activity data, define a calculated field for the project using a text string and the appropriate data. For information about defining calculated fields, see Defining Calculated Fields.
>
> For more information and field descriptions, see **Views and Reports » Barchart View » Tabs and Dialog Boxes » Text Layout Dialog Box** in the Deltek Open Plan online help.

To customize text layout, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Bar Sets**.
   - Right-click an empty area within the barchart and click **Bar Sets** on the context menu.

2. From the **Bar Sets** dialog box, select the bar set you would like to edit and click **Edit**.
3. On the **Bar Attributes** tab of the **Bar Set Preferences** dialog box, click **Text Layout**.
4. Enter the information for the text to be display for each bar type.
   - To attach a new text field to a bar type, enter the information for the field on a new line.
   - To delete an existing assignment, select the row with the assignment and press the **Delete** key.
   - To define the font used to display the text, click the **ellipsis** button next to the edit box for that setting.
5. When your changes are complete, click **OK** to return to the **Bar Attributes** tab of the **Bar Set Preferences** dialog box.

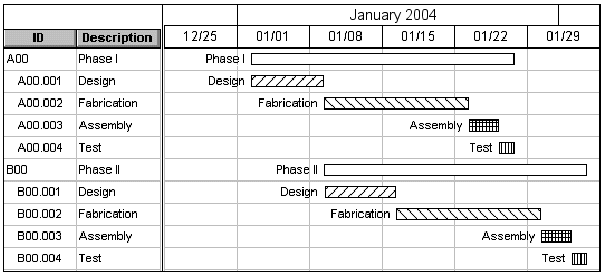
Examples of Custom Bars

By setting up different combinations of bar attributes and bar types, you can produce customized barchart views that can display selected levels of summarization or detail. In the following examples, you can see how to define three sets of special-purpose bars in Open Plan:

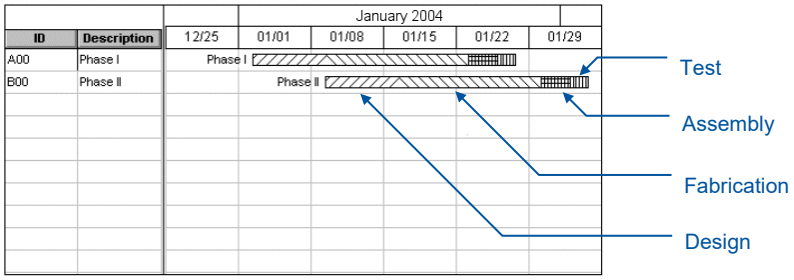
- Summary bars based on an activity code
- Summary bars with milestones
- Summary bars with major milestones

Summary Bars Based on an Activity Code

For this example of summary bars, assume that you want to produce a barchart in which bars showing early dates have fill patterns that correspond to an activity code that represents different organizational responsibilities:



When you roll up the detail bars using outlining, you want the summary bars to continue to display the different fill patterns:



To set up this type of barchart, you need to define a single bar type to show early dates:

Bar Type	Start	Finish	Left	Right	Bar	Drag	Height	Offset	Key
Early Dates	Early Start	Early Finish				Yes	10	9	Yes

Assume that you want an activity code stored in a code field to drive the fill patterns for the bars:

Code 1 Value	Description
D	Design
F	Fabrication
A	Assembly
T	Test

You could then set up filters based on the different code values (these will serve as bar criteria) and then define the following attributes for the detail bars and a summary bar:

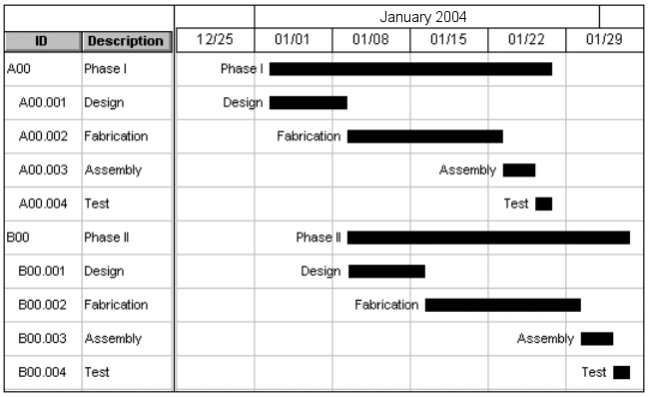
Criterion	Bar Type	Pattern	Outline	Fill	Visibility	Label
Code_D	Early				Detail and immediate rollup level	Design
Code_F	Early				Detail and immediate rollup level	Fabrication
Code_A	Early				Detail and immediate rollup level	Assembly
Code_T	Early				Detail and immediate rollup level	Test
Summary	Early				Detail level only	Summary

Notice that while all bars use the same bar type, the various criteria serve to associate particular codes with different fill patterns.

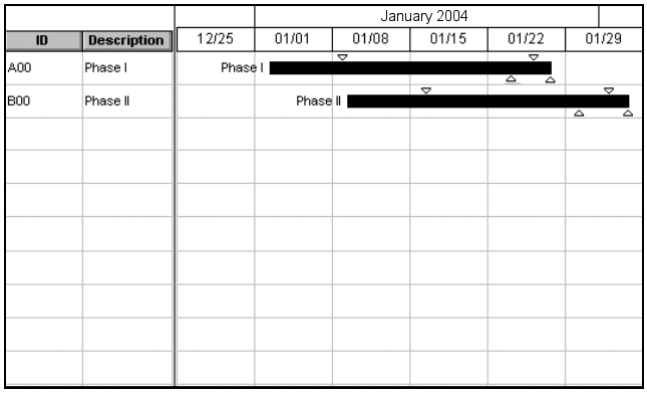
Notice also the use of the visibility parameter to control the display of detail and summary bars. Bars with the code-based fill pattern will display bars at both the detail level and at the summary level immediately above the detail level. This allows the bars to be displayed for detail activities as well as when the activities are rolled up to the next summary level. The summary bar, on the other hand, appears at the detail level only.

Summary Bars with Milestones

Another common reporting requirement is to show multiple milestones for a single summary bar. For example, assume that your project consists of the following detail and summary activities:



When rolled up, the bars would look like this:



To achieve this effect, you must first define three bar types: an early bar showing early dates for detail and summary bars, and two special bars for milestones that appear above and below the summary bar when the detail activities are rolled up:

Bar Type	Start	Finish	Left	Right	Bar	Drag	Height	Offset	Key
Early	Early Start	Early Finish				No	10	10	Yes
MUp		Early Finish		△	<None>	No	9	1	No
MDown		Early Finish		▽	<None>	No	9	21	No

Notice that the two milestone bar types do not have a left symbol or a bar shape. When these bars appear in a barchart, only the right symbol is displayed. Notice also how the height and offset parameters for the milestone bars straddle the position occupied by the early bar.

To understand how the bar attributes could be set up for this type of barchart, assume that you want an activity code to determine whether an activity milestone should be displayed above or below the summary bar:

Code 2 Value	Description
U	Above the summary bar
D	Below the summary bar

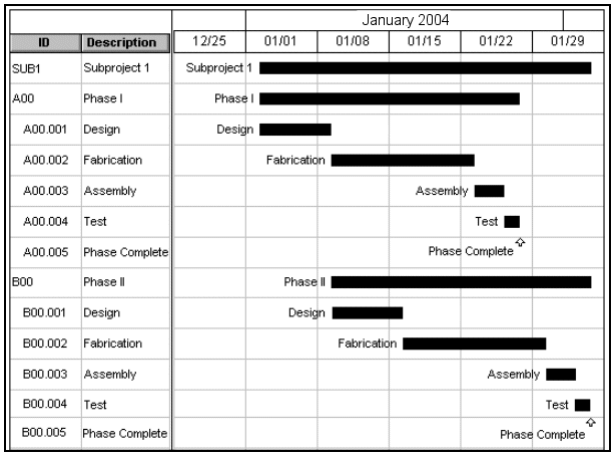
You could then set up filters based on the two code values (these will serve as bar criteria) and then define the following attributes for the bars:

Criterion	Bar Type	Pattern	Outline	Fill	Visibility	Label
Code_U	MUp				Immediate rollup level only	Milestone
Code_D	MDown				Immediate rollup level only	Milestone
Detail	Early				Detail level only	Detail
Summary	Early				Detail level only	Summary

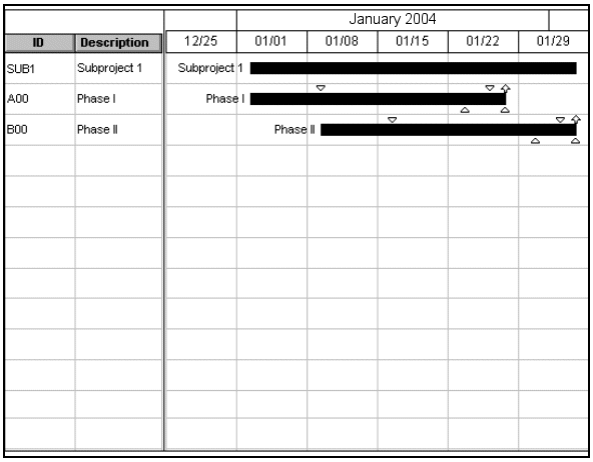
As in the previous example, notice the use of the visibility parameter to control the display of detail and summary bars. In this case, early bars are displayed for detail activities; when rolled up to the next level, however, the activities appear as milestones. The value of the **Code 2** field determines whether the milestone marker for the activity appears above or below the summary bar.

Summary Bars with Major Milestones

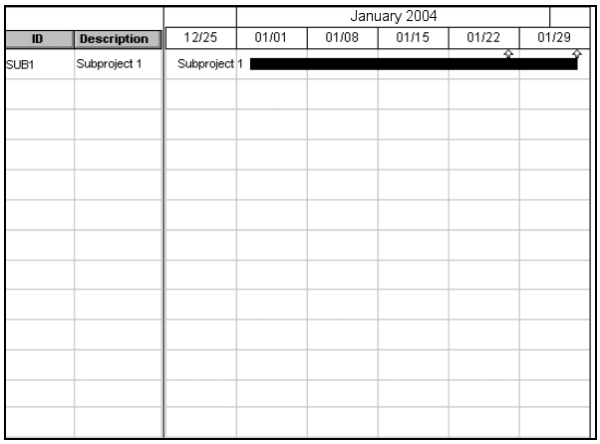
For high-level executive summaries, it is useful to set up barcharts in which detail activities can be rolled up through a series of milestones representing higher levels of summarization. For example, assume that a subproject has been broken down into two phases:



When rolled up to the next level, the detail activities appear as milestones, with two milestones representing the completion dates of each phase:



When rolled up to the highest level of summarization, only the major milestones appear:



Displaying major milestones requires defining a bar type in addition to the bar types used to display lower-level milestones:

Bar Type	Start	Finish	Left	Right	Bar	Drag	Height	Offset	Key
Early	Early Start	Early Finish				No	10	10	Yes
MUp		Early Finish		△	<None>	No	9	1	No
MDown		Early Finish		▽	<None>	No	9	21	No
Major		Early Finish		⇧	<None>	No	9	1	No

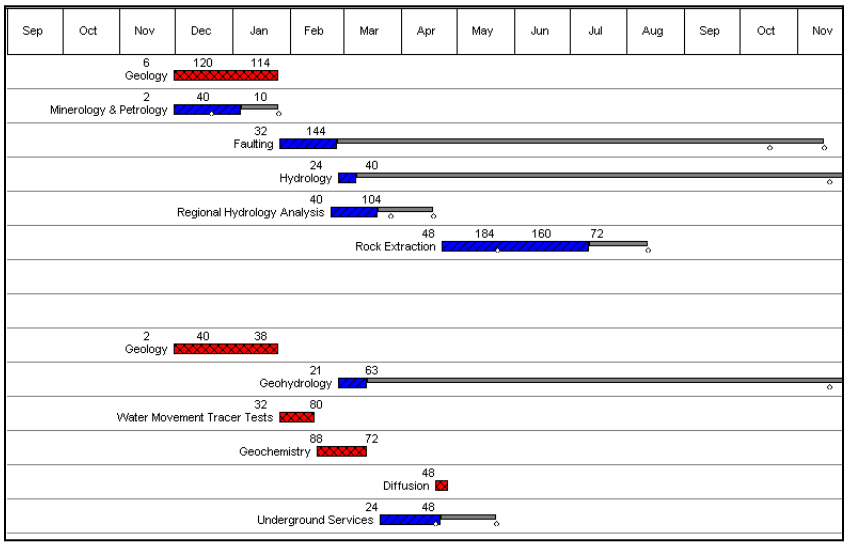
Bar attributes for this barchart would be as follows:

Criterion	Bar Type	Pattern	Outline	Fill	Visibility	Label
Code_U	MUp				Immediate rollup level only	Milestone
Code_D	MDown				Immediate rollup level only	Milestone
Detail	Early				Detail level only	Detail
Summary	Early				Detail level only	Summary
Code_M	Major				All levels of rollup and detail	Major Milestone

For this example, a bar attribute for major milestones is defined, and another value for Code 2 (M for major milestones) is utilized for the criterion. The visibility parameter for major milestones is set to be visible at all levels.

Displaying Time-Phased Data

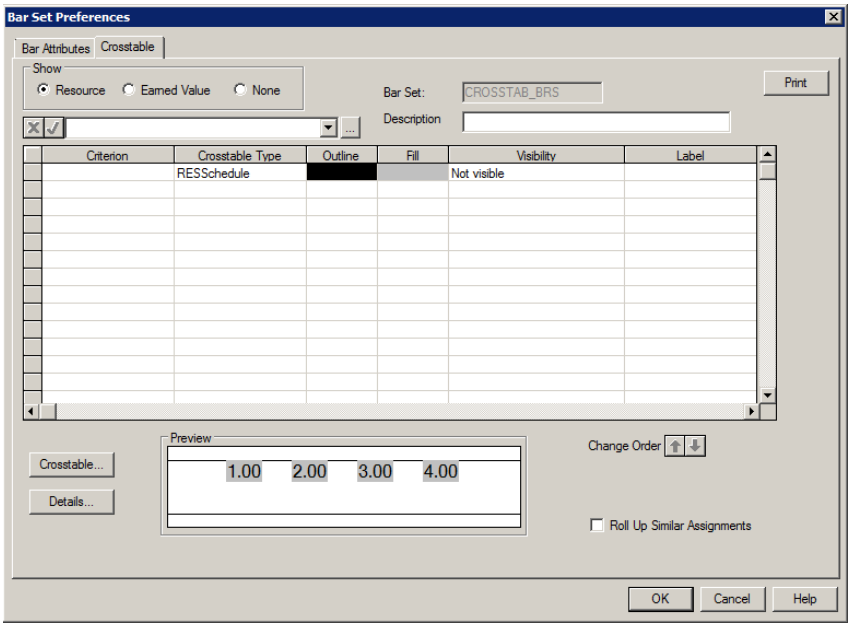
The standard multi-table barcharts in Open Plan allow you to display time-phased cost and resource data in the barchart portion of the view:



You can display cost and resource information based on either resources or earned value (forecasts, BCWS, BCWP, or ACWP). Open Plan time-phases this information based on the smallest time period defined by the barchart date scale. Thus, the information displayed by a data bar corresponds to the cost information displayed in a resource histogram that includes a table.

Crosstable Tab

To control the display of time-phased data, multi-table barchart views include an additional tab in the **Barchart Preferences** dialog box — the **Crosstable** tab:



Similar to the **Bar Attributes** tab, the **Crosstable** tab allows you to define a number of parameters for cost data “bars”:

- Criterion
- Crosstable bar type (which corresponds to the Bar Type setting on the Bar Attributes tab)
- Outline (used to specify the font color)
- Fill color
- Visibility (five levels of detail):
- All levels of rollup and detail
- Detail and immediate rollup level
- Detail level only
- Immediate rollup level only
- Not visible
- Label

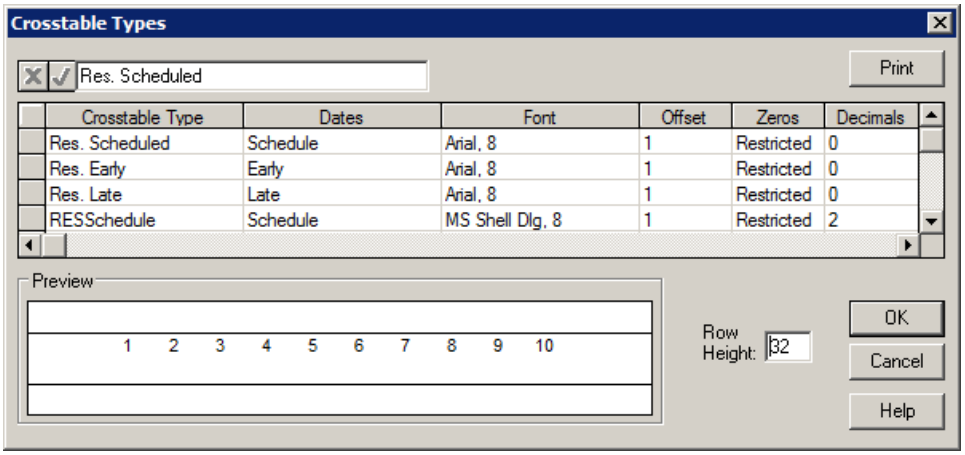
> **Attention:** For a complete description of each of these parameters, see Bar Set Attributes.

You can also use this tab to indicate if you want to display resource costs, earned value costs, or suppress the display of costs altogether. The **Crosstable** tab controls the set of bar attributes appear in the dialog box.

The **Crosstable** tab also includes buttons for two commands: the **Crosstable** command and the **Details** command. The following sections describe each of these commands.

The Crosstable Command

If you click the **Crosstable** button on the **Crosstable** tab, Open Plan displays the **Crosstable Types** dialog box:



Note: This dialog box corresponds to the **Bar Types** dialog box that you can use to define custom bar types.

- 06_07.jpg

- Use the Crosstable Types dialog box to define the following information for each crosstable bar type:

- **Crosstable Type** — The name of the crosstable type typically reflects the dates represented by the bar. For example, the crosstable type RESchedule might refer to data based on schedule dates.

- **Dates** — If you are defining a bar to display resource data, you can define one of the following pairs of dates to use for the bar:

- **Early** — Resource assignments based on the early dates calculated by time analysis

- **Late** — Resource assignments based on the late dates calculated by time analysis

- **Schedule** — Resource assignments based on the scheduled dates calculated by resource scheduling (these assignments can show the effects of splitting, stretching, and reprofiling activities)

- **Baseline 1, 2, and 3** — Baseline resource assignments using the dates (early, late, or scheduled) selected at the time that you created the baseline(s) currently attached to the project

- **Actual** — Actual resource costs or quantities based on resource progress information

- **Availability** — Resource availabilities

  If you are defining a bar to display earned value, use the **Dates** setting to display one of the following types of costs:

- **Forecast** — Open Plan determines the forecast cost or quantity (Estimate At Complete or EAC) of a resource by adding any actuals prior to Time Now to any planned costs or quantities subsequent to Time Now.

- **Budgeted Cost of Work Scheduled (BCWS)** — BCWS is derived from the planned resource budget stored in the current baseline for the project. This budget may be based on early, late, or scheduled dates, depending on the dates used to create the baseline.
- **Actual Cost of Work Performed (ACWP)** — ACWP is based on the actual costs recorded for the resource.
- **Budgeted Cost of Work Performed (BCWP)** — To calculate BCWP, Open Plan applies the value for the physical completion of the activity (or activities) to the corresponding portion of the planned resource budget (as defined in the current project baseline). For example, if you indicate that an activity has a physical complete value of 50%, Open Plan calculates BCWP for an assigned resource by determining how much of its planned budget (BCWS) corresponds to the first half of the activity duration.
- **Font** — Specifies the font.
- **Offset** — Defines the distance from the top of the bar to the top of the row. Specify this setting in hundredths of an inch.
- **Zeros** — Controls the display of zero values in a bar as follows:
- **Ignore** — Never display zero values
- **Restricted** — Restrict the display of zero values to the date span defined for the bar
- **Always** — Display zero values for each time period defined for the barchart
- **Decimals** — Controls the number of decimal places that Open Plan should display on the bar.

The Crosstable Types dialog box also includes a setting for defining the row height in hundredths of an inch.

> **Attention:** For more information about defining row height, see Defining Custom Bar Types.

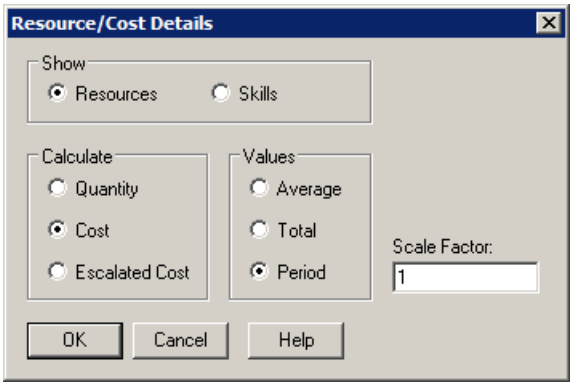
To customize crosstable bar types, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Bar Sets**.
   - Right-click an empty area within the barchart and click Bar Sets.
2. From the Bar Sets dialog box, select the bar set you would like to edit and click Edit.
3. On the Bar Set Preferences dialog box, click the Crosstables tab.
4. Indicate whether you want to display resource or earned value data, and click Crosstable.
5. Enter the information for each crosstable bar type you want to customize.
   - To define a new bar type, scroll to the empty line at the bottom of the list and enter the information for the crosstable type.
   - To delete an existing crosstable type, click the crosstable type row and press the **Delete** key.
   - To select multiple crosstable types, press the **Shift** or **Ctrl** keys and click the selection buttons to the left of the desired rows.
6. When your changes are complete, click **OK** to return to the **Crosstable** tab.

The Details Command

If you click the **Details** command from the **Crosstable** tab, Open Plan displays a dialog box that allows you to define the information appearing in a bar. The type of dialog box that appears depends on whether you choose to display resource or earned value data.

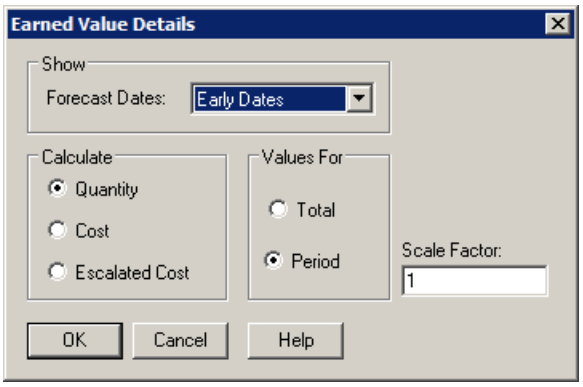
If you have indicated that you want to display resource data, Open Plan displays the following dialog box when you click the **Details** button:



Use this dialog box to enter the following information:

- **Show** — You can display values for one of the following:
- **Resources** — Select this option to display details for resources.
- **Skills** — Select this option to display details for skills.
- **Calculate** — You can have Open Plan display resource data in terms of either resource units, base unit costs, or escalated costs.
- **Values** — You can control which values appear with one of the following options:
- **Average** — Resource costs represent an average value per the default duration unit used by the project. Open Plan calculates this average over the smallest time unit displayed on the date scale and uses the default project calendar to determine valid working periods. (If you have not assigned a calendar to the project, Open Plan assumes a 5-day, 40-hour work week with no holidays.)
- **Total** — Total resource costs or quantities.
- **Period** — Resource data corresponds to the smallest time unit displayed on the date scale.
- **Scale Factor** — You can define a scale factor for the information so that, for example, all values represent thousands of dollars.

If you have indicated that you want to display earned value data, Open Plan displays the following dialog box when you click **Details** on the **Crosstable** tab:



Use this dialog box to enter the following information.

- Show
- **Forecast Dates —** If you choose to display time-phased forecast data, you can base planned costs or quantities on early, late, or scheduled dates.
- **Calculate** — You can have Open Plan display earned value in terms of either resource units, base unit costs, or escalated costs.
- **Values for** — You can control which values appear in the view with one of the following options:
- **Total** — Earned value costs represent the total for the resource.
- **Period** — Earned value data corresponds to the smallest time unit displayed on the date scale.
- **Scale Factor** — You can define a scale factor for the information so that, for example, all values represent thousands of dollars.

To define crosstable details, complete the following steps:

1. Take one of the following actions:
   - On the View tab, click **Bar Sets**.
   - Right-click an empty area within the barchart, and click **Bar Sets**.
2. From the Bar Sets dialog box, select the bar set you would like to edit and click **Edit**.
3. On the Bar Set Preferences dialog box, select the Crosstables tab.
4. Indicate whether or not you want to display resource or earned value data, and click **Details**.
5. Enter the information for the contents of the cost data bars.
6. When your changes are complete, click **OK** to return to the Crosstables tab.

Customizing Network Views

You can customize both the appearance and the contents of activity boxes displayed in a network view. This document describes the facilities for customizing box attributes and text.

Attention: For information about modifying other aspects of a network view, see **Views and Reports » Network View** in the Open Plan online help.

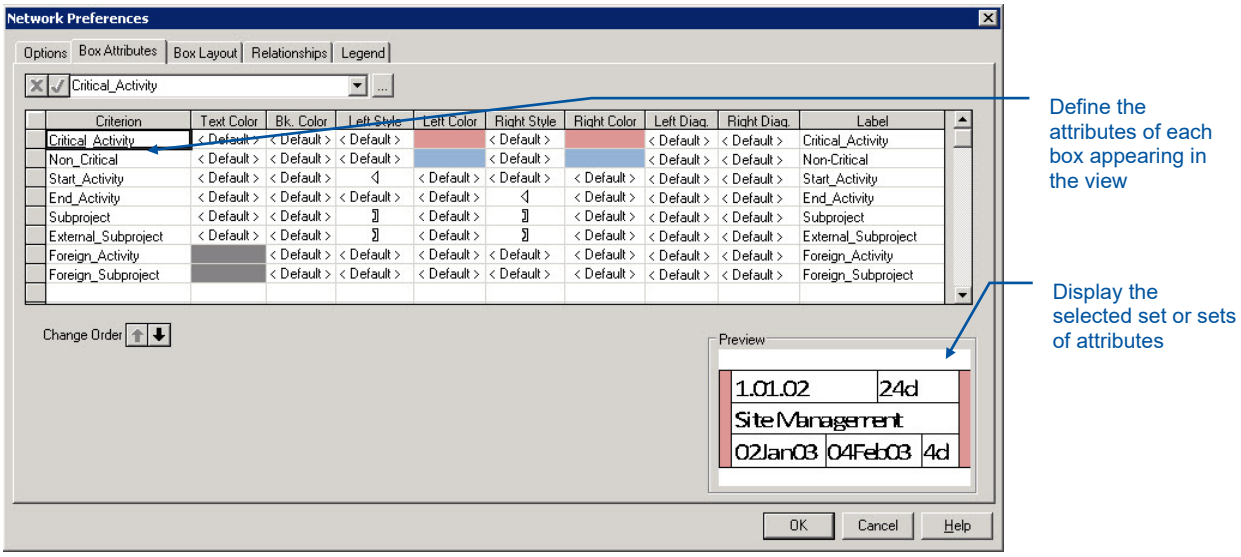
Custom Box Attributes

Open Plan allows you to control the appearance of activity boxes in a network view by using data-driven box attributes. Like bar attributes in a barchart view, box attributes specify the criterion an activity must meet to display a particular style of box in a network view.

Each set of box attributes consists of a filter expression, a definition of how an activity satisfying the filter expression should appear in the view, and a box label for the legend. You can use these attributes to define the following characteristics of a box:

- The color of the lines and text of the box
- The shape of the box ends
- The fill color appearing within the box ends
- The presence or absence of diagonal markings in the box

To set custom attributes for boxes, display the **Box Attributes** tab of the **Network Preferences** dialog box:



Open Plan uses a set of standard attributes to display boxes for all activities that do not satisfy any of the listed criteria. These standard attributes are as follows:

- Text color — black
- Left end style — square ([ )
- Right end style — square ( ])
- Left fill color — white
- Right fill color — white
- Left diagonal — none
- Right diagonal — none
- Label — blank

This set of standard attributes also serves as the basis for all subsequent attributes appearing below it on the list. This feature of obtaining characteristics from previously defined attributes allows for activity boxes that can display different combinations of attributes by satisfying two or more criteria.

You can customize each box attribute with the following settings:

- **Criterion** — The box criterion defines the filter condition that the activity must meet in order to display a specific type of box. When specifying a box criterion, you can select an existing filter from the list.
- **Text Color** — The color of the activity box and text
- **Bk. Color** — The background color.
- **Left Style** — The end style used for the left edge of the box. Open Plan allows you to select an end style from a list displayed when you enter the field.
- **Left Color** — The color at the left end of the box.
- **Right Style** — The end style used for the right edge of the box
- **Right Color** — The color at the right end of the box.
- **Left Diag** — The use of the left diagonal mark (\)
- **Right Diag** — The use of the right diagonal mark (/)
- **Label** — Box labels appear next to boxes in the network legend. If you do not enter a label for a set of attributes, Open Plan uses the name of the criterion for the box label.

All attributes other than box criteria and labels have a possible setting of **<Default>**. If you select this setting, the box obtains the setting from a set of attributes appearing higher on the list.

> **Attention:** For more information about the combination of box attributes, refer to the “Combining Box Attributes” section below.

To define custom box attributes, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click an empty area within the view, and click **Preferences** on the context menu.
2. On the **Network Preferences** dialog box click the **Box Attributes** tab.
3. Enter the information for each attribute.
   - To display all of the attribute settings in the dialog box, use the scroll bar.
   - To delete an attribute, select any cell in the row, and press the **Delete** key.
   - To select multiple sets of attributes, press the **Shift** or **Ctrl** keys and click the selection buttons to the left of the desired rows.
4. When your changes are complete, click **OK** to return to the view.

Combining Box Attributes

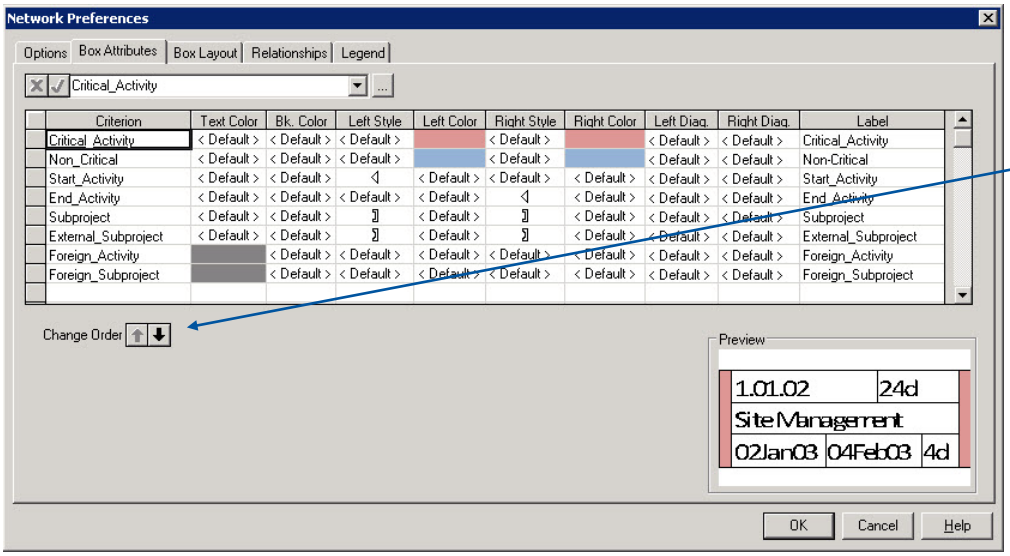
In one important respect, defining box attributes for a network view differs from defining bar attributes for a barchart view. In a barchart view, bar attributes are never combined — each bar attribute affects the

appearance of a bar without regard for other attributes. In a network view, different box attributes can be combined for activities that satisfy more than one criterion.

To produce these combinations, Open Plan uses the settings from the lowest-listed criterion satisfied by the activity to display the box. Any undefined attributes obtain their values from relevant criteria appearing higher in the list.

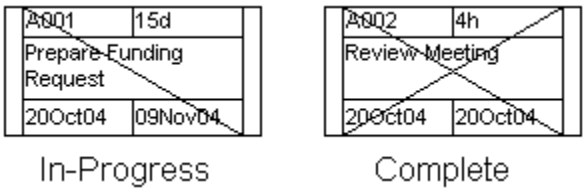
At the highest level of the attribute hierarchy is a standard set of attributes. Open Plan uses these standard attributes to display activities that do not satisfy any of the listed criteria as well as to provide settings for undefined attributes.

Since attributes appearing at the bottom of the list can obtain values from attributes listed above them, the order in which attributes appear in the dialog box is significant.

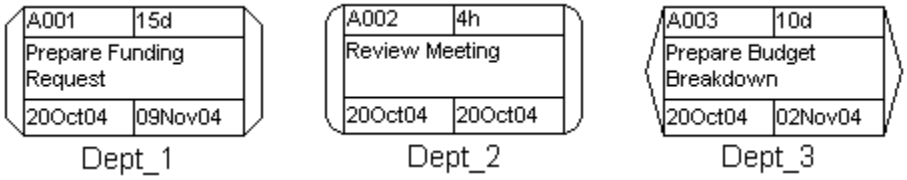


To understand how the combination of box attributes works, imagine that you want to base the appearance of boxes on three sets of overlapping criteria: activity status, responsibility, and criticality.

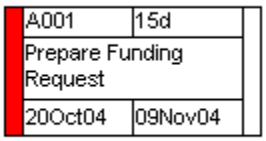
For activity status, assume that you want to use diagonal markings to indicate if an activity is either in-progress or complete:



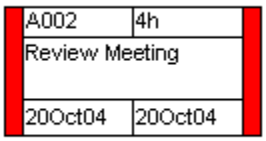
To indicate organizational responsibility for the activity, you want to use the left and right end styles to represent the following codes from an OBS code file:



Finally, you want the fill of the box ends to indicate how critical the activity is:



Critical



Super-critical

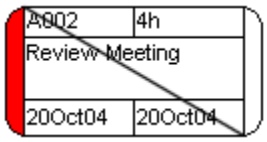
It is possible to express all the possible combinations of these conditions by defining 27 different criteria for activities, each with its own set of attributes. It is also possible to set up just 7 criteria as the following table illustrates.

Criterion	Text Color	Left Style	Left Fill	Right Style	Right Fill	Left Diag.	Right Diag.
In progress	<D>	<D>	<D>	<D>	<D>	\	<D>
Complete	<D>	<D>	<D>	<D>	<D>	\	/
Dept_1	<D>	Blunt	<D>	Blunt	<D>	<D>	<D>
Dept_2	<D>	Round	<D>	Round	<D>	<D>	<D>
Dept_3	<D>	Angle	<D>	Angle	<D>	<D>	<D>
Critical	<D>	<D>	Black	<D>	<D>	<D>	<D>
Super-critical	<D>	<D>	Black	<D>	Black	<D>	<D>

Note: In the above table, **<D>** represents the **<Default>** value for a setting.

To understand how Open Plan displays a given activity box based on these attributes, start at the bottom of the list and find the first criterion satisfied by the activity. Any attributes defined for that row are displayed. To determine settings for undefined attributes, continue up the list to the next criterion that the activity satisfies. If an attribute remains undefined after you have examined all the listed criteria, Open Plan uses the standard setting for that attribute.

For example, a critical in-progress activity that is assigned to Dept_2 would be displayed as follows:



This specific combination of attributes is the result of Open Plan applying the following rules when displaying the activity:

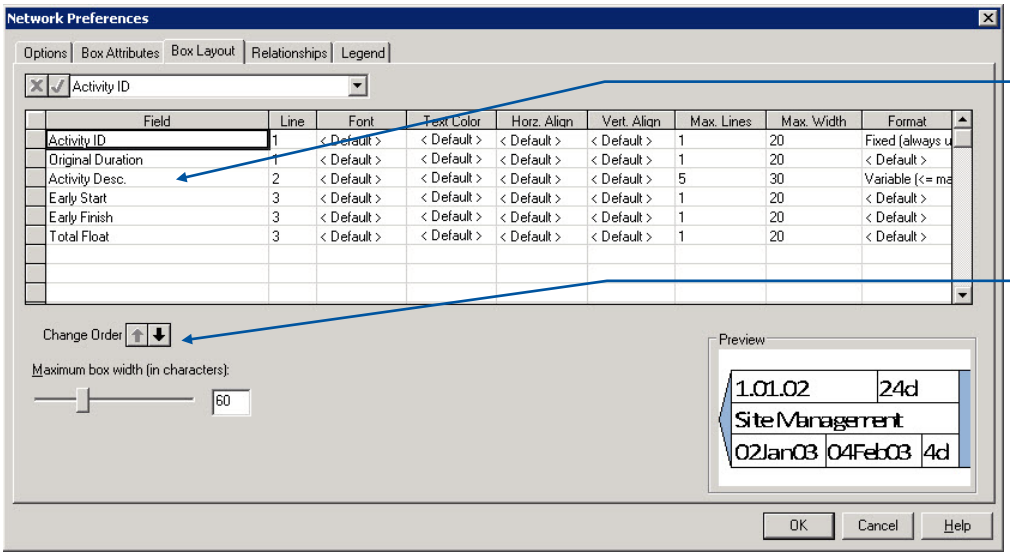
- The initial definition of the activity is based on the critical criterion, the first criterion that is satisfied by the activity. Only the left fill color is defined for this criterion — all other settings are based on criteria higher in the list.
- The left and right end styles are based on the definition for activities with an OBS code of Dept_2, the next criterion satisfied by the activity.

- The In-progress criterion results in the left diagonal being drawn in the activity box.
- The settings for text color, right fill color, and right diagonal are based on the standard set of attributes since these settings are not defined for any criterion applying to the activity.

Custom Box Text

You can customize a network view to display any activity field (including calculated fields) in the activity boxes. Boxes can contain as many as 50 different data items displayed on a maximum of 20 lines. You can also display activity fields as graphic progress bars or display blank lines in a box.

To customize the layout of activity boxes, display the **Box Layout** tab of the **Network Preferences** dialog box:



The preview area of the dialog box displays the current definition of the activity boxes.

> **Note:** This representation may differ slightly in size and aspect ratio from the actual display of boxes in the view.

Notice that Open Plan displays data items in the dialog box in order of line number. When you add, delete, or change data items, you must exit and return to the dialog box to display the items in the correct line sequence.

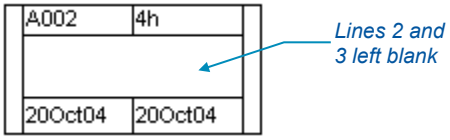
Open Plan displays items sharing the same line from left to right based on the order in which the items appear in the list. To move an item to a different position in the list, use the **Change Order** buttons.

You can customize the layout of the activity box with the following settings.

- **Field** — You can select any activity field for display.
- **Line** — For each field appearing in the activity box, you must assign a line number. Activity boxes can contain up to 20 lines, and each line can display one or more fields. If you assign multiple fields to a single line, the order in which the fields are listed in the dialog box determines how Open Plan displays the fields in the box. To display a blank line in the activity box, do not enter settings for that line. For example, to display activity boxes with a double-high blank line, use the following settings:

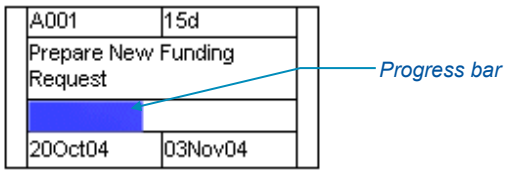
Field	Line	Format
Activity Identifier	1	1 line
Duration	1	1 line
Early Start Date	4	1 line
Early Finish Date	4	1 line

This results in the following activity box:



- **Font** — You can select the font type for the standard or calculated fields in the activity file.
- **Text Color** — You can specify the text color for the standard or calculated fields in the activity file.
- **Horz. Align** — You can specify the horizontal alignment of the text. The options available for this column are the following: Center, Left, and Right.
- **Vert. Align** — You can specify the vertical alignment of the text. The options available for this column are the following: Bottom, Center, and Top.
- **Max. Lines** — You can specify the maximum lines in the box on which the text is to be displayed.
- **Max Width** — You can specify the maximum width of the box on which the text is to be displayed.
- **Format** — You can specify the number of lines that the field may use to display data.

The option to show an activity field as a progress bar allows you to define activity boxes that can represent some types of activity information (for example, schedule progress) graphically:



To be displayed as a progress bar within the activity box, a field should represent a percentage value such as percent complete or physical percent complete. While a single activity box can contain multiple progress bars, you can display only one progress bar per line.

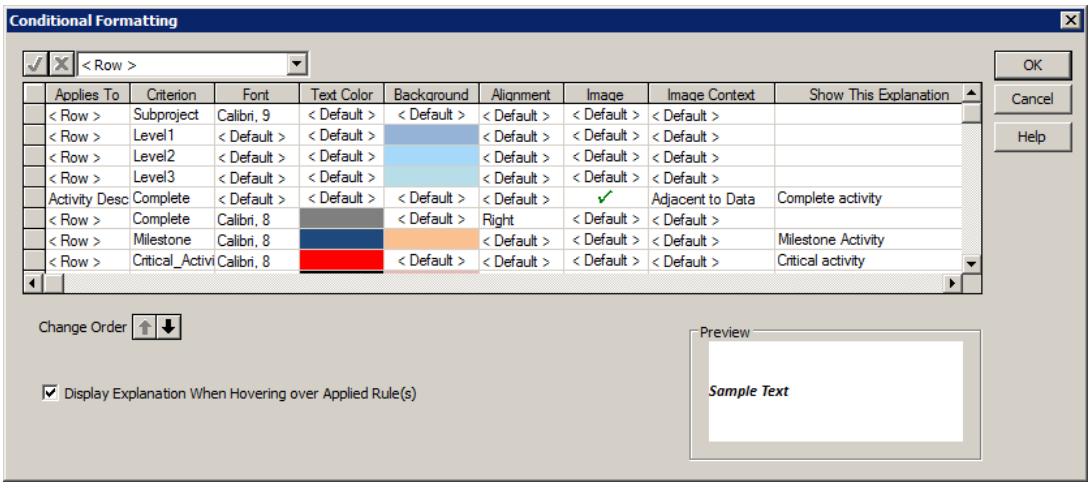
To customize the box layout, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click an empty area within the view and click **Preferences** on the context menu.
2. On the **Network Preferences** dialog box click the **Box Layout** tab.

3. Enter the information for each data item.
   - To change the order of the items, click any cell in a row and click one of the **Change Order** buttons.
   - To delete an item from the list, click any cell in the row and press the **Delete** key.
4. When your changes are complete, click **OK** to return to the view.

Conditional Formatting

You can customize the display of the activity boxes for the Network View by using the **Conditional Formatting** dialog box:



With the **Conditional Formatting** dialog box, you can override these default settings by specifying the conditions under which the properties of the activity boxes will change. For example, you may want an activity box that exceeds a particular criterion or filter to appear in bold red text, show a symbol, or display in a different background.

> **Note:** This dialog box shares similar functionalities with the **Set Column/Row Attributes** dialog box. Although they differ in the fact that you are not able to assign formats to rows in the **Applies To** column.

The following columns in the grid are used to define each formatting rule:

- **Applies To** — select any of the standard or calculated fields.
- **Criterion** — select one of the following:
  - A calculated field that evaluates to a Boolean (True/False) value or a filter
  - An on-the-fly expression that evaluates to a Boolean
  - A blank, indicating that the rule will be unconditionally applied
- **Font** — select a font, font style, and/or size.
- **Text Color** — select from the color palette.
- **Background** — select from the color palette

- **Alignment** — select either center, left or right.
- **Image** — select the appropriate symbol from either the drop-down list or the **Manage Symbols** dialog box.
- **Image Context** — select one of the following:
- **< None >** — Removes any image that may have been set in a rule on a higher row in the grid.
- **< Default >** — Retains the attributes of the prevailing rule on a higher row in the grid.
- **Centered in cell** — data is superimposed on the symbol which is centered.
- **Adjacent to data** — both data and symbol are displayed in the same cell.
- **Replaces data** — only symbol is displayed.
- **Show This Explanation** — specify the explanatory text you want to display when you hover your cursor over the row or cell. Use explanatory text to describe not only why a value is highlighted, but what the user should do about it.

To customize the display of the activity boxes for the Network View using the Conditional Formatting dialog box, complete the following steps:

1. Open the project in Network View.
2. Right-click the view and select **Conditional Formatting**.
3. The **Conditional Formatting** dialog box opens.
4. Select a field under the **Applies To** column.
5. Once you have selected a field to format, you may choose a criterion or any other attribute to modify the same field.
6. To make a formatting selection, click the drop-down list and/or the ellipsis box, if present.
7. When finished making selections, click **OK**.

Customizing Resource Histogram Views

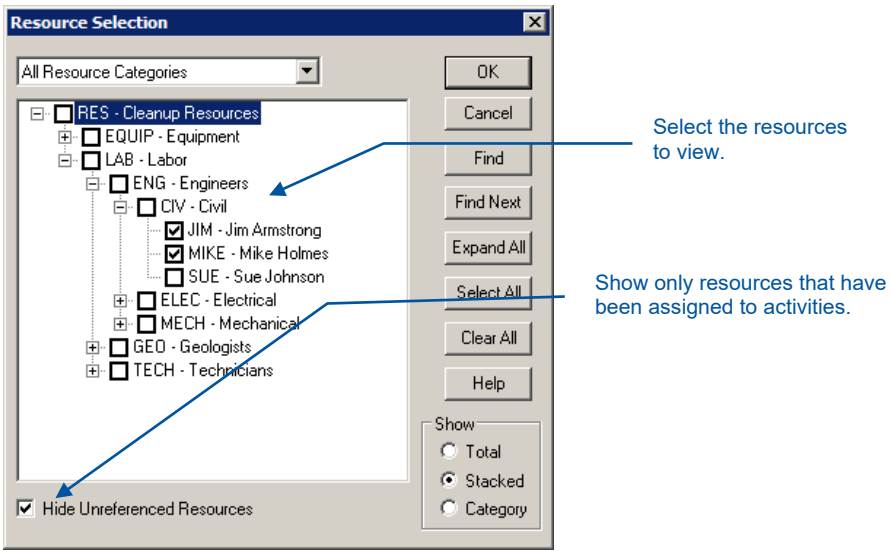
Three operations allow you to customize resource histogram views in Open Plan:

- Selecting resources
- Defining histogram display options
- Customizing the date scale

Selecting Resources

Before you can display data in a resource histogram, you must select the resource(s) to include in the view. When you click **Select Resource** on the **View** tab (or select **Select Resource** from the context menu), Open Plan displays the **Resource Selection** dialog box:





The list field at the top of the dialog box allows you can to filter the display of resources by category:

- All Resource Categories
- Labor
- Material
- Other Direct Costs
- Subcontract

A resource's category is defined on the **General tab** of the **Resource Details** dialog box.

The **Expand All** button will expand the entire hierarchy for viewing.

To display a resource from the list, simply select the check box next to the resource name. To select all the resources, click the **Select All** button.

You can also choose to display resource pools in the histogram. Keep in mind, however, that all availabilities, requirements, and usage displayed for a pool include both the pool and a roll-up of the child resources belonging to the pool. Clearing the **Hide Unreferenced Resources** setting displays the entire resource use file hierarchy, which makes the resource pools available for selection.

> **Note:** When selecting resources to display, be sure that all selected resources use the same time unit. For example, you should not mix resources measured in person-hours and resources measured in person-days or person-weeks in the same bar.

The **Show** options determine how Open Plan displays bars or curves representing more than one resource. There are three choices:

- **Total** — This option displays the resource bars or curves for multiple resources as a total quantity. You cannot discern the amount of resource requirement or usage within a time frame for any specific resource.
- **Stacked** — This option displays the resources bars for multiple resources in a stacked format. Each selected resource is displayed in a separate color. In the case of cumulative values, Open Plan displays separate S-curves for each resource.

- **Category** — This option displays the resource bars for each resource category. In the case of cumulative values, Open Plan displays separate S-curves for each resource category.

If you select both a pool and any other pool or individual resource, the histogram displays each pool as a total histogram bar or S-curve.

To select resources, complete the following steps:

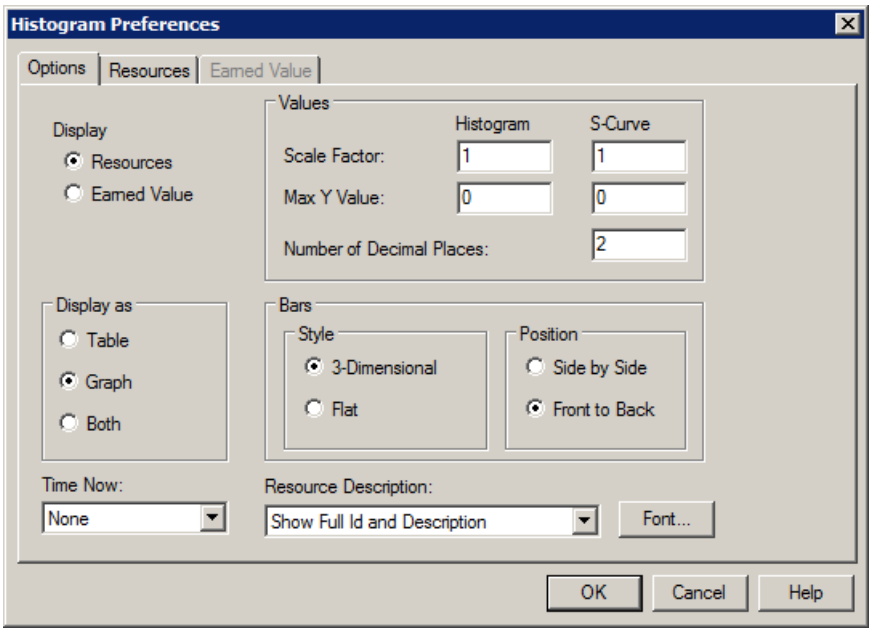
1. Take one of the following actions:
   - On the **View** tab, click **Select Resource**.
   - Right-click anywhere in the view, and click **Select Resource** on the context menu.
2. Select one or more resources from the list displayed in the dialog box.
3. To select all the resources, click **Select All**.
4. Indicate if the bars and curves should be total, stacked, or category.
5. When the information is complete, click **OK** to return to the view.

If you have applied a filter to the view, a resource histogram represents only those activities that satisfy the filter.

> **Attention:** For information about filters, see Histogram Views.

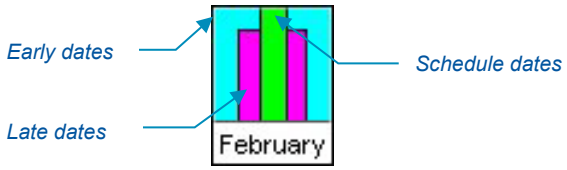
Resource Histogram Display Options

To define the appearance of bars in a resource histogram view, click **Preferences** on the **View** tab or right-click within the histogram view and click **Preferences** on the context menu to display the **Options** tab of the **Histogram Preferences** dialog box:



With this dialog box, you can indicate whether you want to display resource information or earned value information and if you want to display a graph only, a table only, or a combination of the two. If you display a table, you can indicate how many decimal places should be displayed.

You also have the option of specifying the bar style (3-dimensional or flat). If multiple bars are displayed, you can specify their display as either side-by-side or front-to-back. If you choose to display flat bars front-to-back, Open Plan varies the width of the bars to distinguish between the different types of information being displayed, for example:



You can also set the following options:

- **Scale Factor** — This setting determines how Open Plan scales the values appearing in the histogram bar or S-curves. For no scaling, set the scale factor to 1.
- **Max Y Value** — This setting determines the highest value displayed on the vertical axis of the histogram bars or S-curves. To allow Open Plan to set a maximum value based on the data, leave this setting at 0.
- **Number of Decimal Places** — This field controls the number of decimal places that Open Plan should display on the histogram. You can specify any value between 0 and 5.
- **Time Now** — You can select from the following options:
- **None** — If you select this option, Open Plan does not display the Time Now line on the histogram.
- **Line** — If you select this option, Open Plan identifies Time Now using a vertical line. There is no identifying text beside the Time Now line.
- **Line with Text** — If you select this option, Open Plan identifies Time Now using a vertical line. In addition, the words "Time Now" are displayed next to the Time Now line.

  If the histogram is displayed as a pane in the barchart view, Open Plan also features the following additional options:
- **Conditional Line** — If you select this option, Open Plan identifies Time Now using a vertical line only if the barchart identifies Time Now with a vertical line.

  If the barchart does not identify Time Now in this way, the histogram does not identify Time Now.
- **Conditional Text** — If you select this option, Open Plan identifies Time Now using both a vertical line and the text specified as the label in the Barchart Preferences dialog box only if the barchart identifies Time Now with a vertical line and a label.

  If the barchart does not identify Time Now in this way, the histogram displays Time Now with only a vertical line.
- **Resource Description** — You can select from the following options:
- **Show Full ID** — If you select this option, Open Plan identifies each resource by the full resource ID. For example, the resource may be identified as:

  TEAM.LAB.JOAN
- **Show Local ID** — If you select this option, Open Plan identifies each resource by the local portion of the resource ID. For example, the resource may be identified as:

  JOAN

  Where:

  JOAN is the local portion of the TEAM.LAB.JOAN resource ID.

- **Show Description** — If you select this option, Open Plan identifies each resource only by its description. For example, the resource may be identified as:

  JOAN SHERWOOD

  Where:

  JOAN SHERWOOD is the resource description.

- **Show Full ID and Description** — If you select this option, Open Plan identifies each resource by both the full resource ID and the description. For example, the resource may be identified as:

  TEAM.LAB.JOAN JOAN SHERWOOD

  Where:

  TEAM.LAB.JOAN is the full resource ID.

  JOAN SHERWOOD is the resource description.

- **Show Local ID and Description** — If you select this option, Open Plan identifies each resource by both the local portion of the resource ID and the description. For example, the resource may be identified as:

  JOAN JOAN SHERWOOD

  Where:

  JOAN is the local portion of the resource ID.

  JOAN SHERWOOD is the resource description.

- **Font** — Clicking this button displays the Font dialog box that you can use to customize the font used in the histogram view.

To define histogram display options, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Preferences** from the **Histogram** group (in a barchart view) or **Preferences** (in a histogram view).
   - Right-click anywhere in the view, and click **Preferences** on the context menu.
2. On the **Options** tab of the **Histogram Preferences** dialog box, enter the information for the display options.
3. When the information is complete, click **OK** to return to the view.

Customizing the Date Scale

You can customize the date scale appearing in a stand-alone resource histogram view using the same procedures that you use to customize the date scale in a barchart view. (In fact, when a resource histogram appears in a barchart view, the barchart date scale controls the histogram portion of the view as well.)

> **Attention:** For information about customizing the date scale, see Customizing the Date Scale.

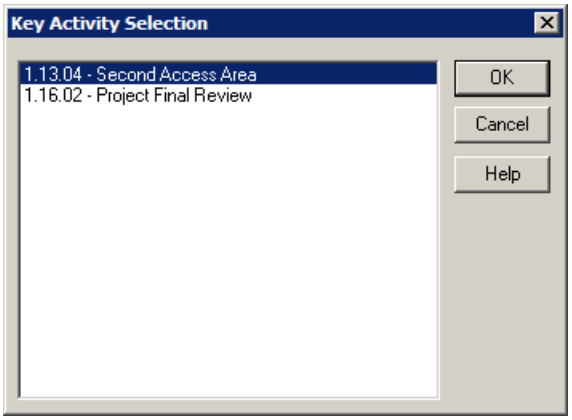
Customizing Risk Histogram Views

Three operations allow you to customize risk histogram views in Open Plan:

- Selecting a key activity
- Defining histogram display options
- Customizing the date scale

Selecting a Key Activity

Open Plan allows you to select the activity that appears in the view using the following dialog box:

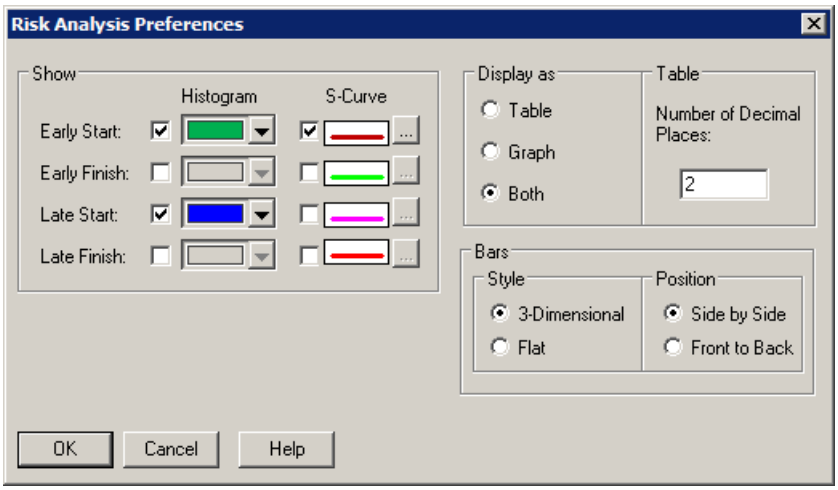


To select a key activity, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Select Activity**.
   - Right-click anywhere in the view, and click **Select Activity** on the context menu.
2. Select an activity from the list displayed in the dialog box, and click **OK** to return to the view.

Risk Histogram Display Options

To define the appearance of a risk histogram view, display the **Risk Analysis Preferences** dialog box:



With this dialog box, you can indicate which dates (**Early Start**, **Early Finish**, **Late Start**, or **Late Finish**) to display in the view and whether you want to display histogram bars or cumulative S-curves. You can also indicate whether you want to display a graph only, a table only, or a combination of the two. If you display a table, you can indicate how many decimal places should be displayed.

If you are displaying histogram bars, you have the option of specifying the bar style (3-dimensional or flat). If multiple bars are displayed, you can specify their display as either side-by-side or front-to-back. If you choose to display flat bars front-to-back, Open Plan varies the width of the bars to distinguish between the different types of information being displayed.

To define risk histogram display options, complete the following steps:

1. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click anywhere in the view, and click **Preferences** on the context menu.
2. Enter the information for the display options.
3. Click **OK** to return to the view.

Customizing the Date Scale

As in the case of a resource histogram, you can customize the date scale appearing in a risk histogram view using the same procedures that you use to customize the date scale in a barchart view.

> **Attention:** For information about customizing a histogram date scale, see Customizing the Date Scale.

Customizing the Web Publisher

Each time you use the Open Plan Web Publisher to set up an HTML file that points to .pdf versions of project views, Open Plan creates a small text file, <*projectname*>.wbx, that contains the parameters for that operation. These .wbx files store information such as the project name and description as well as a listing of the contents of each batch defined for the project and the output options selected by the user.

> **Note:** Do not change the default printer during an Open Plan Web Publisher session since this may cause Open Plan to output the reports to the wrong device.

Customizing Headers and Footers

It is possible to specify custom headers and footers for the resulting HTML file by adding [Header] and [Footer] sections to the .wbx file. You can define a custom header or footer by setting one or more options to lines of standard HTML code as in the following example:

```
[Header]
H1=<CENTER>This is a custom header</CENTER>
H2=<CENTER>Second line of a custom header</CENTER>

[Footer]
F1=<CENTER>This is a custom footer</CENTER>
```

> **Note:** The option names used in this example (H1, H2, and F1) are for demonstration purposes only and can, in fact, be any text string.

It is also possible to define default headers and footers for all HTML files created by your installation of Open Plan. The contents of all project-specific .wbx files can be based on the settings defined in a default .wbx file. This file must be named Opweb.wbx and must be located in the System folder of your Open Plan installation. If you add [Header] and [Footer] sections to this file, Open Plan automatically adds the sections to all new .wbx files created thereafter.

The following is a typical example of Opweb.wbx:

```
[DefaultBatch]
CurrentDevice=Acrobat PDFWriter
MakeHTML=1
DestinationDir=C:\OUTPUT
WebSite=/
View001=Progress Network|C:\Opp\System\Progflow.tpl|501.GIF
View002=Activity Network|C:\Opp\System\Flowvw.tpl|501.GIF

[Header]
H1=<CENTER>ABC Industries</CENTER>

[Footer]
F1=<CENTER>For Internal Distribution Only</CENTER>
```

If this file is present in the Open Plan system folder, all new batches will default to these output options and will automatically include the two views specified. In addition, a header and footer will appear in the final HTML file.

When you are launching the Open Plan Web Publisher from outside Open Plan, you can include a command line parameter that specifies a group of .wbx files and report batches to output as follows:

```
OPWPUB32.exe @<textfile>
```

The *<textfile>* parameter, in turn, refers to a file containing one or more statements that specify .wbx files and batch names using the following syntax:

```
/W=<.wbxfile> /B=<batchname>
```

For example, you might set up a command statement as follows:

```
OPWPUB32.exe @c:\Opp\Data\Mylist.txt
```

The file Mylist.txt might, in turn, consist of the following statements:

```
/W=C:\Program Files\Deltek\Open Plan Professional\USERS\Proj1.wbx /B=Batch1
/W=C:\Program Files\Deltek\Open Plan Professional\USERS\Proj1.wbx /B=Batch2
/W=C:\Program Files\Deltek\Open Plan Professional\USERS\Proj2.wbx /B=Batch1
/W=C:\Program Files\Deltek\Open Plan Professional\USERS\Proj2.wbx /B=Batch2
```

Invoking the Open Plan Web Publisher in this way would result in the four batches being output automatically.

Utilities

Open Plan Configuration Files

Settings can be configured to individual user preferences and stored in a Config.dat file saved in a separate folder. All users accessing the same program executable are using the same Config.dat file that is stored in the program folder.

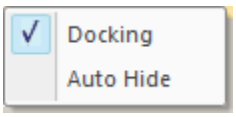
Open Plan uses other .dat files that allow you to customize settings and view stored information. Tab options can be added to the AddIns.dat file to instruct Open Plan to launch external applications. A complete list of available data sources can be viewed in the Datasources.dat file. The Briefcase.dat file lists the data sources that are briefcased.

> **Note:** On the following pages, **Location** indicates where Open Plan expects to find the specific setting.

Working with Dockable Web Windows

The Task Form and Resource Selector Add-ins, which are Open Plan Web Windows, are dockable. When the web window is docked, you can click and drag the title bar to undock the window or move the window to a new docking position.

- When you want to move the web window without snapping it into place (floating mode), you can press the Ctrl key while dragging the window.
- While the window is floating, you can return the window to its previous docked position by double-clicking the title bar.
- When you right-click the title bar of the window, the following context menu appears:



You can choose to dock or auto hide the web window. When auto hidden, you can re-launch the window by clicking the window tab located at the bottom of the screen. You also have the option to temporarily display the window by placing your mouse cursor over the same window tab.

[Open Plan Professional]

This section contains user and license information collected during the installation of Open Plan.

License

Location: Config.dat

Type: String

Description: Application License Key

SerialNumber

Location: Config.dat

Type: String

Description: Open Plan serial number

[System]

The **[System]** section contains configuration information that allows Open Plan to be tailored to the requirements of your runtime environment.

Company

Location: Config.dat

Type: String

Description: Company name.

DataSource

Location: Config.dat

Type: String

Description: Specifies the name of a default Open Plan Data Source that will be used the first time a new user logs into Open Plan.

DefaultLanguage

Location: Config.dat

Type: String

Description: Filename of .dll containing Open Plan localized resources of the default language choice.

DataSources

Location: Config.dat

Type: String

Description: Path and name of the file containing Open Plan data source configuration tables.

Note: DataSources.dat is present in Config.dat only if DataSources.dat is stored in a non-default location.

LogDir

Location: Config.dat

Type: String

Description: Path and name where log files are stored.

MaxLogin Tries

Location: Config.dat

Type: Number

Description: Specifies the maximum number of times a user can try to log in to Open Plan when an invalid User Name or Password is entered.

Setting the value of the MaxLoginTries option to 0 in combination with enabling the WindowsAuthentication option will suppress the display of the Open Plan login dialog under any circumstances.

The default value is 3.

Values: 0, 1, 2, 3

UserDir

Location: Config.dat

Type: String

Description: Path and name of the file that creates a working directory for each user.

[SSCE]

The **[SSCE]** section stores information that allows you to customize the dictionary Open Plan uses for spell checking.

MainLexFiles

Location: Config.dat

Type: String

Description: Controls the default dictionary

MainLexPath

Location: Config.dat

Type: String

Description: Path to folder containing the default dictionary

UserLexFiles

Location: Windows Registry

Type: String

Description: Allows you to customize the default dictionary to your specifications

UserLexPath

Location: Windows Registry

Type: String

Description: Path to the folder containing the customized dictionary

Windows Registry Entries

The **Windows Registry** contains configuration information that allows Open Plan to be tailored to the individual user. The setup program places settings for this section in the **HKEY_CURRENT_USER\Software\Deltek\Open Plan Professional\3.2** section of the **Windows Registry**.

CurrentLanguage

Location: Windows Registry

Type: String

Description: Filename of the Open Plan resource .dll that contains the language resources of the currently selected language choice for the Open Plan user interface.

Note: This filename internally defaults to the **DefaultLanguage** setting in the **[System]** section of the Config.dat file.

Version

Location: Windows Registry

Type: String

Description: Contains the current version of the Open Plan executable file

DataSource

Location: Windows Registry

Type: String

Description: Name of the Open Plan data source the user is currently using

Sample Tools

Open Plan ships with the following programmed tools. The items in **bold** display on the Add-Ins tab by default. The remaining items can be enabled by uncommenting their entries in the Add-Ins.dat file.

- **Task Form**
- **Renumber Activities**
- **DCMA 12 VBS**
- **Options Barchart**
- **Resource Selector**
- **Resource Utilization Chart**
- **Resource Management Wall Chart**
- **Date & Status Report (XML)**
- **Predecessor & Successor Report (XML)**
- **Assignment Barchart**
- **Resource/Activity Report (XML)**
- **Schedule Comparer**
- **Edit AddIns.dat**
- **Reload Add-Ins**
- Sample Dialog
- Document Launcher
- Document Library Objects
- Deltek Web site
- Cobra Link
- Cobra Sync
- Parts Database
- Cobra Cost Detail

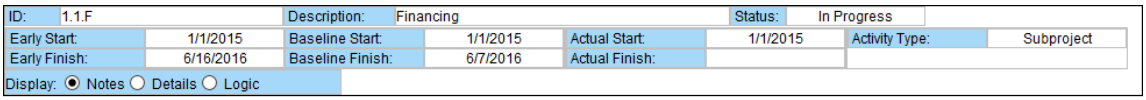
Each of these tools can be modified to create tools specific to user needs. To edit some of the tools, you will need Microsoft Visual Basic.

Task Form

The Task Form add-in is an Open Plan Web Window that docks at the bottom of the view. It displays data about the activity selected in the view. There is a header for basic activity information a radio button to select what additional details are displayed in the bottom of the screen.

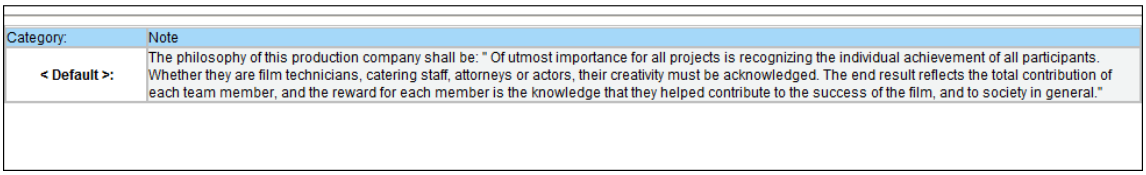
The Task Form tool is available when a project view is in focus and an activity is selected.

Notes display all populated note categories for the selected activity.



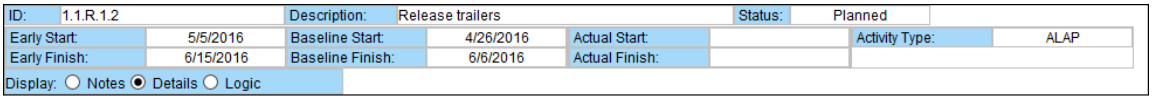
ID:	1.1.F	Description:	Financing	Status:	In Progress		
Early Start:	1/1/2015	Baseline Start:	1/1/2015	Actual Start:	1/1/2015	Activity Type:	Subproject
Early Finish:	6/16/2016	Baseline Finish:	6/7/2016	Actual Finish:			

Display: ◉ Notes ○ Details ○ Logic



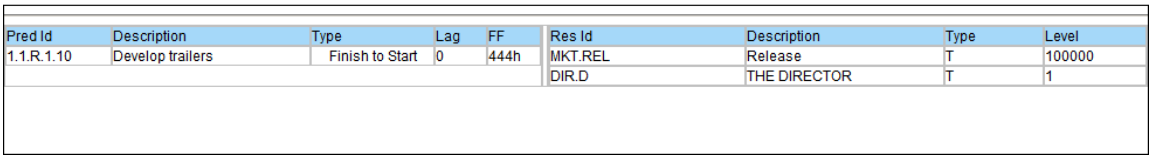
Category:	Note
< Default >:	The philosophy of this production company shall be: " Of utmost importance for all projects is recognizing the individual achievement of all participants. Whether they are film technicians, catering staff, attorneys or actors, their creativity must be acknowledged. The end result reflects the total contribution of each team member, and the reward for each member is the knowledge that they helped contribute to the success of the film, and to society in general."

Details display predecessors and resources for the selected activity.



ID:	1.1.R.1.2	Description:	Release trailers	Status:	Planned		
Early Start:	5/5/2016	Baseline Start:	4/26/2016	Actual Start:		Activity Type:	ALAP
Early Finish:	6/15/2016	Baseline Finish:	6/6/2016	Actual Finish:			

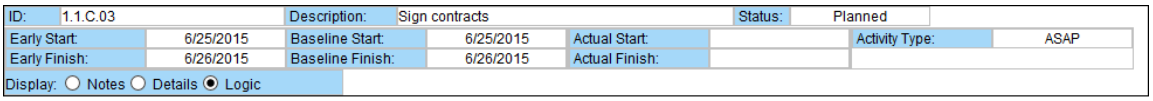
Display: ○ Notes ◉ Details ○ Logic



Pred Id	Description	Type	Lag	FF
1.1.R.1.10	Develop trailers	Finish to Start	0	444h

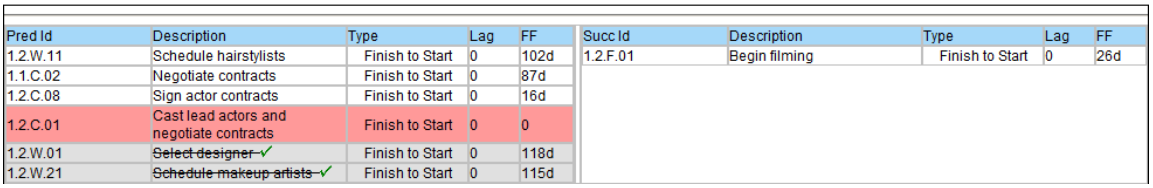
Res Id	Description	Type	Level
MKT.REL	Release	T	100000
DIR.D	THE DIRECTOR	T	1

Logic displays predecessors and successors for the selected activity.



ID:	1.1.C.03	Description:	Sign contracts	Status:	Planned		
Early Start:	6/25/2015	Baseline Start:	6/25/2015	Actual Start:		Activity Type:	ASAP
Early Finish:	6/26/2015	Baseline Finish:	6/26/2015	Actual Finish:			

Display: ○ Notes ○ Details ◉ Logic



Pred Id	Description	Type	Lag	FF
1.2.W.11	Schedule hairstylists	Finish to Start	0	102d
1.1.C.02	Negotiate contracts	Finish to Start	0	87d
1.2.C.08	Sign actor contracts	Finish to Start	0	16d
1.2.C.01	Cast lead actors and negotiate contracts	Finish to Start	0	0
1.2.W.01	~~Select designer~~ ✓	Finish to Start	0	118d
1.2.W.21	~~Schedule makeup artists~~ ✓	Finish to Start	0	115d

Succ Id	Description	Type	Lag	FF
1.2.F.01	Begin filming	Finish to Start	0	26d

In this view controlling relationships are highlighted with a red background. Completed relationships are shown with a gray background and strikethrough text.

Resource Summary displays assignments for the selected activity and the project summary usage for those resources found on other project.

ID:	1.1.C.02	Description:	Negotiate contracts	Status:	In Progress		
Early Start:	1/15/2016	Baseline Start:	1/13/2016	Actual Start:	1/15/2016	Activity Type:	ASAP
Early Finish:	2/25/2016	Baseline Finish:	2/22/2016	Actual Finish:			

Display: ○ Notes ○ Details ○ Logic ◉ Resource Summary

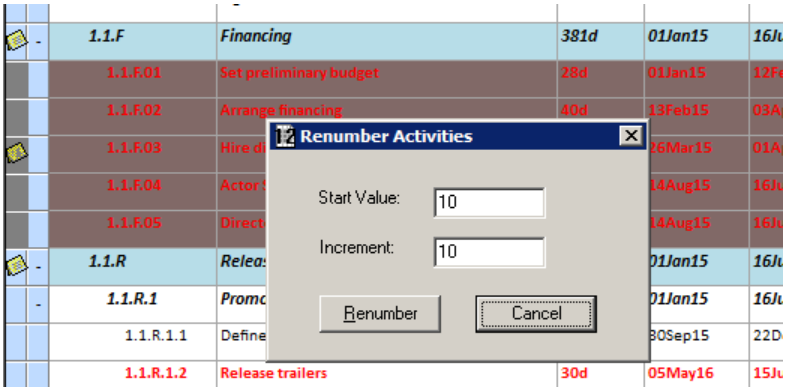
Activity Assignments			Other Project Assignments			
Assignment:	Type	Level	**Project**	Start	Finish	Qty
LAW.L : Staff Attorneys	T	120.00 Hours	MOVIE	6/6/2016	11/29/2016	1,611.00 Hours
LAW.P : Paralegals	T	200.00 Hours	MOVIE	6/6/2016	11/4/2016	885.00 Hours

This can help find where critical resources are being used. You may need to negotiate for these resources with other projects.

String: Tool19=System:Task Form;%SYSTEMDIR%\Sample Tools\OPXMLVIEW\TaskForm\oppdataFrames.htm?DOCKSIDE=BOTTOM&DOCKLENGTH=300 %P %A %V %D %I %T

Renumber Activities

The Renumber Activities tool is available when an activity view is in focus and 2 or more activities are selected. It allows you to renumber the local portion of the Activity ID for selected activities using a start value and increment. It will not renumber activities that have different parents or are at different levels. It is intended for contiguous groups of activities that all have the same parent task.



When selected, you are prompted for the start value and increment. The tool will renumber the activities selected above so that they would:

Activity ID Before	Activity ID Before
1.1.F.01	1.1.F.10
1.1.F.02	1.1.F.20
1.1.F.03	1.1.F.30
1.1.F.04	1.1.F.40
1.1.F.05	1.1.F.50

String: Tool18=System:Renumber Activities;%SYSTEMDIR%\Sample Tools\vbapps\Renumber\actrenumber.exe %P

DCMA 12 VBS

The DCMA 12 VBS tool runs a VBScript file. DCMA 12 has you select an activity on the critical path. When you run the tool it adds 600 days to the activity duration and runs TA. It expects the project finish to go out by 600d if the critical path is complete. If the end does not go out about 600d then the test fails.

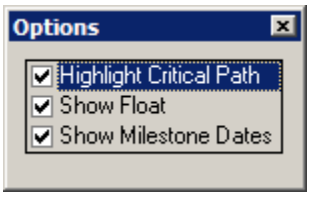
The tool restores the duration on the selected activity and tuns TA again to reset the dates.

Options Barchart

The Options Barchart tool uses the Configurable Options functionality to configure the appearance of the Options Barchart view on the fly.

> **Note:** For more information on the **Configurable Options** functionality, see Configurable Options.

When the tool is run for a selected open project, Open Plan displays the **Options Barchart** view and then displays the following check box on top of the view:



By selecting (or clearing) these check boxes, you can toggle the display of the Critical Path, Float, and Milestone Dates.

The following string enables this tool:

String: Tool11= System:Options Barchart;%SYSTEMDIR%\Sample Tools\vbapps\options\options.exe "_Option_" "OptionsBarchart" %P

where

- _Option_ identifies the calculated fields used by the Options Barchart tool.

  The names of the three calculated fields all begin with "_Option_" (for example, _Option_Highlight_Critical_Path). When displayed in the **Options** check box, the "_Option_" and the remaining underscores are removed.

- OptionsBarchart identifies the name of the activity barchart that is displayed.

The **Configurable Options** sample application allows you to create a custom Add-Ins tool that configures the appearance of an activity barchart view on the fly. This works by changing the value of specific calculated fields. The barchart attributes are defined to use filters that respond to changes in these calculated fields, thus changing the appearance of the barchart view.

To use this functionality, add the following line to the AddIns.dat file:

```
Tool#=Label;Drive:\Path\Options.exe "_Optiontag_" "Viewname" %P
```

Where

- # is a valid Tool number.
- Label is the name that is displayed on the Add-Ins tab.
- Drive:\Path\ is the location of the Options.exe file.
- _Optiontag_ is the string that identifies a calculated field as a configurable option.

- Viewname is the name of the activity barchart view that is to be displayed.

Configurable Options

To illustrate the process for creating a Configurable Options custom tool, assume that we want to create an option to turn the display of non-critical activities for the Early Dates bar on and off for the default activity barchart view BARVW. The **_Optiontag_** we will use is "_Sample_."

> **Note:** On the **Bar Sets Preferences** dialog box for BARVW, notice that the **Non_Critical** filter is already associated with the **Early Dates** bar type. Instead of creating a filter, we will modify a copy of this filter.

To set up a configurable options custom tool, complete the following steps:

1. Create a new calculated field using a specific **_Optiontag_** (for example, _Sample_Show_Noncritical).
2. Create the filter that will respond to the changes in the value of the above calculated field.
3. Modify the bar set preferences for the activity barchart view that will use the custom tool.
4. Add the Tool string to the AddIns.dat file.

Each of these steps is described in detail below.

Step One

1. On the **Tools** tab, click **Calculated Fields**.
2. On the Calculated Fields dialog box, click **New**.
3. On the New Calculated Field dialog box, enter the following:
   - **Name:** _Sample_Show_Noncritical

     > **Note:** When the option is displayed in the **Options** check box, the **OptionTag** and the remaining underscores are removed.

   - **Applies to Table:** Activity
   - **Data Type:** Integer
4. Click **OK**.
5. Enter **1** for the expression, and click **OK**.
6. Click **Close**.

Step Two

1. On the **Tools** tab, click **Filters**.
2. Select the **Non_Critical** filter, and click **Copy**. This is the filter currently associated with the **Early Dates** bar type.
3. Change the filter name to **_Samp_Non_Critical**.

   > **Note:** You can name the filter anything you want.

4. On the row beneath the existing parameters, add the following;
   - **Logic:** And
   - **Field Name:** _Sample_Show_Noncritical
   - **Operator:** Equals
   - **Value 1:** 1
5. Click **OK** twice.

Step Three

1. On the **View** tab, click **Bar Sets**.
2. On the **Bar Sets** dialog box, select **BARVW_BRS**, and click **Edit**.
3. Select the **Criterion** field for **Early Dates**.
4. Using the drop-down list, select **_Samp_Non_Critical**.
5. Click **Ok** twice.

Step Four

1. On the **Add-Ins** tab, click **Edit AddIns.dat**.
2. When the AddIns.dat file displays, enter the following string at the end of the list:

   Tool#=Show Non-Critical;C:\Program Files\Deltek\Open Plan Professional\Sample Tools\VBAPPS\Options\options.exe "_Sample_" "BARVW" %P

   > **Note:** Substitute the **#** with the next **Tool** number available.

3. **Save** and **Close** the modified file.
4. On the **Add-Ins** tab, click **Reload Add-Ins** to initialize the new tool without having to restart Open Plan.

When you run the **Show Non-Critical** tool for a selected open project, Open Plan automatically displays the BARVW view with the **Options** check box on top showing the option you created:



Resource Selector

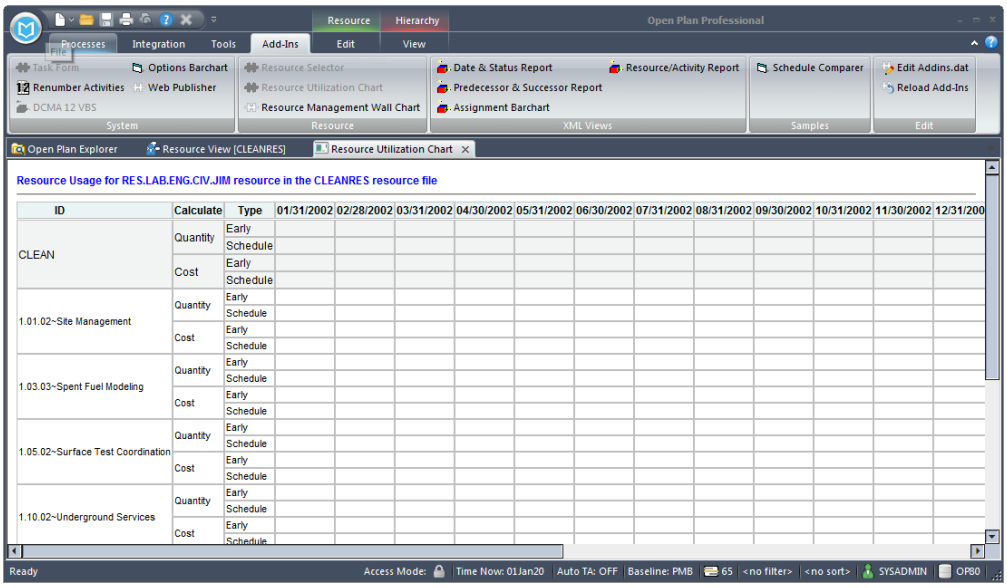
When launched, this Web Window builds a hierarchical tree of the resources assigned to the current project. You can then use the tree to filter a barchart view to show the activities using the selected resource. It also displays the selected resource in the histogram.

String: Tool17=Resource:Resource Selector;%SYSTEMDIR%\Sample Tools\OPXMLview\XMLResSel\ResSelFrame.htm %d %P %V

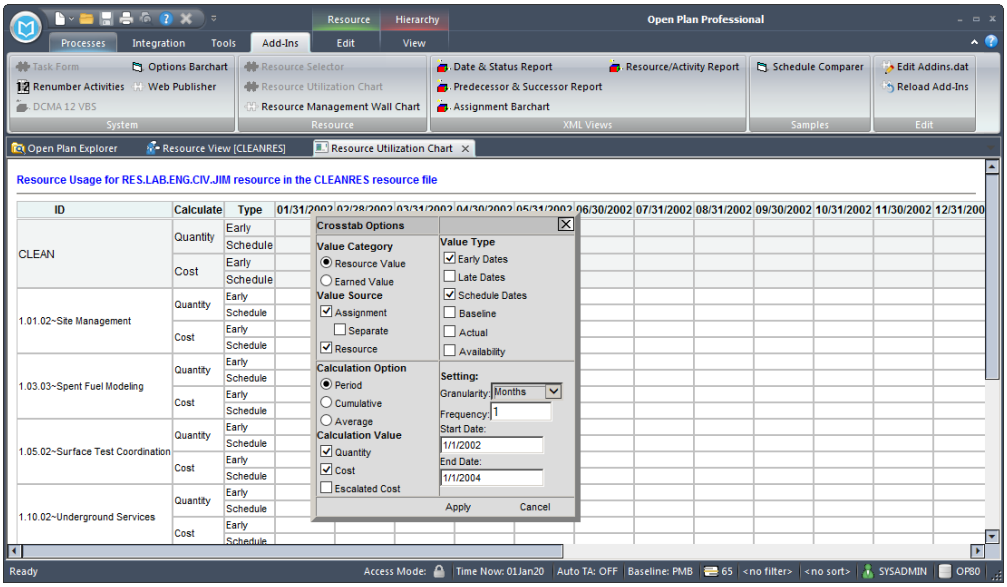
Resource Utilization Chart

The Resource Utilization Chart tool allows you to view crosstable information dynamically rather than in a text file with the Crosstab Export tool. After selecting a resource within a resource view, run the Resource

Utilization Chart tool. The following is displayed with a table containing resource usage information for the resource selected.



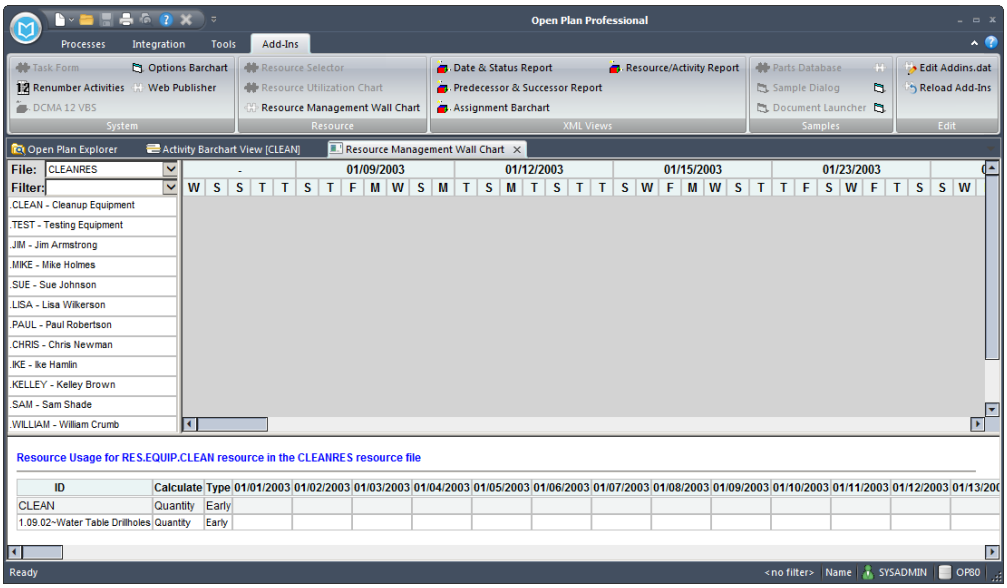
You can change the type of data displayed by right-clicking the screen and selecting your options.



String: Tool5=Resource Utilization Chart;%SYSTEMDIR%\Sample Tools\OPXMLview\crosstab\crosstab.htm %i %R %r

Resource Management Wall Chart

The Resource Management Wall Chart tool is similar to the Open Plan histogram view showing the loading of various resources. While a histogram view gives detail loading information, it is not easy to see the loading of multiple resources on a single page.



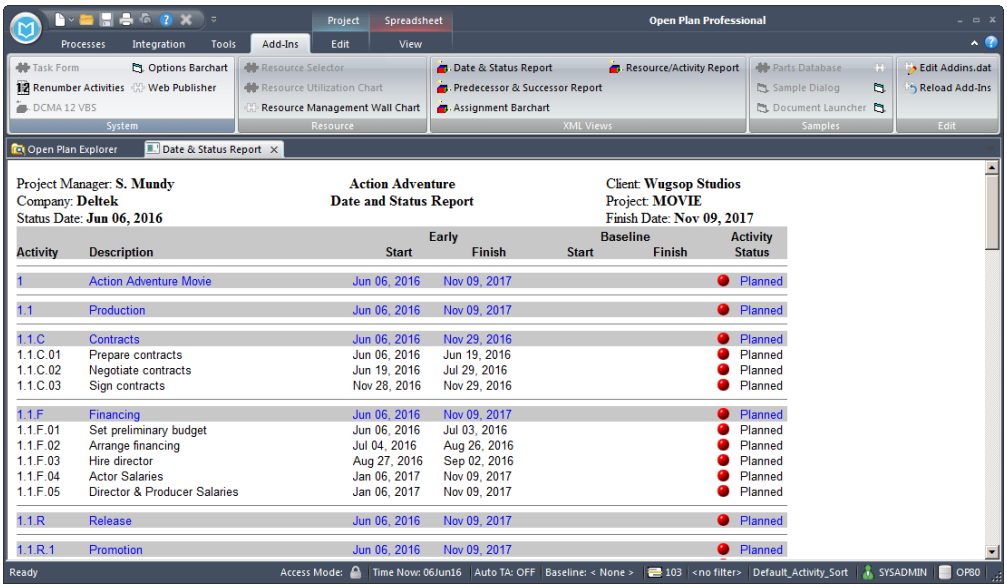
Each resource is listed on a single line on the chart. Against each resource are a series of colored blocks representing the following:

- **Grey** — non-working time (for that resource)
- **Blue** — the resource is used (somewhere between 0 and 100% of availability)
- **Red** — the resource is overloaded

String: Tool9=Resource:Resource Management Wall Chart;%SYSTEMDIR%\Sample Tools\OPXMLview\wallchart\wallchart.htm %i %R %T

Date & Status Report (XML)

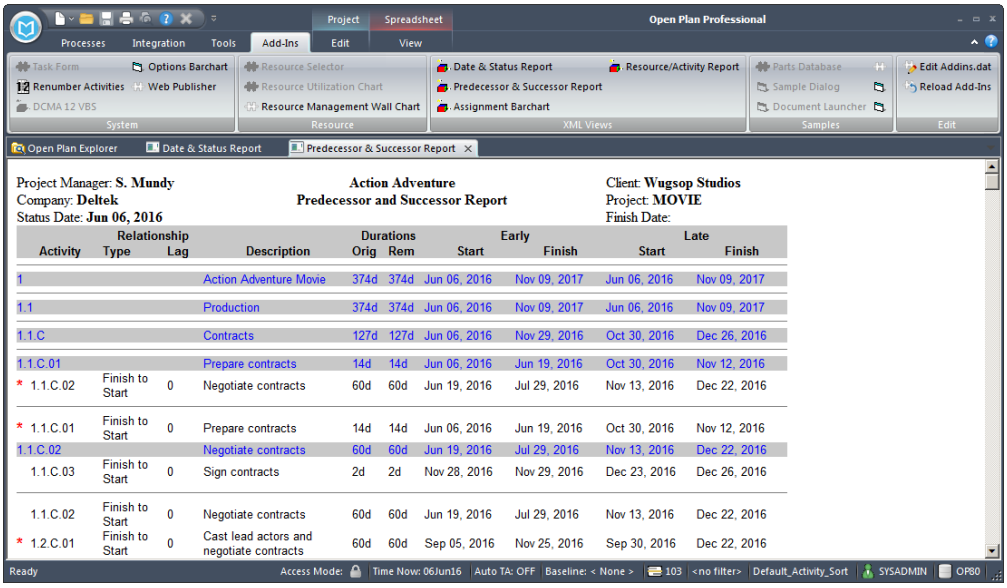
The Date & Status Report (XML) tool allows you to view dates and statuses of a project's activities in a summarized table. The information is displayed in XML format allowing more functionality and a user-friendly appearance. To use the tool, first select a project and then run the Date & Status Report (XML) tool.



String: Tool6=XML Views:Date && Status Report;wscript.exe "%SYSTEMDIR%\Sample Tools\vbscript\RunXMLReport.vbs" "%SYSTEMDIR%\Sample Tools\vbscript\actdata2.lst" /D %P

Predecessor & Successor Report (XML)

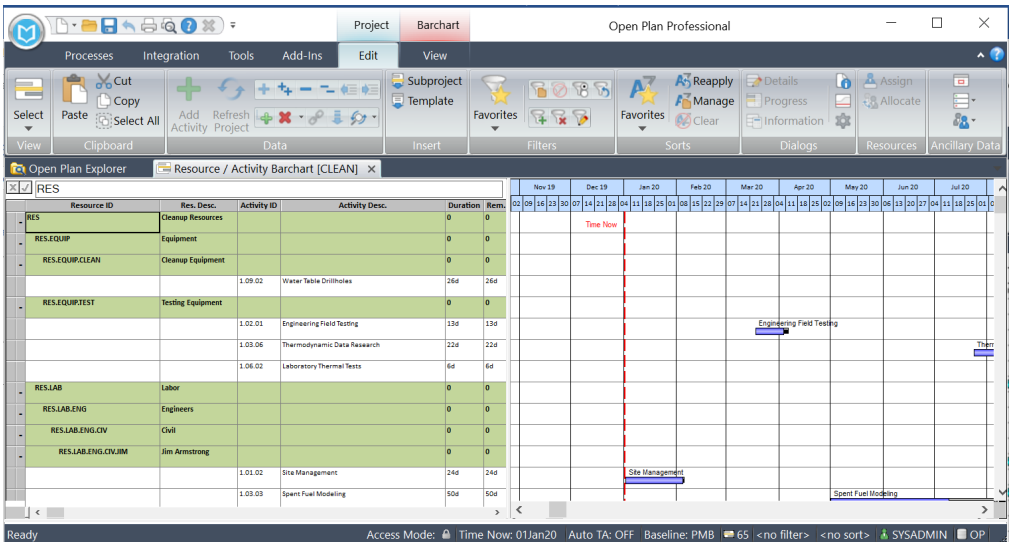
The Predecessor & Successor Report (XML) tool allows you to view a table showing the relationships between the activities within a project. Information in the table includes relationship lag, activity descriptions, durations, early start and finish dates, and late start and finish dates. To use the tool, first select a project and then run the Predecessor & Successor Report (XML) tool.



String: Tool7=XML Views:Predecessor && Successor Report;wscript.exe "%SYSTEMDIR%\Sample Tools\vbscript\RunXMLReport.vbs" "%SYSTEMDIR%\Sample Tools\vbscript\actdata1.lst" /D %P

Assignment Barchart

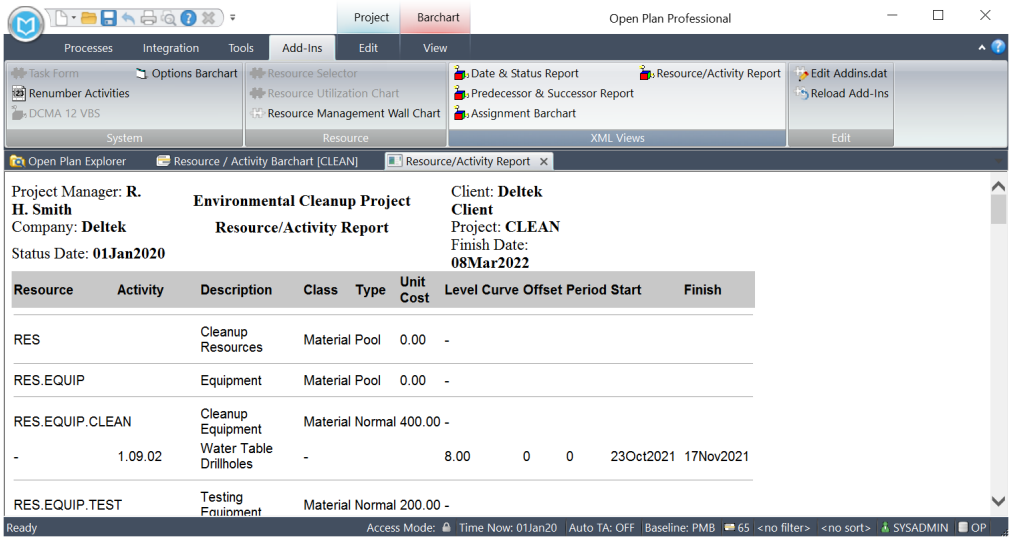
The Assignment Barchart tool displays a view showing each resource and its assigned activity ID along with bars indicating when the resource is being used. This allows users to see the allocation of resources to activities in one view.



String: Tool10=XML Views:Assignment Barchart;wscript.exe "%SYSTEMDIR%\Sample Tools\vbscript\asgbar.vbs" %P %R

Resource/Activity Report (XML)

The Resource/Activity Report (XML) tool provides you with a breakdown of a project's resources and the activities that use those resources. To use the tool, first select a project and then run the Resource/Activity Report (XML) tool.



String: Tool8=XML Views:Resource/Activity Report;wscript.exe "%SYSTEMDIR%\Sample Tools\vbscript\RunXMLReport.vbs" "%SYSTEMDIR%\Sample Tools\vbscript\rdsdata.lst" /D %P

Schedule Comparer

The Schedule Comparer allows you to compare two copies of a schedule or two baselines on a project. You can select which fields are compared.

Edit AddIns.dat

The Edit AddIns.Dat tool opens the AddIns.dat file in a text editor. This allows you to customize tools within the Add-Ins tab.

You can create new tools, edit existing tools, or delete tools within this text document. You need to save the changes to the file and click the Reload Add-Ins to view the changes you made.

You can add up to 30 custom tools to the Add-Ins tab in Open Plan by modifying the AddIns.dat files. For example, you can add an item to the Add-Ins tab that will start the WordPad program supplied by Windows. By making the required modification to the copy of the AddIns.dat located in the executable folder, all users of the same Open Plan installation can access this tab item.

You can also add an item to the Add-Ins tab that will start Outlook. Since your copy of Outlook may contain confidential information, you would make the required modifications to the AddIns.dat located in your Open Plan User folder.

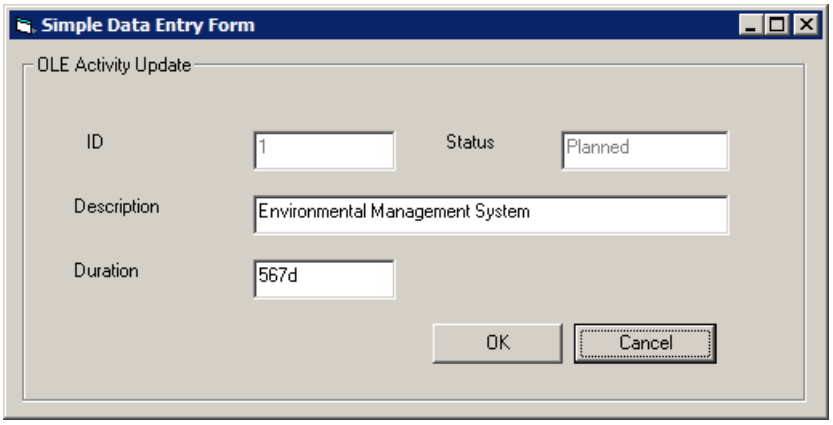
String: Tool14=Edit AddIns.Dat;notepad.exe %SYSTEMDIR%\addins.dat

Reload Add-Ins

Click the Reload Add-Ins tool to reload all of the add-ins from the AddIns.dat file.

Sample Dialog

The Sample Dialog tool allows you to create a customized dialog box within Open Plan. The source code provided for the Sample Dialog tool upon installation will open the Simple Data Entry Form dialog box.



String: Tool1=Samples:Sample Dialog;%SYSTEMDIR%\Sample Tools\VBAPPS\Dialog\simple.exe %P %A

You can modify the source code to create a dialog box for your specific needs by using Microsoft Visual Basic.

Document Launcher

With the Document Launcher tool, you can instruct Open Plan to open a document from its associated application. Given a selected activity in one of the spreadsheet, barchart, or network views, a document path or web address may be associated with the activity by specifying a link in the User Character 1 field.

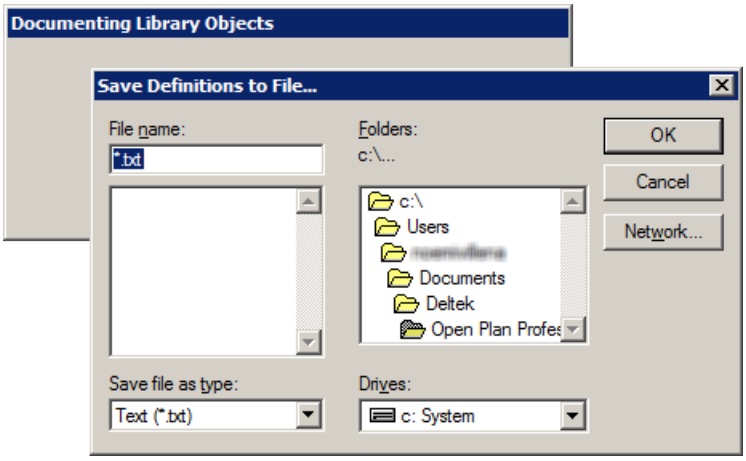
Note that in the case of a document, the path to the executable is not specified. Windows opens the file with a suitable application based on the document's file extension.

Activity ID	Activity Desc.	User Character Field 1
1	Determine Overall Layout for Phase 1	www.deltek.com
2	Survey and Subdivide	D:\Notes\phone.txt

String: Tool2= Samples:Document Launcher;%SYSTEMDIR%\Sample Tools\vbapps\actrun\act_run.exe %P %A

Document Library Objects

The Document Library Objects tool allows you to create a text file containing customized calc field, filter, sort, and global edit definitions within a project. When selected, the Document Library Objects tool instructs you to select a destination for the definitions of library objects.



A dialog box will indicate the status of the files as the objects are being generated. Once the files have been saved, the **View File** dialog box will ask you if you want to view the text file.

String: Tool3= Samples:Document Library Objects;%SYSTEMDIR%\Sample Tools\vbapps\doclib\DOCLIB.EXE %P

Deltek Web site

The Deltek Web site tool instructs Open Plan to open Deltek's Web site within an Open Plan window.

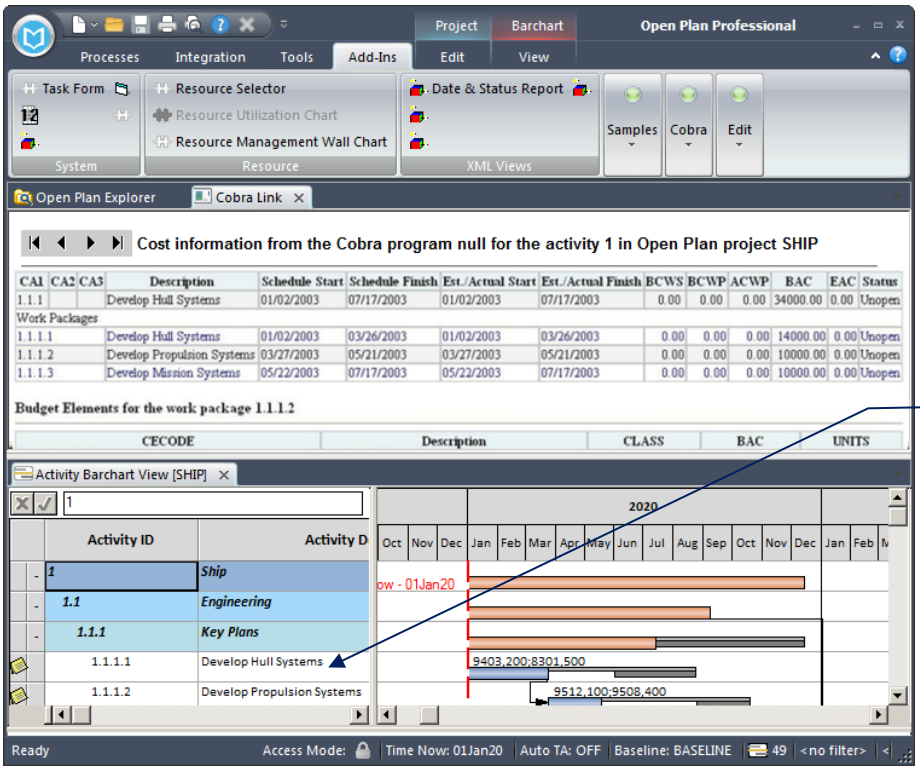
You can change this option to open the Deltek Web site in an external Internet browser window by modifying the parameters for the tool in the AddIns.dat file.

String: Tool4=Deltek Website;www.Deltek.com %I

Cobra Link

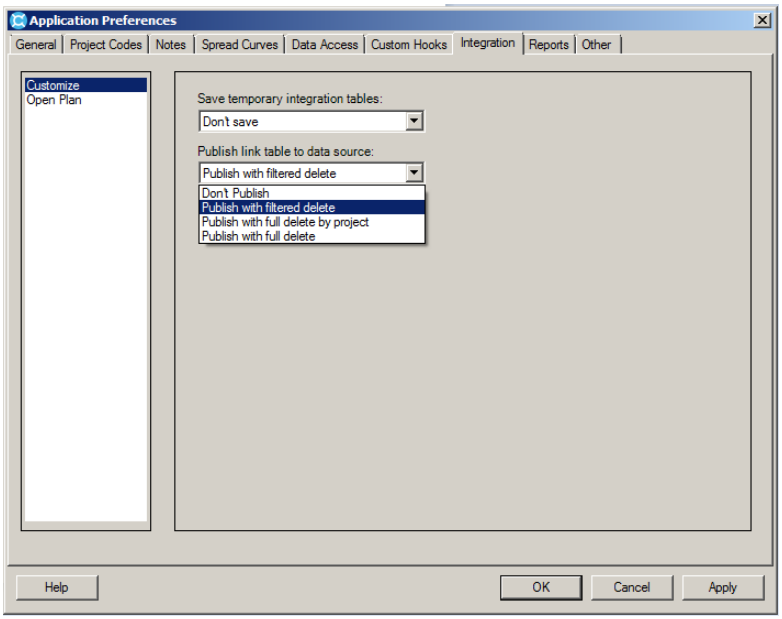
The Cobra Link tool allows users with both Cobra and Open Plan installed, to view integrated cost data in Cobra from within Open Plan. Through integration, you can see the derived costs of resources within Open Plan along with actual cost and earned value information.

When you run the Cobra Link tool, Open Plan opens the Cobra Link window along with the activity view from Open Plan.



To view cost information for an activity, select the activity in the Open Plan view. The Cobra Link window displays the cost account and work package information for the activity selected in Open Plan. You can view budget elements and direct and derived costs within this window.

To run Cobra Link, Cobra and Open Plan tables should reside in the same database. Once the tables are in the appropriate database, run Application Preferences in Cobra. Click the Integration tab, and select **Customize**.



Select the Publish Link table to data source option. Continue with the rest of the Integration Wizard.

Attention: For information on running the Integration Wizard in Cobra, refer to the Deltek Cobra User's Guide.

By default, Cobra Link tool is commented out in the AddIns.dat file. To enable it, select **Edit AddIns.dat** from the Add-Ins tab, Locate the string for **Cobra Link**:

String: Tool13= Cobra:Cobra Link;%SYSTEMDIR%\Sample Tools\OPXMLview\OPCobra\OPCobra.htm %i %P %A %T ; "%SYSTEMDIR%\Sample Tools\Cobra.ico;0"

Remove the semicolon at the beginning of the string. Save the file and click **Reload Add-Ins** from the Add-Ins tab, **Cobra Link** should now appear under your Add-Ins tab option.

Cobra Sync

This tool lets you run the Cobra Integration Wizard as a batch process from Open Plan. You are prompted to select the configuration file for the wizard to use. Also, you can choose to have Open Plan remember this for each project or prompt you each time you run the Integration Wizard.

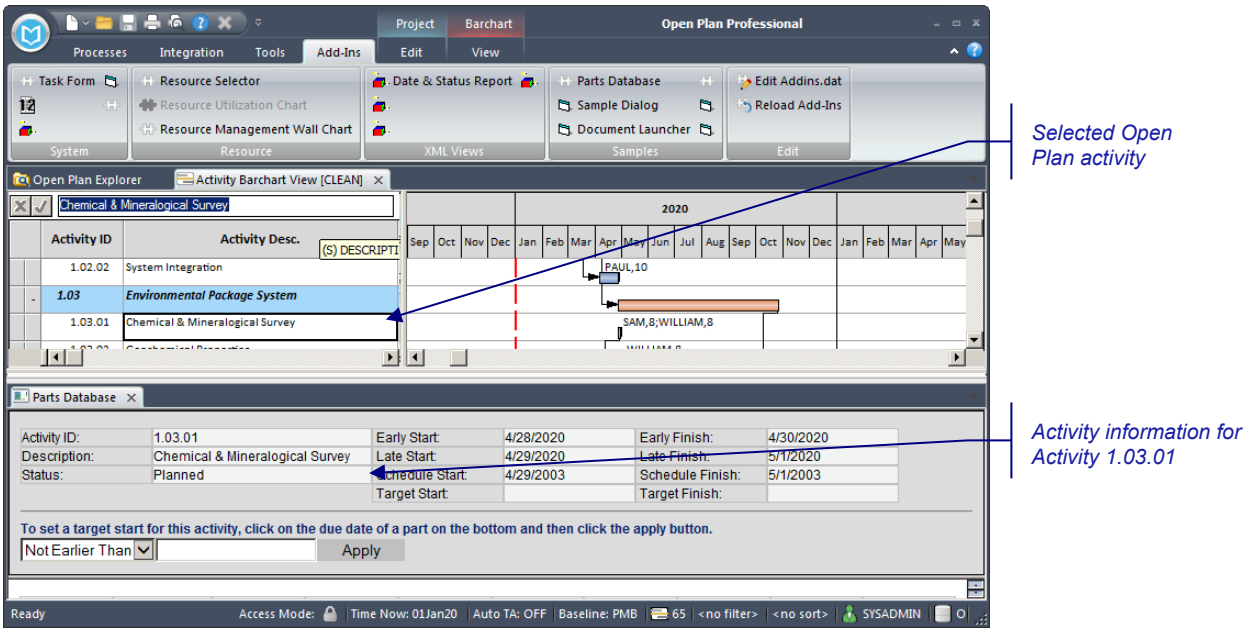
By default, **Cobra Sync** is commented out in the AddIns.dat file. To enable the tool, select the **Edit AddIns.dat** tool from the Add-Ins tab. Locate the string for **Cobra Cost Detail**:

String: Tool15=Cobra:Sync with Cobra;%SYSTEMDIR%\Sample Tools\vbapps\CobraSync\OpCobSync.exe %p ;%SYSTEMDIR%\Sample Tools\Cobra.ico;0

Remove the semicolon at the beginning of the string. Save the file and click **Reload Add-Ins** from the Add-Ins tab. **Cobra Sync** should now appear under your **Add-Ins** tab.

Parts Database

The Parts Database tool is an example of how you can integrate Open Plan with third party applications such as Oracle. The sample tool uses a Microsoft® Access sample database containing parts information for the project SHIP. Using the Parts Database tool, you can click on an activity within Open Plan and view the parts needed to complete it.



The Parts Database window displays the activity information along with its start and finish dates. Below this information is a table listing parts associated with the activity, their order status, due dates or date received, quantity, unit cost, price and ship date. If the part is past due, the due date field in the table is highlighted in red. You can change the target start for an activity using the drop-down box in the Parts Database window.

The Parts Database tool has been created to provide an example of how Open Plan can be integrated with a third party application containing information specific to your project. Just as any of the sample tools provided in the Add-Ins tab, it can be modified for customization.

String: Tool12=Samples:Parts Database;%SYSTEMDIR%\Sample Tools\OPXMLview\ExtData\ExtDataView.htm %i %P

Cobra Cost Detail

This WebWindow shows the Cobra budget elements associated with the selected activity and their time-phased details.

By default, **Cobra Cost Detail** is commented out in the AddIns.dat file. To enable the tool, select the **Edit AddIns.dat** tool from the Add-Ins tab. Locate the string for **Cobra Cost Detail**:

String: Tool16=Cobra:Cobra Cost Detail;%SYSTEMDIR%\Sample Tools\OPXMLview\costdetail\costdetail.htm?DOCKSIDE=BOTTOM&DOCKLENGTH=310 %d %P %A %T ;%SYSTEMDIR%\Sample Tools\Cobra.ico;0

Remove the semicolon at the beginning of the string. Save the file and click **Reload Add-Ins** from the Add-Ins tab. **Cobra Cost Detail** should now appear under your Add-Ins tab option.

Open Plan Batch Processor

The Open Plan Batch Processor (OPBATCH) allows users to perform OLE Automation functions from a scripting language without the difficult task of high-level programming. The objective is to allow non-programmers to automate repetitive tasks without assistance from the programming staff. As an added benefit, the Open Plan Batch Processor can generate native Visual Basic Script that can be executed without requiring the end user to have access to a copy of the Batch Processor.

> **Attention:** For information on the Open Plan Batch Processor and how it is implemented, refer to the document titled *Open Plan Batch Processor* located within the Open Plan destination folder.

Datasources.dat File

The Datasources.dat file specifies the complete list of available data sources. This file resides in the folder specified in the **DataSources** setting in the **[System]** section of the Config.dat file. It is maintained through the Data Sources dialog box displayed when you click **File » Preferences » Data Sources**.

Briefcase.dat File

The Briefcase.dat file specifies a user-specific briefcase data source. Briefcasing data sources cannot be shared by multiple users. The Briefcase.dat file resides in the folder specified by the **UserDir** setting in the Windows Registry.

Spread Curve Profiles

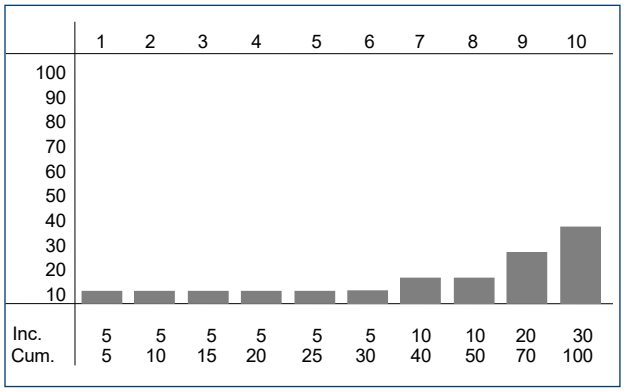
In Open Plan, resource and cost assignment profiles can be defined by spread curves that describe how the assignment is spread through the duration of an activity. Open Plan provides the following standard profiles:

- Back load (B)
- Double peak (D)
- Early peak (E)
- Front load (F)
- Late peak (L)
- Bell (Normal) (N)
- Linear (T)

All assignment profiles are described by defining values for 10 evenly spaced percentile points, which are then mapped to the activity duration. The following sections discuss each of these assignment profiles in more detail.

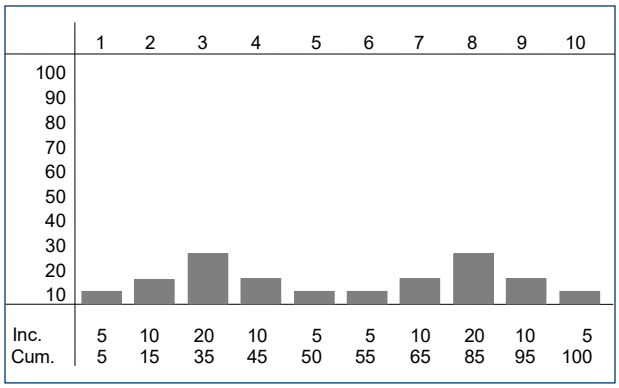
Back Load Profile

The back load assignment profile describes an assignment in which most of the resource effort is allocated toward the end of the activity duration. The curve code for a back load assignment profile is B.



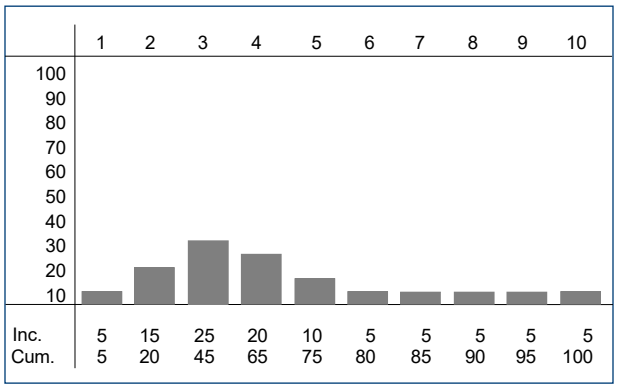
Double Peak Profile

The double peak assignment profile describes an assignment in which most of the resource effort is allocated to two peaks — one near the beginning of the activity duration, and one near the end. The curve code for a double peak assignment profile is D.



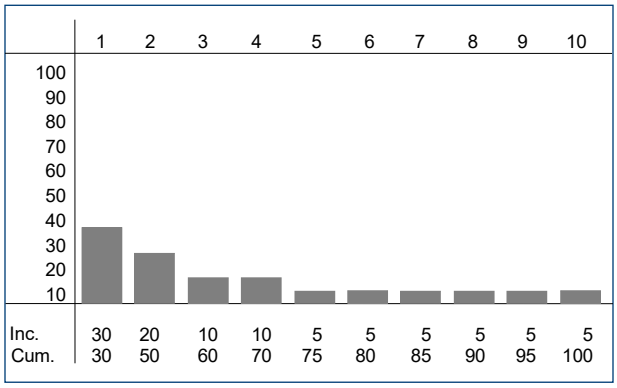
Early Peak Profile

The early peak assignment profile describes an assignment in which most of the resource effort is allocated near the beginning of the activity duration. The curve code for an early peak assignment profile is E.



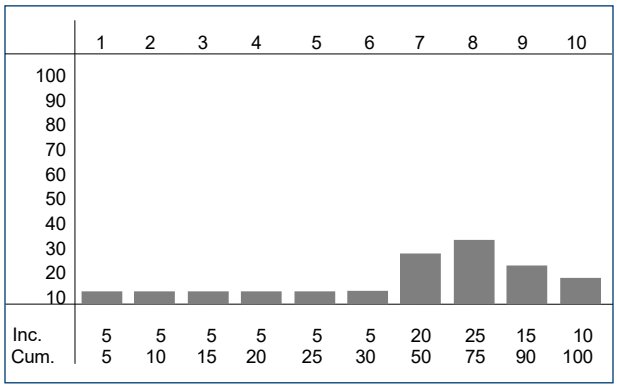
Front Load Profile

The front load assignment profile describes an assignment in which most of the resource effort is allocated at the beginning of the activity duration. The curve code for a front load assignment profile is F.



Late Peak Profile

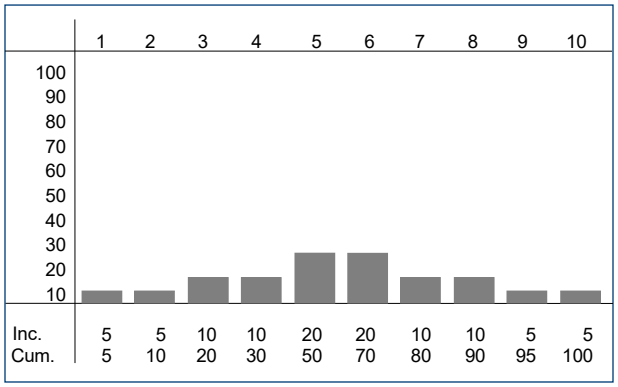
The late peak assignment profile describes an assignment in which most of the resource effort is allocated near the end of the activity duration. The curve code for a late peak assignment profile is L.



Bell (Normal) Profile

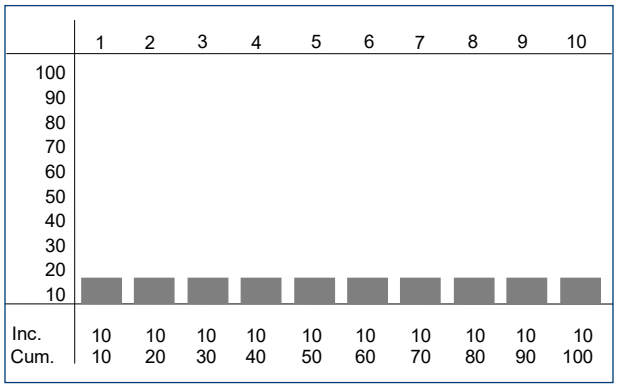
The bell (normal) assignment profile describes an assignment in which a large proportion of the resource effort is allocated near the middle of the activity duration. The resource effort starts with a minimal effort,

gradually increases to the middle of the duration, and then tapers off toward the end. The curve code for a bell (normal) assignment profile is N.



Linear Profile

The linear assignment profile describes an assignment in which the resource effort is allocated as evenly as possible throughout the activity duration. The curve code for a linear assignment profile is T.



Date Formats in Open Plan

In Open Plan, you can manually define date formats using the **Date Scale Preferences** dialog box when:

- You define the date scale in a barchart view
- You define the date scale in a resource or risk histogram view
- You format dates using the **DATEFORMAT()** function in a calculated field or global edit expression.

You can also define date formats from an examples list on the **Preferences** tab of the **Project Properties** dialog box.

Note: For more information on date formats, see the *Date Scale Preferences Dialog Box* topic (Views and Reports » Tabs and Dialog Boxes) and the *Preferences Tab* topic (Working with Projects » Tabs and Dialog Boxes » Project Properties Dialog Box) in the Open Plan online help.

Defining Date Formats

You can specify the appearance of a date format using one or more of the following parameters:

Parameter	Definition	Example
%T	Minute	15
%H	Hour	8
%Z	AM or PM	AM or PM
%OZ	AM or PM (lower case)	am or pm
%D	Day of the month	31
%V	Day	Tuesday
%OV	Day (lower case)	Tuesday
%UV	Day (upper case)	TUESDAY
%W	Day abbreviation	Tues
%OW	Day abbreviation (lower case)	tues
%UW	Day abbreviation (upper case)	TUES
%F	Single character day	T
%OF	Single character day (lower case)	t
%K	Week number	22
%M	Numeric month	10
%L	Month	October
%OL	Month (lower case)	october
%UL	Month (upper case)	OCTOBER
%A	Month abbreviation	Oct
%OA	Month abbreviation (lower case)	oct
%UA	Month abbreviation (upper case)	OCT
%S	Single character month	O
%OS	Single character month (lower case)	o

Parameter	Definition	Example
%Q	Numeric quarter	4
%C	Year	2003
%Y	Two-digit year	03

You can combine parameters to define date formats. You can also include spaces and literal characters in the format, as shown in the following examples:

Format	Example
%H:%T %Z	07:15 PM
%W %D %A %H:%T	Mon 13 Sep 19:15
%M/%D/%C	09/13/2003
Q%Q	Q1

Note: In Open Plan, hours and minutes default to the 24-hour (military) convention. The %Z or the %OZ parameter in the definition of the format changes hours and minutes to the standard time format.

To display relative dates in a barchart or histogram date scale, use the following parameters:

Parameter	Definition
%R%H	Relative hours
%R%D	Relative days
%R%K	Relative weeks
%R%M	Relative months
%R%Q	Relative quarters
%R%Y	Relative years

Note: In barchart and histogram views, the values displayed for dates using a relative format are based on the Reference Date setting on the Manual tab of the Date Scale Preferences dialog box. To have Open Plan convert the currently selected date format to a relative date format, select the Relative to Reference Date option from the Date Formats dialog box.

All of these different types of format parameters can be combined with literal characters to produce the following types of labels:

Format	Example
%R%H Hours	100 Hours
Day %R%D	Day 3
Week %R%K	Week 15
FY%R%Y	FY3

Project Utilities

Open Plan includes a number of project-level utilities that facilitate the manipulation of project data. These include:

- Filters
- Sorts
- Calculated Fields
- Global Edits
- User Defined Fields
- User Defined Validations
- Spread curves

Filters

Filters limit the display of items in a view based on a specific criterion. For example, you may wish to display a view that shows only activities requiring a particular resource. Open Plan provides a number of predefined filters common to many project management reports. In addition to these predefined filters, you can define custom filters of your own.

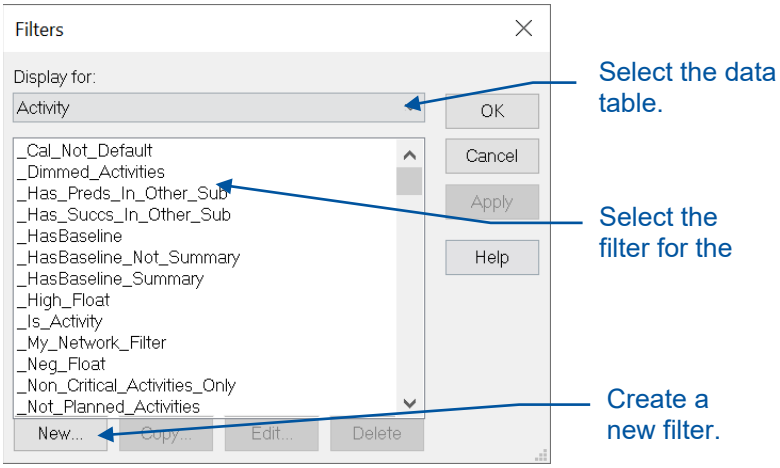
When applying a filter to a view, keep the following points in mind:

- When selecting filtered activities, the status of the parent item determines the status of the child. For example, assume that you apply a filter to a view and then select a parent activity or resource pool. This selects all children of the item, even if they do not meet the original filter criteria.
- If you apply a filter to a spreadsheet or barchart view with outlining turned on, the status of child items is determined by the status of the parent when you select the Filter from the Top Down option. For example, if a parent activity fails to meet the criteria of a filter, none of that parent's children will appear in the view, even in cases where the individual child activities satisfy the filter.
- If you apply a filter to a linked spreadsheet view, the filter applies only to information from the primary data table for the view.

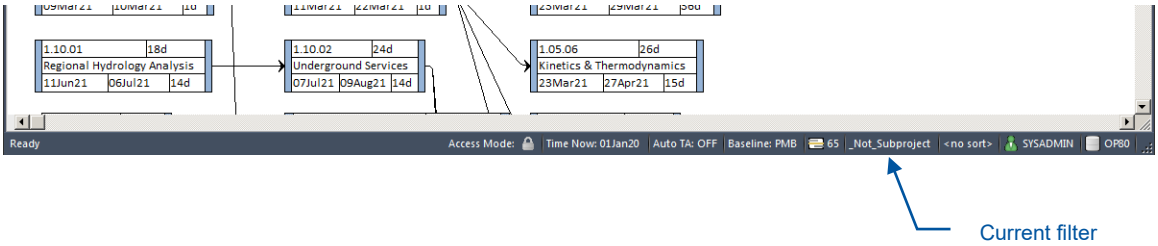
Attention: For information about outlining and linking in spreadsheets, see **Views and Reports » Spreadsheet Views** in the Open Plan online help.

Using Filters

To display a list of available filters, on the **Tools** tab click the **Filters » Manage Filters**. Open Plan responds by displaying the following dialog box:



Each view for a project can have a different filter in effect. The name of the filter currently in effect appears in the status bar at the bottom of the Open Plan window:



To reapply the filter (or the sort) without exiting the view, click the **Refresh** button on the toolbar.

Once a filter is in effect for a view, you can continue to add items to the view. Open Plan displays the items as you add them, even if the new items do not satisfy the current filter condition. However, if you refresh the view or close the view and return to it later, the items you added may not be displayed if the filter expression excludes them.

To apply a filter in a network view, you must use the **Placements** command.

> **Attention:** For more information on applying a filter in a network view, see **Views and Reports » Network View** in the Open Plan online help.

To apply a filter, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Filters**
   - On the **Edit** tab, click **Filters**.
   - Right-click within the view, and click **Filters** on the context menu.
   - Double click the current filter (or No Filter) name in the status bar.
2. From the **Filters** dialog box, select the filter you want to apply.

3. Click **OK**.

To refresh a filter, complete the following steps:

1. On the toolbar, click the **Refresh** button.

To cancel a filter, complete the following steps:

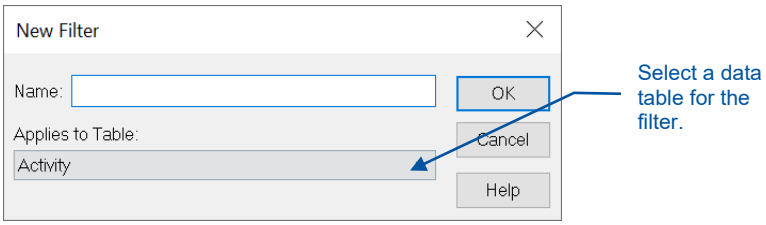
1. Take one of the following actions:
   - On the **Tools** tab, click **Filters**
   - On the **Edit** tab, click **Filters**.
   - Right-click within the view, and click **Filters** on the context menu.
   - Double click the current filter (or No Filter) name in the status bar.
2. From the **Filters** dialog box, select **<Cancel filter>**, and click **OK**.

Defining Filter Expressions

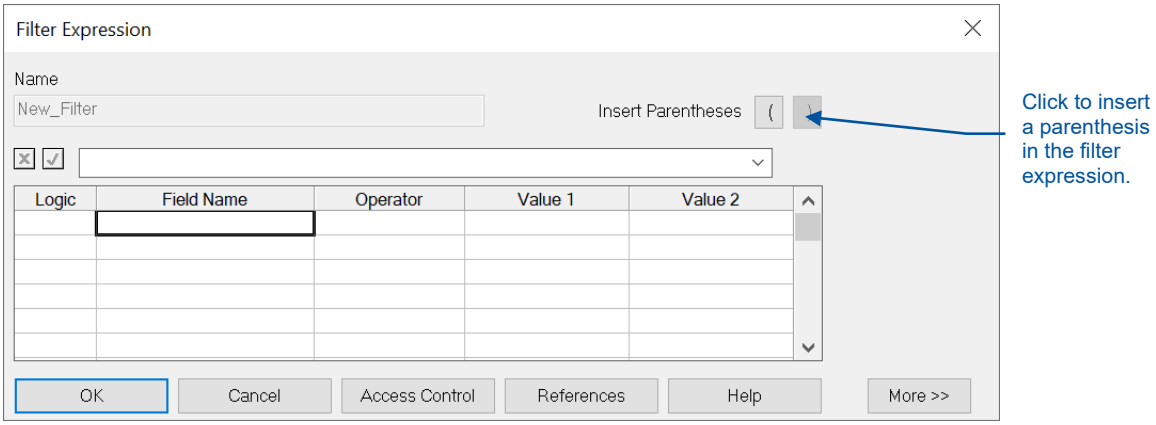
Open Plan allows you to define custom filter expressions that can be applied to views of project information. These custom expressions can be based on any field from the activity- or resource-related data tables and can feature complex expressions using logical operators and parentheses. Once you create a filter, you can make that filter available to other users in the network.

In addition to their use as view filters, filter expressions can appear as criteria for bar attributes in a barchart view or for box attributes in a network view.

When you create a new filter, Open Plan displays a dialog box in which you can enter a filter name and indicate the source of the fields to appear in the filter.



Once you have entered a name and a data table for the filter, Open Plan allows you to enter the expression for the filter criterion using the following dialog box:



With this dialog box, you can build a filter using the following components.

- **Logic** — You can build complex filter expressions using the following logical operators:
- AND
- AND NOT
- OR
- **Field Name** — Filter expressions can include any field from the selected data table, including calculated fields and any filters already defined for that table. Previously defined filters specified as fields can have one of two values: true or false.

Attention: For information about calculated fields, see Defining Calculated Fields.

- **Operator** — Filter expressions can contain any of the following operators:
- Between
- Contains (This operator is case-sensitive. Criteria entered using this expression must be typed in the same case as the original value. If the same case is not used, no results will be displayed.)
- Equals
- Greater or Equals
- Greater Than
- Is Empty (available for text fields only)
- Less or Equals
- Less Than
- Not Between
- Not Contains
- Not Empty (available for text fields only)
- Not Equals
- **Value 1** — The values available for a filter expression depend on the type of field used in the expression. For example:
- If you select a text, numeric, or date field, Open Plan allows you to select another field from the table (including calculated fields and previously defined filters), or you can enter a text, numeric, or date constant.

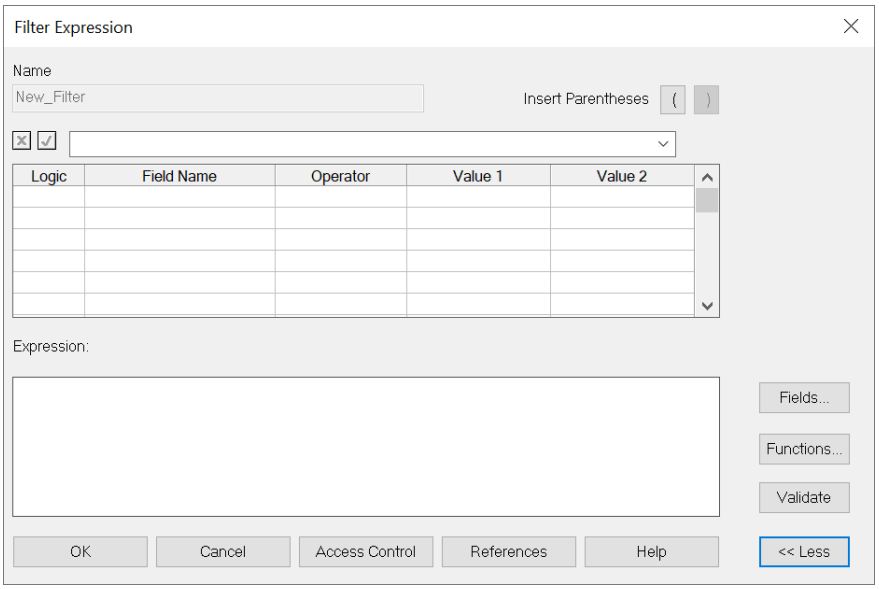
  If you are defining a filter that includes an activity or resource ID, make sure that you enter the value using upper-case characters.
- If you select an enumerated field for the expression, Open Plan displays a list of valid choices from which you must select. For example, the **Activity Type** field is an enumerated field with valid values such as ASAP, ALAP, or Start Milestone.
- If you select a logical field for the expression, Open Plan allows you to select either true or false as a value.
- **Value 2** — If the expression uses either the **Between** or **Not Between** operators, you can enter a second value for the filter.

- **Share this Item with Others** — This option allows you to permit other users to use the filter that you are creating. This option is enabled only for permanent filters that you create. You cannot share a temporary filter.

This setting is useful, for example, for managers who want to create a filter that they do not want others to have access to.

- **Insert Parentheses** — You can insert a left and right parenthesis in any expression. Left parentheses can be inserted at the beginning of the **Field** column; right parentheses can be inserted at the end of a row in the **Value 2** setting. Parentheses allow you to develop complex expressions such as the following examples:
- C1 Equals “1.1” AND (C2 Equals “QA” OR C2 Equals “Pubs”)
- (C1 Equals “1.1” AND C2 Equals “QA”) OR C2 Equals “Pubs”

When you click **More**, the **Filter Expression** dialog box extends to display additional controls:



- **Expression** — This field reflects the actual expression that is displayed in the grid. If you prefer, you can enter the filter expression directly in the text box.
- **Fields** — Clicking this button displays the **Fields** dialog box. You can use this dialog box to select fields to include in the filter expression.
- **Functions** — Clicking this button displays the **Functions** dialog box. You can use this dialog box to select functions to use in the filter expression.
- **Validate** — When you have created a filter expression, you can click this button to have Open Plan validate the expression. If the filter expression is valid, Open Plan populates the grid with the appropriate values.
- **Less** — Clicking this button hides the dialog box extension.

To add a filter expression, complete the following steps:

1. Take any of the following actions:
   - On the **Tools** tab, click **Filters »** select a filter**,** then click **New**.
   - On the **Edit** tab, click **Filters »** select a filter**,** then click **New**.

- Right-click within the view, and click **Filters** on the context menu **»** select a filter**,** then click **New**.
- On the **Edit** tab, click **Edit Filter**.

2. In the **New Filter** dialog box, enter a name, and select a source data table for the new filter.
3. Click **OK**.
4. In the **Filter Expression** dialog box, define the filter expression.
5. When the information for the filter expression is complete, click **OK** to return to the **Filters** dialog box.

To edit a filter expression, complete the following steps:

1. Take any of the following actions:
   - On the **Tools** tab, click **Filters »** select a filter**,** then click **Edit**.
   - On the **Edit** tab, click **Filters »** select a filter**,** then click **Edit**.
   - Right-click within the view, and click **Filters** on the context menu, **»** select a filter**,** then click **Edit**.
2. In the **Filter Expression** dialog box, update the filter expression.
3. When the information for the filter expression is complete, click **OK** to return to the **Filters** dialog box.

To copy a filter expression, complete the following steps:

1. Take any of the following actions:
   - On the **Tools** tab, click **Filters »** select a filter**,** then click **Copy**.
   - On the **Edit** tab, click **Filters »** select a filter**,** then click **Copy**.
   - Right-click within the view, and click **Filters** on the context menu, **»** select a filter**,** then click **Copy**.
2. In the **Filter Expression** dialog box, enter a new name for the filter.
3. When the information for the new filter expression is complete, click **OK** to return to the **Filters** dialog box.

To delete a filter expression, complete the following steps:

1. Take any of the following actions:
   - On the **Tools** tab, click **Filters »** select a filter**,** then click **Delete**.
   - On the **Edit** tab, click **Filters »** select a filter**,** then click **Delete**.
   - Right-click within the view, and click **Filters** on the context menu, **»** select a filter**,** then click **Delete**.
2. When Open Plan asks you to confirm the deletion, click **Yes**.

Additional Filtering Options

The following options are also available for filtering:

- **Add to Filter** — This option takes a selection range and adds the selection values to the currently selected filter based on the following rules:

- Multiple selections within a column are added to the filter with an OR EQUALS condition
- Multiple selections across columns are added to the filter with an AND EQUALS condition
- **Remove from Filter** — This option takes a selection range and adds the selection values to the currently selected filter based on the following rules:
- Multiple selections within a column are added to the filter with an OR NOT EQUALS condition
- Multiple selections across columns are added to the filter with an AND NOT EQUALS condition
- **Clear Filter** — This option removes the currently selected filter. It is equivalent to displaying the **Manage Filters** dialog box and choosing **Cancel Filter**.
- **Toggle Filter** — Once a filter has been applied, this option turns the filter on or off while still remembering it.
- **Edit Expression** — This option displays the Filter Expression dialog box while bypassing the Manage Filters dialog box.
- **Reapply Filter/Sort** — Select this option to apply the last filter and sort again.

To apply one of these options, complete the following steps:

1. On the **Edit** tab, click **Add to Filter**.

Temporary Filters

In addition to creating permanent filters, you can also create temporary filters. Temporary filters are available only in the view in which they are defined and valid until superseded by another filter. You cannot make these temporary filters available to other projects, nor can you make them available to other users with the **Share this Item with Others** option.

Temporary filters can be created in two ways:

- By selecting **<Temporary Filter>** on the **Filters** dialog box

  Creating a temporary filter in this manner uses the same **Filter Expression** dialog box used for creating a new filter but with two exceptions: the name of the filter is **<temporary>** and the **Share this Item with Others** option is disabled.

- By typing the temporary filter expression directly in the field where it will be used.

  For example, you can type the temporary filter expression directly into the **Criterion** field on the **Bar Attributes** tab of:

  - The Bar Set Preferences dialog box
  - The Network View Preferences dialog box

The manner in which the **Filter Expression** dialog box for a temporary filter is displayed depends on the view from which it is accessed:

- In network views, you can display the **Filter Expression** dialog box for a temporary filter by clicking **Filters** on the **Placements** dialog box and then selecting **<Temporary Filter>** from the **Filters** dialog box.
- In spreadsheet and barchart views, you can display the **Filter Expression** dialog box for a temporary filter by clicking **Filters** on the **Tools** tab and then selecting **<Temporary Filter>** from the **Filters** dialog box.

The temporary filter created in this manner can be applied only to a primary table. In multi-table views, you can apply a filter to a secondary table using the **Define Link** tab of the **Spreadsheet Preferences** dialog box.

To define a temporary filter using the Filter Expression dialog box

1. Display the view in which you want to apply the temporary filter.
2. Take one of the following actions:
   - On the **Tools** tab, click **Filters**
   - On the **Edit** tab, click **Filters**.
   - Right-click within the view, and click **Filters** on the context menu.
   - On the **Tools** tab, click **Filters**.

   Open Plan displays the **Filters** dialog box.
3. Take one of the following actions:
   - Select **<Temporary Filter>**, and click **Edit**.
   - Double-click **<Temporary Filter>**.
4. In the **Filter Expression** dialog box, define the temporary filter in the grid.

   Click **More** to expand the **Filter Expression** dialog box if you want to use the **Fields** and **Functions** buttons as an aid in building your filter expressions.
5. Click **OK**.

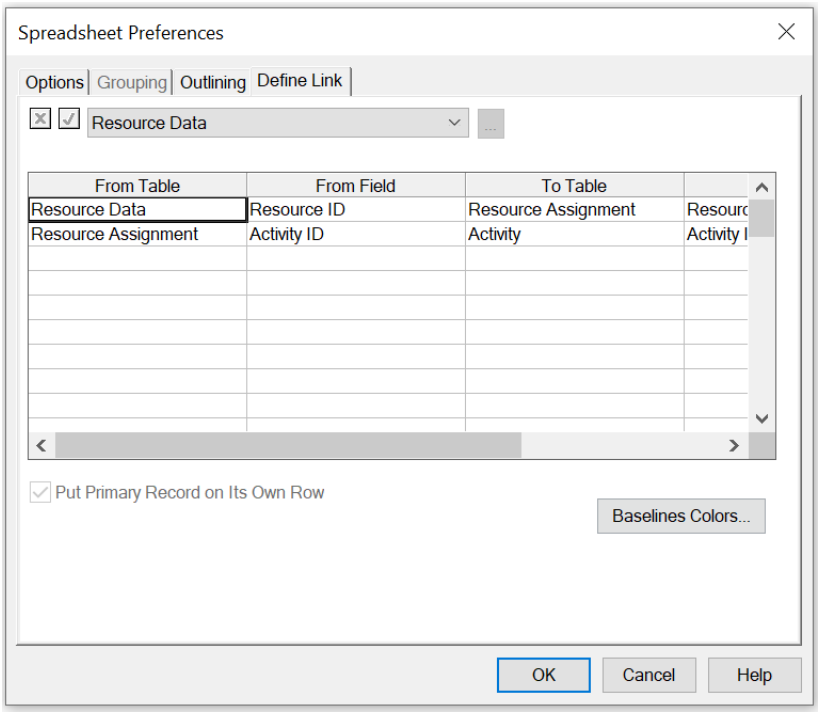
To define a temporary filter in a Network view, complete the following steps:

1. On the **View** tab, click **Placements**.
2. On the **Placements** dialog box, click the **Filters** button to display the **Filters** dialog box.
3. Take one of the following actions:
   - Select **<Temporary Filter>**, and click **Edit**.
   - Double-click **<Temporary Filter>**.
4. In the **Filter Expression** dialog box, define the temporary filter in the grid.

   Click **More** to expand the **Filter Expression** dialog box if you want to use the **Fields** and **Functions** buttons as an aid in building your filter expressions.
5. Click **OK**.

Temporary Filters on a Secondary Table

You can apply a temporary filter to a secondary table in a multi-table view by using the **Define Link** tab of the **Spreadsheet Preferences** dialog box:



The filter for data from all subsidiary tables is controlled independently, so a new filter on a subsidiary table is in addition to a filter on a higher level.

If you inadvertently enter the filter expression on the wrong row, Open Plan attempts to automatically move the filter expression to the correct row. This will be visible the next time that you display the **Define Link** tab.

To define a temporary filter on a secondary table in a multi-table view, complete the following steps:

1. Open the multi-table view to which you want to apply a temporary filter.
2. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click within the view, and click **Preferences** on the context menu.

   Open Plan displays the **Spreadsheet Preferences** dialog box.
3. Click the **Define Link** tab.
4. In the **Filter** field, enter the temporary filter you want to apply, and click **OK**.

   Open Plan applies the filter to the secondary table identified in the table column.

These temporary filters are not displayed in the **Filters** dialog box. They are not saved as part of the project. If you save a view while a temporary filter is in effect, Open Plan saves the view with the filter applied.

In addition to using temporary filters in Open Plan views, you can create and use temporary filters in the following circumstances:

- Export scripts
- OLE automation

Sorts

By default, Open Plan displays activities and resources in barchart and spreadsheet views in order of ID. However, you can assign a different sorting sequence to a view so that, for example, activities appear in order of early start dates.

Open Plan provides a number of predefined sorts that are commonly used in project management reports. In addition to these predefined sorts, you can define custom sorts of your own.

If a view is displayed using outlining or groupings, Open Plan sorts the items within the outline or grouping hierarchy.

Default Sort Order Rules

The following rules apply to the default sort order, which is in effect when there is no sort specified. Explicitly sorting on Activity ID will still sort alphabetically.

- The default sort order for all hierarchical fields (activity IDs, resource IDs, and codes) is interpreted numerically, if possible. Therefore, "1.2" is sorted before "1.10." Character data is sorted alphabetically.
- Mixed character and numeric data is also sorted alphabetically, except that any trailing numeric digits are sorted numerically. For example, "A2" comes before "A10." Since each component of the name is sorted independently, some data can be numeric and some not.
- IDs with leading zeros are sorted first. For example, "A01" comes before "A1."

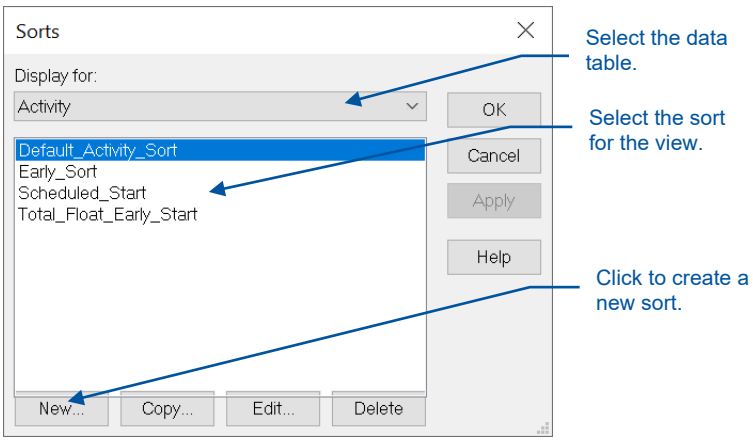
Also, when you have a table based upon a hierarchy, the **Refresh** button on the toolbar is enabled even when no sort or filter is specified.

The rules for the sort order are fundamental to the workings of the automatic numbering and indent/outdent features.

> **Attention:** For information on automatic numbering and the indent/outdent features, see **Views and Reports » Spreadsheet View** in the Open Plan online help.

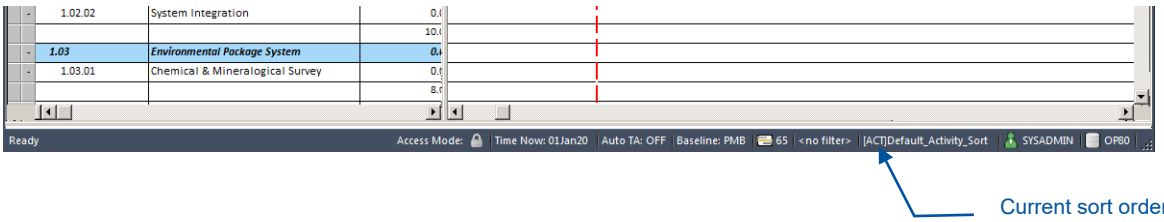
Using Sorts

Issuing the **Sorts** command displays a list of available sorts in the **Sorts** dialog box:



You can use this dialog box to apply an existing sort to the view as well as to cancel the sort currently in effect.

Each view for a project can have a different sort in effect. The name of the sort currently in effect appears in the status bar at the bottom of the Open Plan window:



To reapply the sort (or the filter) without exiting the view, click the **Reapply Filter And Sort** Reapply button on the toolbar.

> **Note:** Sorts do not affect the display of activities in a network view.

To apply a sort, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.
   - Double click the current sort (or No Sort) name in the status bar.
2. From the **Sorts** dialog box, select the sort you want to apply, and click **Close**.

To refresh a sort, complete the following steps:

On the **Edit** tab, click the **Reapply Filter And Sort** Reapply button.

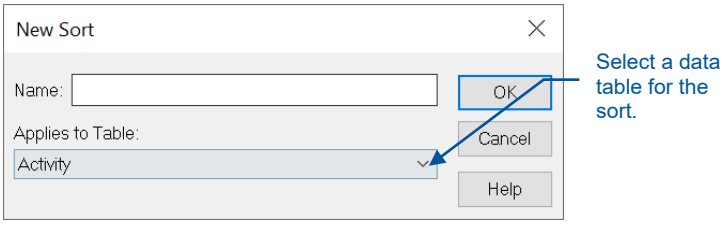
To cancel a sort, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.
   - Double click the current sort (or No Sort) name in the status bar.
2. From the **Sorts** dialog box, select **<Cancel sort>**, and click **Close**.

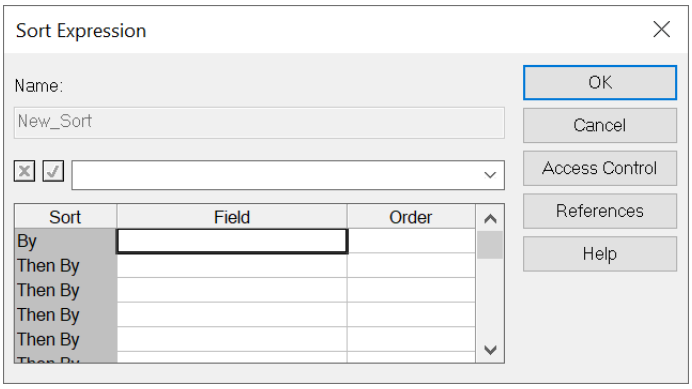
Defining Custom Sorts

You can define custom sort sequences based on data from any activity- or resource-related data table. Custom sorts can also include any calculated fields for those tables.

When you create a new sort, Open Plan displays a dialog box in which you can enter a sort name and indicate the source of the sort fields.



Once you have entered a name and selected a data table, Open Plan allows you to define the sort expression using the following dialog box.



With this dialog box, you can build a sort expression by indicating each sorting field and whether Open Plan should use an ascending or a descending order when sorting activities using that field. You can also choose to share this item with others.

You can define sort expressions based on substrings of a text field (for example, the activity description) by creating a calculated field containing the substring and then including the calculated field in the sort expression.

> **Attention:** For information about creating calculated fields, see Defining Calculated Fields.

To add a sort, complete the following steps:

1. Take one of the following actions:

- On the **Tools** tab, click **Sorts**.
- Right-click within the view, and click **Sorts** on the context menu.
- On the **Edit** tab, click **Manage** from the **Sorts** group.

2. On the **Sorts** dialog box, click **New**.
3. In the **New Sort** dialog box, enter a name, and select a source data table for the new sort expression.
4. Click **OK**.
5. In the **Sort Expression** dialog box, define the sort expression.
6. When the information for the expression is complete, click **OK** to return to the **Sorts** dialog box.

To edit a sort, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.
2. From the **Sorts** dialog box, select the sort you want to change, and click **Edit**.
3. In the **Sort Expression** dialog box, update the sort expression.
4. When the information for the expression is complete, click **OK** to return to the **Sorts** dialog box.

To copy a sort, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.
2. From the **Sorts** dialog box, select the sort you want to copy, and click **Copy**.
3. In the **Sort Expression** dialog box, enter a new name for the sort.
4. When the information for the new expression is complete, click **OK** to return to the **Sorts** dialog box.

To delete a sort, complete the following steps:

1. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.
2. From the **Sorts** dialog box, select the sort you want to delete, and click **Delete**.
3. When Open Plan asks you to confirm the deletion, click **Yes**.

Temporary Sorts

In addition to creating permanent sorts, you can also create temporary sorts. Temporary sorts are available only in the view in which they are defined and valid until they are superceded by another sort. You cannot make these temporary filters available to other projects or users.

You can create temporary sorts in the following ways:

- By clicking the column headings in a spreadsheet view if the **Click to Sort** option is enabled. Using this method you can sort the data in the spreadsheet based on either a single column or on multiple columns.

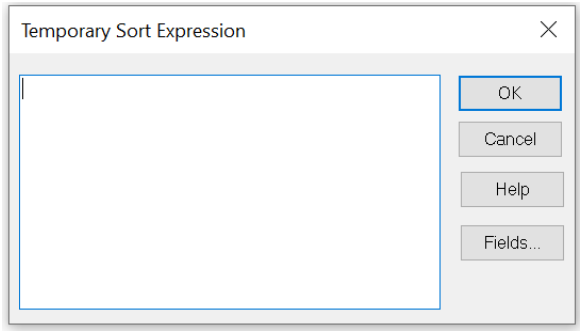
  > **Attention:** For more information on the **Click to Sort** option, see Click to Sort.

- By selecting **<Temporary Sort>** from the **Sorts** dialog box to display the **Temporary Sort Expression** dialog box.

The temporary sort created in the two manners described above can be applied to a primary table. In a multi-table spreadsheet view, you have the option of applying a sort to a secondary table using the **Define Link** tab of the **Spreadsheet Preferences** dialog box or by clicking the column headings as described above.

In a multi-table view, the primary table is the main table upon which the view is based. The secondary table is the linked table.

While you can simply type the sort expression in the text box of the **Temporary Sort Expression** dialog box, you can also use the **Fields** and **Functions** buttons to display dialog boxes that you can use as an aid in building the sort expression.



Open Plan temporary sorts are formed using the following syntax:

The syntax for a sort expression is:

<sort_field1>,<sort order>|<sort_field2>,<sort order>|…

where:

- Each of the **<sort_fieldx>** values is either the name of an existing field on the appropriate table or the definition of a calculated field for that table.
- The **<sort order>** indicates an ascending (**0**) or a descending (**1**) sort.
- The piping symbol (**|**) separates the sorts you create.

For example, the following expression:

ORIG_DUR,0|ACT_ID,1

Would first sort ascending the original duration field and then within that would sort descending the activity ID field.

In addition to using temporary sorts in Open Plan views, you can also use temporary sorts in the following circumstances:

- Export scripts
- OLE automation

To define a temporary sort using the Temporary Sort Expression dialog box, complete the following steps:

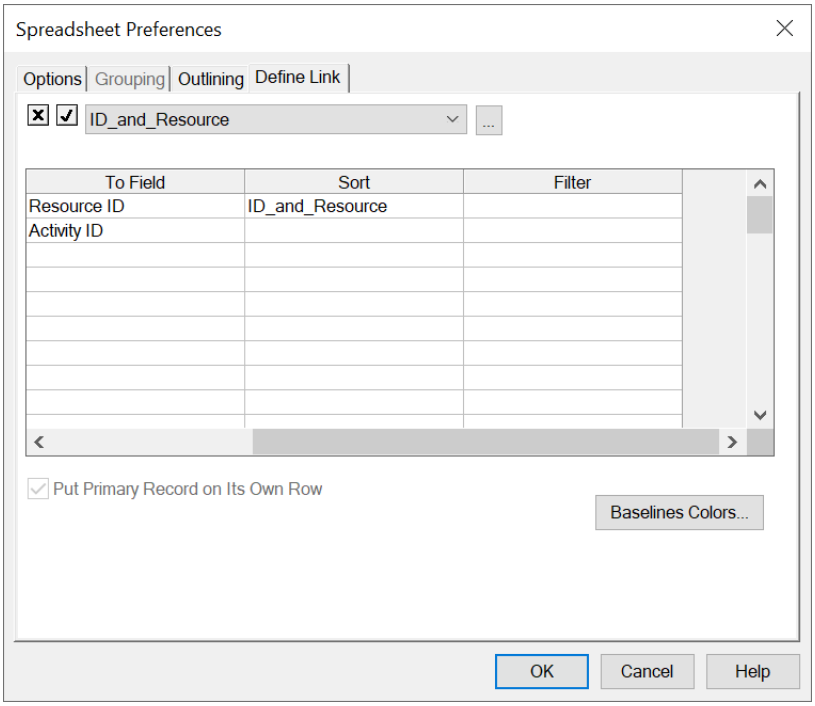
1. Display the view to which you want to apply the temporary sort.
2. Take one of the following actions:
   - On the **Tools** tab, click **Sorts**.
   - Right-click within the view, and click **Sorts** on the context menu.
   - On the **Edit** tab, click **Manage** from the **Sorts** group.

   Open Plan displays the **Sorts** dialog box.
3. Display the **Temporary Sort Expression** dialog box by taking one of the following actions:
   - Double-click **<Temporary Sort>**.
   - Select **<Temporary Sort>**, and click **Edit**.
4. Enter the temporary sort in the text box.

   You can use the **Fields** and **Functions** buttons as an aid in building your sort expressions.
5. Click **OK**.

Temporary Sort on a Secondary Table

You can apply a temporary sort to a secondary table in the multi-table spreadsheet view by using the **Define Link** tab of the **Spreadsheet Preferences** dialog box:



The sort order for data from all subsidiary tables is controlled independently, so a new sort on a subsidiary table is in addition to a sort on a higher level.

> **Note:** You can also create a temporary sort for a secondary table by applying the **Click to Sort** feature to column headings of the secondary table. For more information, see Click to Sort.

If you inadvertently enter the sort expression on the wrong row, Open Plan attempts to automatically move the sort expression to the correct row. This will be visible the next time that you display the **Define Link** tab.

To define a temporary sort on a secondary table in a multi-table view, complete the following steps:

> **Note:** When you sort columns based on a secondary table, using the **Click to** Sort option records the temporary sort in the **Sort** column of the **Define Link** tab of the **Spreadsheet Preferences** dialog box.

1. Open the multi-table spreadsheet view to which you want to apply a temporary sort.
2. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click an empty area of the view, and click **Preferences** on the context menu.
3. On the **Spreadsheet Preferences** dialog box, click the **Define Link** tab.

4. In the **Sorts** field, enter the temporary sort you want to apply, and click **OK**.

   Open Plan applies the sort to the secondary table identified in the **To Table** column.

   These temporary sorts are not displayed in the **Sorts** dialog box. They are not saved as part of the project. If you save the view while a temporary sort is in effect, Open Plan saves the view with the sort applied.

Click to Sort

In addition to the temporary sort, the spreadsheet and barchart views allow column sorting by clicking the column headings.

With the **Click to Sort** option enabled, you can click the column heading on which you want to base the sort. Clicking the same column heading again will toggle the sort operation between ascending and descending order.

To sort the data based on multiple columns, click the first column heading on which you want to sort. For non-adjacent columns, **Ctrl-click** additional columns until you have defined the entire sort expression. For adjacent columns, **Shift-click** additional columns until you have defined the entire sort expression.

When you sort columns based on a primary table in multi-table views, using the **Click to Sort** option records the temporary sort on the **Temporary Sort Expression** dialog box. If you sort the spreadsheet based on a secondary table, the sort expression is reflected in the **Sort** column on the **Define Link** tab of the **Spreadsheet Preferences** dialog box.

To enable the Click to Sort option, complete the following steps:

Take one of the following actions:

- Right-click a column heading, and click **Click to Sort** on the context menu.
- On the **View** tab, click **Click to Sort**.

To disable the Click to Sort option, complete the following steps:

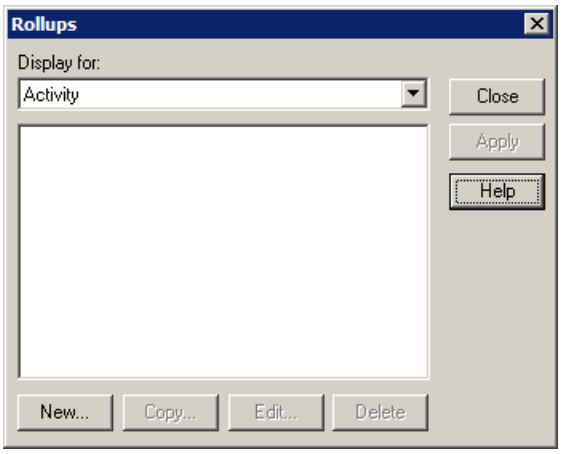
Take one of the following actions:

- Right-click a column heading, and click the **Click to Sort** option on the context menu to clear it.
- On the **View** tab, click **Click to Sort**.

Rollups

The Rollup utility provides the ability to perform a roll-up of date and numeric fields selected by the user. Rollups may be defined for Activities, Resources, and Codes.

Selecting **Rollups** from the **Tools** tab displays the following dialog box:



This dialog box allows you to perform a rollup of selected numeric and date fields for Activities, Resources, and Codes. For numeric fields, the parent object is the sum of its children. For date fields, the Start date of the parent object is the earliest date of its children, and the Finish date is the latest date of its children.

> **Note:** If any of the children have a blank finish date, a blank finish date is rolled up to the parent.

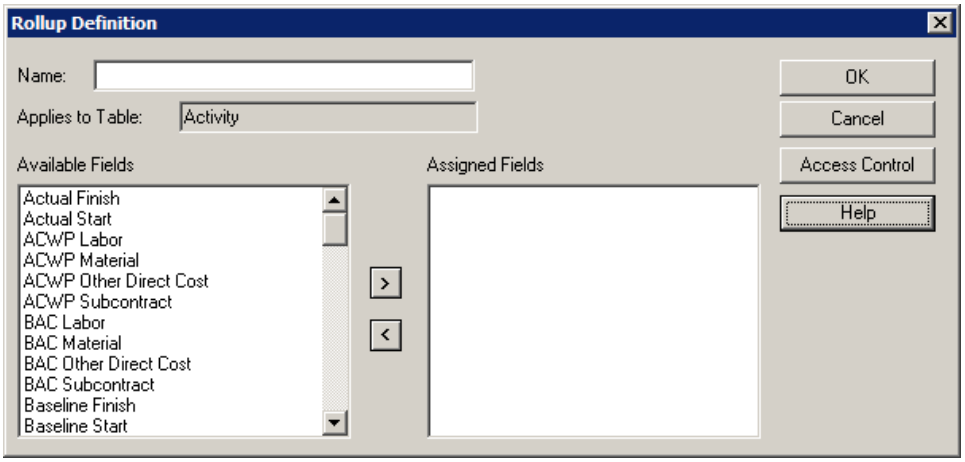
The **Display for** field allows you to select the table that contains the rollup definition you want to use.

When a table is selected, the dialog box is populated with all of the rollup definitions that have been created for that table. Selecting a rollup definition and clicking **Apply** performs the rollup for the current file.

For numeric fields, the parent object is the sum of its children. For date fields, the **Start** date of the parent object is the earliest date of its children, and the **Finish** date is the latest date of its children.

If any of the children have a blank finish date, a blank finish date is rolled up to the parent.

When you click **New** or select a rollup definition and click **Copy** or **Edit**, the **Rollup Definition** dialog box is displayed:



The **Rollup Definition** dialog box displays the following controls:.

- **Name** — If you are creating a new definition, this field will display blank, allowing you to add a name. If you are copying an existing definition, this field will be populated with the name, preceded by copy_ *name*. You can either accept the new name or replace it with a different name. If you are editing a definition, the name will display dimmed, and cannot be changed.
- **Applies to Table** — This field is populated with the table name that was previously selected, and cannot be changed on this dialog box.
- **Available Fields (left box)** — This lists all date and numeric fields created for the selected table. To apply a field to the rollup you are defining, double-click the edit name or click the > button to move the listed edit to the **Assigned Fields** window.

  Note: To move the entire list, select the list by clicking the first name, then depress the **Shift** key and click the last name in the list, then click the > button. To select a random group from the list to move, press and hold the **CTRL** key and click the mouse pointer on the items to be moved to highlight, then click the > button.

- **Assigned Fields (right box)** — This lists the fields the rollup definition will use. You can also remove any or all of the items in the **Assigned Fields** list by either double-clicking the item name, or clicking it to highlight and then clicking the < button. This will move the selected item or items to the **Available Fields** list.
- **Share this Item with Others**—By default, a rollup definition is only accessible to the user who creates it. By clicking the **Share this Item with Others** option, it can be made available to other users.

After you have created or edited rollup definitions, the display returns to the **Rollups** dialog box.

To execute a rollup for the current file, complete the following steps:

1. Select s rollup definition from the **Rollups** dialog box.
2. Click **Apply**.

Group Process Rollups

You can also perform a rollup in **Batch Mode**. This allows you to use a rollup definition for multiple projects at one time. To use a rollup definition in **Batch Mode**, select more than one project from the **Details** pane of the Open Plan Explorer, and click **Rollups** on the **Tools** tab. The **Rollups [Batch Mode]** dialog box will be displayed. Select the rollup definition to use, and click **Apply**.

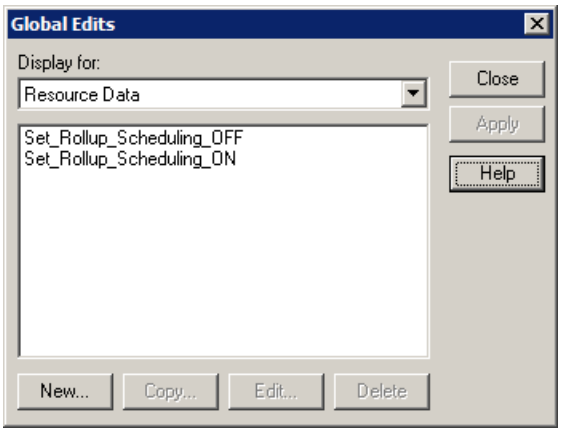
If a project is closed, Open Plan will open it in **Exclusive** mode, perform the process, save the changes, and then close the project. If a project cannot be opened in **Exclusive** mode (because another user has it open in **Exclusive** or **Shared** mode), it will be skipped.

If you have a project open when the batch process is run, rollup will be performed on the project, but the changes will not be saved and the project will remain open. You must manually save the changes yourself.

Global Edits

With the **Global Edits** utility, you can modify multiple activity, resource description, or resource assignment records with a single operation. Using the global edit utility you can, for example, increase all the durations in a project by 25% or change all the assignments for a specific resource. Global edit operations can modify all the records in a file or selected records based on a filter expression.

When you click **Global Edits** on the **Tools** tab, Open Plan responds with the following dialog box:



This dialog box displays a list of currently available global edits. Since they change all instances of the data, global edits may be applied regardless of whether or not any specific view is open.

The **Display For** field contains a list that segregates the global edits by type of data. For example, if you select **Project** from the list, the global edits that are displayed are project related.

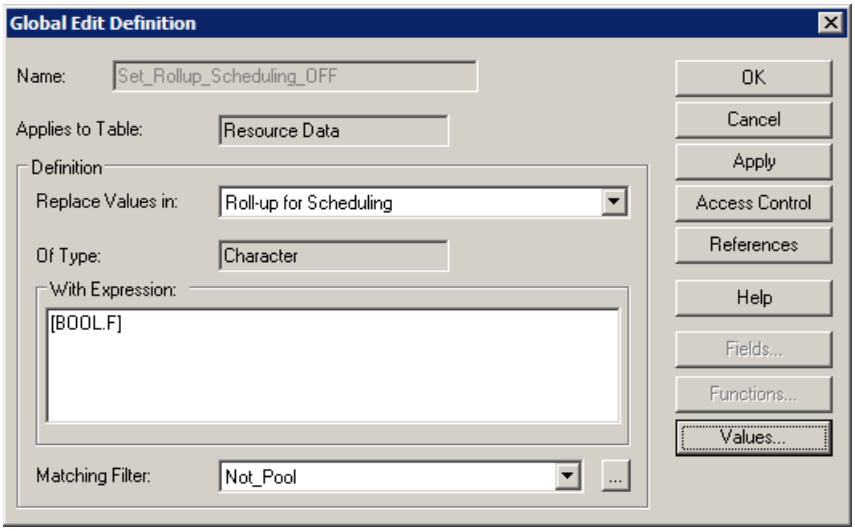
To apply a global edit, complete the following steps:

1. Open the project or resource file to which you want to apply the global edit.
2. On the **Tools** tab, click **Global Edits.**
3. Select the global edit you want to apply.

   To display the expression for the edit, click **Edit**.
4. Click **Apply**.

Defining Global Edits

You can define global edits for both project and resource files. By default, Open Plan stores the edit expression with the project or resource file that is open when you define the field. When you define a new global edit, Open Plan displays the following dialog box:



You can use the following controls to define the global edit:

- **Name** — If you have a project or resource file open, the default value in this field is **<temporary>**. By leaving this default value, you can create global edits without saving them.

> **Warning:** In order to save a global edit, you must change the default **<temporary>** value to a different name. When you then save the global edit, the name you entered is displayed on the **Global Edits** dialog box.

If you have accessed this dialog box without first opening a project or resource file, the **Name** field is blank since there is no target for a temporary sort.

- **Applies to Table** — Controls the data table to which the new global edit applies. The choices from which you can select depend on whether you are creating a global edit for a project or for a resource file:

Project	Resource
Activity	Resource Availability
Relationship	Resource Description
Resource Assignment	Resource Escalation
Resource Cost	Summary Usage
Resource Usage	
Risk Detail	
Subproject	

- **Replace Values in** — Controls the field of the data table to which the global edit applies.
- **Of Type** —Displays which of the following types of data is stored in the selected field of the data table:
- Date
- Decimal
- Duration
- Finish Date
- Integer
- Character
- Logical
- Enumerated values
- **With Expression** — Controls the expression that will replace the data in the selected data table field. The expression must create data of the same type as the data it will replace. For example, if the global edit replaces a date, the global expression must also create a date.

An expression defining a global edit can include the following elements:

- Character operators
- Constants
- Duration operators
- Field names
- Functions
- Logical operators
- Mathematical operators
- Calculated fields
- Relational operators
- **Matching Filter** — The **ellipsis** button at the right edge of this field displays the **Filters** dialog box that you can use to apply a filter to the global edit. In this way, you can limit the records that will be affected by the global edit.
- **Fields** — Displays a dialog box where you can select a field to include in the expression.
- **Functions** — Displays a list of available functions.
- **Values** — Displays the **Select a Value** dialog box that you can use to select a value to include in the global edit.

The expression for a global edit can use the same set of functions, operators, and field names that you use when creating calculated fields.

The following table shows some simple examples of operations you can perform on the original durations of activities in a project:

Expression	Result
ORIG_DUR-|1d|	Subtract one day from each duration
ORIG_DUR*1.25	Increase durations by 25%
SCHED_DUR	Set original durations to scheduled durations

You can also specify a filter to limit the effect of the operation to a selected group of records.

To add a global edit, complete the following steps:

1. On the **Tools** tab, click **Global Edits**.
2. On the **Global Edits** dialog box, click **New**.
3. Enter the following information for the operation:
   - The name of the edit
   - The data table and field to modify
   - The expression for the operation
   - An optional filter for the operation

   To see a list of fields for the expression, click **Fields**.

   To see a list of functions for the expression, click **Functions**.

4. When the definition is complete, click **OK** to return to the **Global Edits** dialog box.

   To create a new filter for the edit from the **Global Edit Definition** dialog box, click the **ellipsis** button displayed next to the filter text box. For information about creating a filter, see Defining Filter Expressions.

To apply a global edit without saving the expression, complete the following steps:

1. Open the project or resource file for which you want to apply the temporary global edit.
2. On the **Tools** tab, click **Global Edits**.
3. On the **Global Edits** dialog box, click **New**.
4. Enter the information for the expression, but leave the name as **<temporary>**.
5. When the expression is complete, click **Apply**.

To update a global edit definition, complete the following steps:

1. On the **Tools** tab, click **Global Edits**.
2. Select the global edit you want to change, and click **Edit**.
3. In the **Global Edit Definition** dialog box, update the definition.
4. When the information for the new edit is complete, click **OK** to return to the **Global Edits** dialog box.

To copy a global edit, complete the following steps:

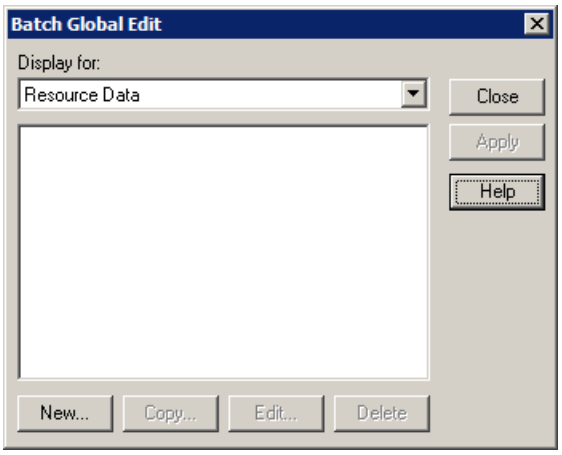
1. On the **Tools** tab, click **Global Edits**.
2. Select the global edit you want to copy, and click **Copy**.
3. In the **Global Edit Definition** dialog box, enter a new name for the definition.
4. When the information for the new edit is complete, click **OK** to return to the **Global Edits** dialog box.

To delete a global edit, complete the following steps:

1. On the **Tools** tab, click **Global Edits.**
2. Select the global edit you want to delete, and click **Delete**.
3. When Open Plan asks you to confirm the deletion, click **Yes**.

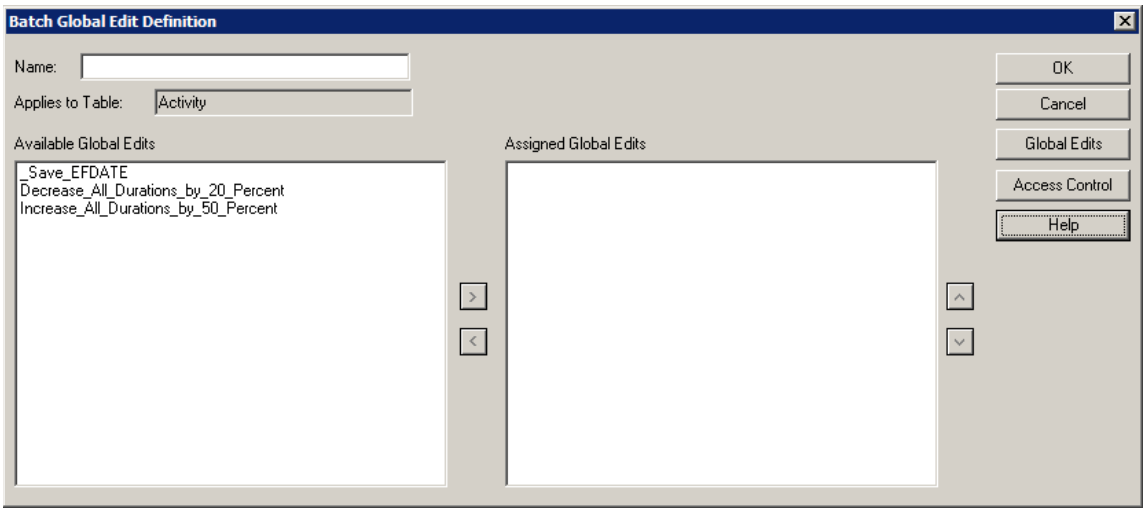
Batch Global Edits

A global edit allows you to modify multiple activities or resources with a single operation. The **Batch Global Edits** dialog box allows you to create groups of global edits (batches) that can be applied to a file in a single operation. When you click **Batch Global Edits** on the **Tools** tab, Open Plan responds with the following dialog box:



The **Batch Global Edits** dialog box displays a list of currently available definitions. The **Display For** field contains a list that segregates the batches by type of data. For example, if you select **Activity** from the list, the batches that are displayed are activity related.

When you click **New** or select a batch and click **Copy** or **Edit**, the **Batch Global Edit Definition** dialog box is displayed:



You can use this dialog box to select the global edits that the batch will use as well as the order in which the global edits are executed.

The **Batch Global Edit Definition** dialog box displays the following controls:

- **Name**—If you are creating a new definition, this field will display blank, allowing you to add a name. If you are copying an existing definition, this field will be populated with the name, preceded by copy_ *name of edit*. You can either accept the new name or replace it with a different name. If you are editing a definition, the name will display dimmed, and cannot be changed.
- **Applies to Table**—This field is populated with the table name that was previously selected, and cannot be changed on this dialog box.
- **Available Global Edits (left box)** —This field initially lists all global edits created for the selected table. To apply a global edit to the batch global edit you are defining, double-click the edit name or click the > button to move the listed edit to the **Assigned Global Edits** field.

> **Note:** To move the entire list, select the list by clicking the first name, then depress the **Shift** key and click the last name in the list, then click the **>** button. To select a random group from the list to move, press and hold the **CTRL** key and click the mouse pointer on the items to be moved to highlight, then click the **>** button.

- **Assigned Global Edits (right box)** — This field lists the global edits the batch definition will be used, and the order in which the global edits will be executed. To change the order of the list, select the item to be moved and click the up or down arrows adjacent to the list.

You can also remove any or all of the items in the **Assigned Global Edits** list by either double-clicking the item name, or clicking it to highlight and then clicking the < button. This will move the selected item or items to the **Available Global Edits** list.

- **Share this Item with Others**—By default, a batch is only accessible to the user who creates it. By clicking the **Share this Item with Others** option, it can be made available to other users.
- **Global Edits**—Clicking this button opens the **Global Edits** dialog box. You would use this option if the global edit you want to include in the batch definition does not exist or if you need to modify an existing global edit before including it in the batch.

To create a batch global edit definition, complete the following steps:

1. From the **Tools** tab, click **Batch Global Edits**.
2. From the **Display for** list on **Batch Global Edit** dialog box, select the table to which the batch global edit applies.
3. Click **New**.
4. From the **Available Global Edits** lists, select the global edits to include in the batch defintion, and click the right arrow button.
5. You can select multiple global edits to include at once by clicking **Ctrl+** **or Shift +** and selecting the global edits you want to include.
6. If the global edit you want to include is not listed, click the **Global Edits** button to display the **Global Edits** dialog box that you can use to create the appropriate global edit.
7. In the **Assigned Global Edits** list, use the up and down arrow buttons to determine the order in which the global edits are to be executed.
8. If you wish to share the batch definition with other users, select the **Share this Item with Others** option.
9. When you are finished defining the batch, click **OK**.

> **Warning:** When a batch is applied, it is first reconciled to verify all referenced global edits exist. If missing global edits are detected, a message is displayed to let you decide if you want to continue to apply the batch or not. If you choose to continue, the batch is applied and the missing global edits are removed from the definition.

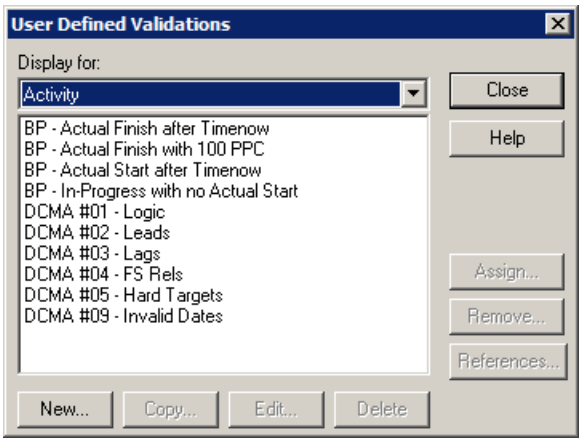
User Defined Validations

Open Plan implements User Defined Validation, allowing you to define and apply your own rules and business logic. This allows customers to ensure that data entered into schedules meets corporate and industry best practice guidelines, and reduces re-work when subsequent schedule quality audits are

performed. This set of rules or validations is applied on a per project basis, which means different organizations or customers can have different rules or standards.

> **Note:** Open Plan applies these rules each time you save information.

When you click **User Validations** on the **Tools** tab, Open Plan displays the following dialog box:



This dialog box allows you to add, assign, copy, edit, and delete rules for projects.

Before you can assign a rule to a project, make sure that you have already defined the rule or the rule that you need already exists. One of the ways that you can assign a rule to a project is through the Assign User Defined Validations dialog box. This is more useful when assigning the rule to multiple projects. The other way of assigning the rule is through the Validations Tab on the Project Properties dialog box.

Before you can delete a validation rule, you need to remove it from any project to which it is assigned.

You cannot delete a rule that has been assigned to a project. If the validation rule that you want to delete is assigned to a project, you need to remove it from that project before you can proceed with the deletion.

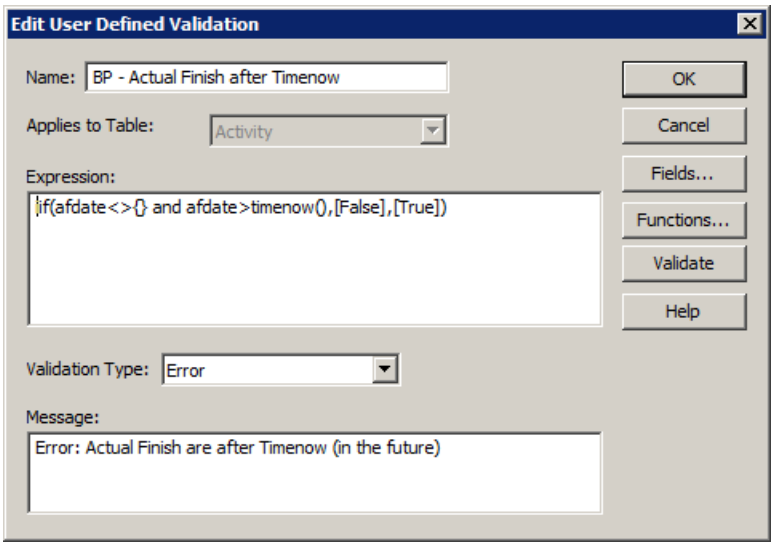
To check the list of projects or files to which the rule is assigned, click the **References** button on the User Defined Validation dialog box.

The User Defined Validations dialog box contains the following fields and buttons:

- **Display for** — This drop-down field allows you to select one of the following areas to which you want to apply the validation rule:
  - Activity
  - Relationship
  - Resource Assignment
- **New** — Click this button to add a rule. Clicking it displays the Edit User Defined Validation dialog box.
- **Copy** — Click this button to copy a selected rule. Clicking it displays the Edit User Defined Validation dialog box with values from the copied rule.
- **Edit** — Click this button to edit a selected rule. Clicking it displays the Edit User Defined Validation dialog box.
- **Delete** — Click this button to remove a selected rule from the list.
- **Close** — Click this button to exit the dialog box.

- **Assign** — Click this button to assign a selected rule to specific projects, projects with specific group access, or projects with a specific owner.
- **References** — Click this button to view the projects to which the rule is assigned. Clicking it displays the References dialog box.

Clicking **New**, **Copy**, or **Edit** on the User Defined Validations dialog box displays the Edit User Defined Validation dialog box.

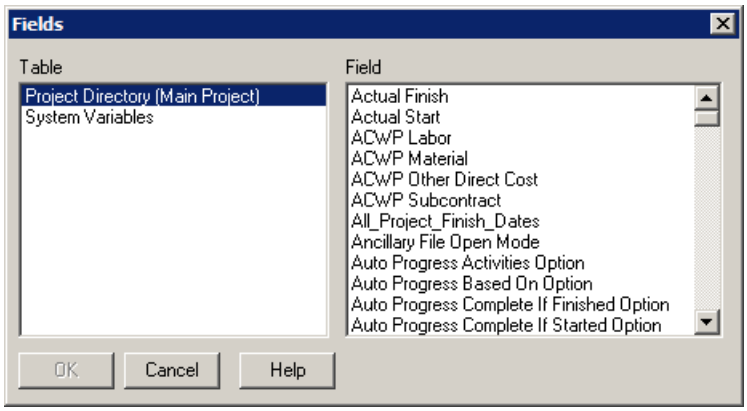


It contains the following fields and buttons:

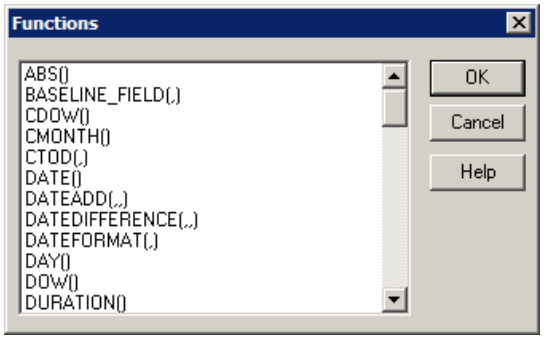
- **Name** — This field allows you to enter or update a validation name.
- **Applies to Table** — This drop-down field allows you to select an area to which you assign the validation:
- Activity
- Relationship
- Resource Assignment

If you click **Edit** on the User Defined Validation dialog box, this field is read-only.

- **Expression** — Enter the syntax or condition of the validation rule. It can be any filter, a calculated field, or a temporary filter expression. Alternatively, you can click **Fields** or **Functions** to select an existing field and/or function when creating or editing expressions.
- **Validation Type** — This drop-down list allows you to select the validation that you want Open Plan to generate:
- Warning
- Error
- **Message** — Enter or modify the message that you want to display when users fail the validation.
- **Fields** — Click this button to display the Fields dialog box, allowing you to select a field from an Open Plan data table that you can include in the expression.



- **Functions** — Click this button to display the Functions dialog box, allowing you to select a function that you can use when building the expression.



- **Validate** — Click this button to validate the expression that you entered.

To create a validation rule, complete the following steps:

1. On the **Tools** tab, click **User Validations**.
2. On the User Defined Validations dialog box, click **New**.
3. On the Edit User Defined Validations dialog box, enter or select the necessary information.
4. When you are done, click **OK**.
5. Click **Close**.

To copy or edit a validation rule, complete the following steps:

1. On the **Tools** tab, click **User Validations**.
2. From the list of rules on the User Defined Validation dialog box, select the rule that you want to copy or edit.
3. Click **Copy** or **Edit**.
4. On the Edit User Defined Validations dialog box, enter or select the necessary information.
5. When you are done, click **OK**.
6. Click **Close**.

To assign the validation rule, complete the following steps:

1. On the **Tools** tab, click **User Validations**.

2. From the list of rules on the User Defined Validation dialog box, select the rule that you need.
3. Click **Assign**.
4. On the Assign User Defined Validations dialog box select one of the following options:
   - **Assign to Specific Project**
   - **Assign to Projects with this Access Group**
   - **Assign to Projects with this Owner**
5. From the drop-down list of the selected option, select a project, group, or owner.
6. Click **OK**.
7. Click **Close**.

To remove the assigned rule from the project, complete the following steps:

1. Open a project.
2. On the **Processes** tab, click **Properties**.
3. On the Project Properties dialog box, click the **Validations** tab.
4. From the Assigned User Defined Validations list, select the assigned rule that you want to remove and then click  . The selected rule displays in the Available User Defined Validations list.
5. Click **OK**.

To delete a validation rule, complete the following steps:

1. On the **Tools** tab, click User Defined Validations.
2. From the list of rules on the User Defined Validation dialog box, select the rule that you want to delete.
3. Click **Delete**.
4. On the confirmation dialog box, click **Yes**.

Calculated Fields

In Open Plan, the data fields displayed in a view are typically one of three types:

- Normal data entry fields — for example, an activity ID or description that can be updated by the user at any time
- Fields generated by Open Plan processes — for example, the early start and finish dates calculated by time analysis.
- Calculated fields defined by the user.

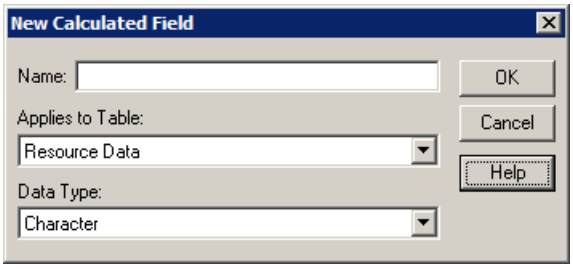
User-defined calculated fields allow you to calculate and display data not stored in the standard versions of the project database tables. With calculated fields, you can extend the flexibility of any view by displaying data that is the result of a custom calculation. Once defined, a calculated field can be treated as any other type of field. For example, you can display that field as a column in a spreadsheet view or in an activity box, just as you can any standard field. You can also include calculated fields in custom filter and sort expressions.

Note: Open Plan does not store the results of calculated fields in the project database. Only the definition of the calculation is saved — the results are recalculated each time the field is required for display. Third-party applications that query the database cannot, therefore, access the results of calculated fields. You can, however, use the global edit feature to place the contents of a calculated field into a user-defined field.

Defining Calculated Fields

User-defined calculated fields allow you to calculate and display data not stored in the standard versions of the project database tables. With calculated fields, you can extend the flexibility of any view by displaying data that is the result of a custom calculation.

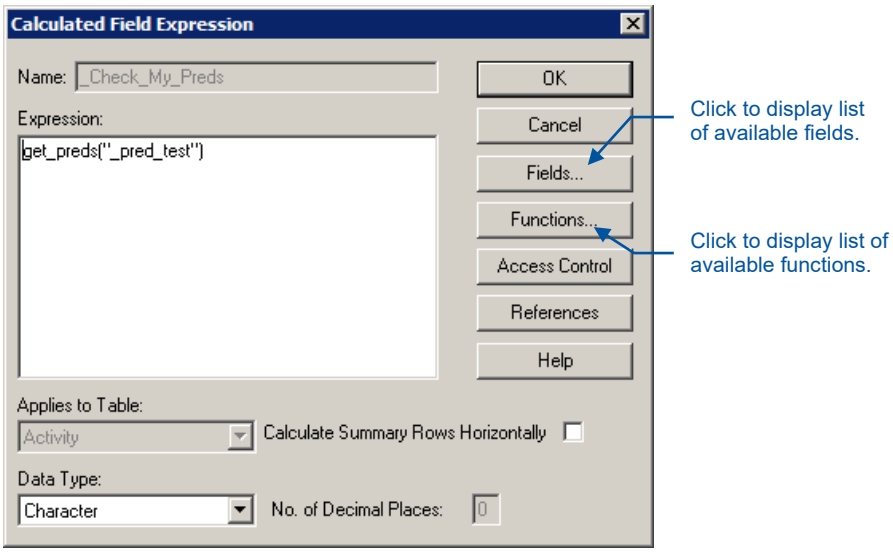
You can use the **Calculated Fields** command from the **Tools** tab to define a new calculated field at any time. When you define a new calculated field, Open Plan displays a dialog box in which you can enter a name for the field, the data table to which you want to associate the field, and the type of result produced by the field.



You can define a new calculated field to produce one of the following types of results:

- **Character** — Returns a character string to the calculated field
- **Date** — Returns a date to the calculated field
- **Decimal** — Returns a decimal number (for example, 7.89) to the calculated field
- **Duration** — Returns a duration to the calculated field
- **Finish Date** — Returns an activity finish date to the calculated field
- **Integer** — Returns a whole number to the calculated field
- **Logical** — Returns True or False to the calculated field

Once you provide a name, data table association, and result type for the calculated field, Open Plan allows you to enter the expression for the field.



Use the **Fields** and **Functions** commands to display lists of items that you can insert into the expression you are editing.

The **Calculate Summary Rows Horizontally** option applies to calculated fields appearing in summary rows of spreadsheets using grouping. If you select this option, Open Plan calculates the value of the field based on the values of the summarized row, effectively disabling the rolling up of the field. If you leave the option unselected, Open Plan simply rolls up the values appearing in the column. For example, assume that you have defined a calculated field that compares two activity dates. When this calculated field appears in a summary row, it is likely that you want to display a value based on summarized dates rather than one that is the simple aggregation of the calculated values appearing in the column.

You can also change the table and data type assignment for the expression. For expressions that result in decimal data types, you can specify the number of decimal places in the result.

> **Attention:** For a complete description of the operators and functions available for calculated fields, see The Elements of Calculated Field Expressions and Calculated Field Functions References.

You can define calculated fields for both project and resource files. By default, Open Plan stores the calculated field with the project or resource file that is open when you define the field. (Open Plan may also store the field with the current view if the field is used by the view.)

After you have defined a calculated field, you can select that field as a column in a spreadsheet view or display it in an activity box just as you can any standard field. You can also include calculated fields in custom filter and sort expressions.

The expression for a global edit uses the same elements (constants, field names, and functions) as calculated field definitions.

To add a calculated field, complete the following steps:

1. On the **Tools** tab, click **Calculated Fields**.
2. On the **Calculated Fields** dialog box, click **New**.
3. Enter the following information:
   - Field name
   - The data table to which you want to associate the field

- The data type of the field. The data type of the field must match the result produced by the calculated field expression.

4. When the information for the field is complete, click **OK**.
5. In the **Calculated Field Expression** dialog box, enter the expression for the calculation.

   To see a list of fields for the expression, click **Fields**.

   To see a list of functions for the expression, click **Functions**.
6. When the expression is complete, click **OK** to return to the **Calculated Fields** dialog box.

To edit a calculated field, complete the following steps:

1. On the **Tools** tab, click **Calculated Fields**.
2. Select the calculated field you want to change, and click **Edit**.

   If you do not see the appropriate calculated field listed in the dialog box, use the **Display For** list to select another data table.
3. In the **Calculated Field Expression** dialog box, update the expression for the field.
4. When the information for the expression is complete, click **OK** to return to the **Calculated Fields** dialog box.

To copy a calculated field, complete the following steps:

1. On the **Tools** tab, click **Calculated Fields**.
2. Select the calculated field you want to copy, and click **Copy**.

   If you do not see the appropriate calculated field listed in the dialog box, use the **Display For** list to select another data table.
3. In the **Calculated Field Expression** dialog box, enter a new name for the field.
4. When the information for the new expression is complete, click **OK** to return to the **Calculated Fields** dialog box.

To delete a calculated field, complete the following steps:

1. On the **Tools** tab, click **Calculated Fields**.
2. Select the calculated field you want to delete, and click **Delete**.

   If you do not see the appropriate calculated field listed in the dialog box, use the **Display For** list to select another data table.
3. When Open Plan asks you to confirm the deletion, click **Yes**.

The Elements of Calculated Field Expressions

An expression defining a calculated field can include the following elements:

- Constants
- Field names
- Functions
- Other calculated fields
- Mathematical operators

- Character operators
- Duration operators
- Relational operators
- Logical operators
- User-Defined Variables

This section discusses each of the elements of a calculated field expression with the exception of functions, which are discussed in the following section.

Constants

You can include text, date, numeric, and logical constants in a calculated field expression according to the following guidelines:

Constant	Description
Text	All text must be enclosed in either single or double quotes. For example: ▪ “Programmers” ▪ ‘Phase I’
Dates	Dates must be enclosed in curly brackets ({ and }) and can be expressed in any valid date format. For example: ▪ {01JAN01} ▪ {12/01/01}
Durations	Durations must be enclosed between pipe characters (|) and can be expressed in any valid duration format. For example: ▪ |4h| ▪ |3.5d|
Numerics	Numeric constants can include any positive or negative value and can appear with or without decimal places. For example: ▪ 1 ▪ 320000 ▪ 12.1 ▪ –123.78
Logical Values	Valid logical constants are as follows: ▪ [TRUE] ▪ [FALSE]
Enumerated Values	Values for enumerated fields (that is, fields with a limited number of valid values) must be enclosed between square brackets ([and]). For example, an expression can include references to any of the possible values for the Act_type field:

Constant	Description
	▪ [ASAP] ▪ [ALAP] ▪ [Start Milestone] ▪ [Finish Milestone] ▪ [Discontinuous] ▪ [Subproject] ▪ [Hammock] ▪ [Effort Driven] ▪ [External Subproject] You can include text, date, numeric, and logical constants in a calculated field or global edit according to the following guidelines: Open Plan recognizes the following enumeration types: ▪ ACTS — Activity Status ▪ ACTT — Activity Type ▪ BOOL — Boolean ▪ CRIT — Critical Flag ▪ CURV — Curve ▪ DIST— Risk ▪ EVTE — Earned Value Technique ▪ LOGI — Activity Logic Flag ▪ PRJS — Project Status ▪ PROG — Progress ▪ RELT — Relationship type ▪ RESC — Resource class ▪ RSCL — Resource scheduling type ▪ TARG — Target type

When including a field in an expression for a calculated field, you must enter the name of the field and not the descriptive name that is displayed in spreadsheet column headings. For example, if you want to define a calculated field expression that references the field containing early start dates, you must identify the field as ESDATE, not Early Start.

If you need to refer to fields stored in tables other than the primary data table, use the name of the linking field, followed by a period (.) and the field name in the linked table. For example:

- **C1.DESCRIPTION** — the description for a code stored in the C1 field.
- **RES_ID.DESCRIPTION** — the description for a resource ID stored in the Res_ID field.

Note: If you display the list of available tables and fields by clicking **Fields** on the **Calculated Field Expression** dialog box, linking fields are distinguished by a double chevron (») next to the field name.

Other Calculated Fields

Expressions defining calculated fields can include references to previously defined calculated fields.

Warning: A calculated field cannot include a reference to itself.

Mathematical Operators

You can include the following mathematical operators in a calculated field expression:

- Add (+)
- Subtract (-)
- Multiply (*)
- Divide (/)
- Group (( ))
- Exponentiate (^ or **)

Expressions containing mathematical operators are evaluated according to the normal rules of precedence:

- Grouping operations are performed before multiplication and division operations.
- Multiplication and division operations are performed before addition and subtraction operations.

Character Operators

You can include the following character operators in a calculated field expression:

- Concatenate (+)
- Is contained in ($)

The **Is contained in** operator returns a logical result.

Duration Operators

You can include the following duration operators in a calculated field expression:

- Add (+)
- Subtract (-)
- Multiply (*)
- Divide (/)

You can use these operators in expressions involving durations as follows:

Operation	Result
Duration + Duration	Duration
Duration + Date	Date
Duration – Duration	Duration
Date – Duration	Date
Duration / Duration	Decimal
Duration / Number	Duration
Duration * Number	Duration

Relational Operators

Expressions for calculated fields can include the following relational operators:

- Equal to (=)
- Not equal to (<>)
- Greater than (>)
- Greater than or equal to (>=)
- Less than (<)
- Less than or equal to (<=)
- Contained in ($)

Logical Operators

You can include the following logical operators in a calculated field expression:

- AND
- NOT
- OR
- AND NOT
- Group (())

User-Defined Variables

When Open Plan parses a calculated field (CF) expression, it optimizes the expression internally by looking for multiple occurrences of a subexpression and replacing these occurrences with an internal variable. For example, let's say you have defined another CF called "TimeNowPlusFive":

Timenow() + |5d|

You then use this CF in another CF:

IIF(EFDATE > TimeNowPlusFive, TimeNowPlusFive, EFDATE)

Internally, Open Plan will evaluate the TimeNowPlusFive CF only once for a particular cell, and use the result of this evaluation each time the same subexpression is encountered within the CF. In order for Open Plan to perform this optimization, each occurrence of the subexpression must be identical in terms of capitalization and spacing (for example, “EFDATE > ASDATE” is not the same as “efdate > asdate”). This optimization is performed automatically and assures efficient evaluation provided that the user has been consistent as explained above in the use of subexpressions that occur multiple times.

Open Plan also provides a second, more pro-active way for the user to write a CF expression by using variables. Two advantages of using this second option are that 1) the expression becomes much more readable and easier to maintain, since redefining the variable needs to be done in only one place, and 2) the parsing of a very complicated expression that contains variables takes less processing time than the same expression using repeated subexpressions.

These are the rules for using variables within a CF:

- Variables are defined above the main CF expression and are contained within a BEGIN VARIABLES/END VARIABLES block.
- Each variable definition must be on its own line.
- The result type of a variable definition (for example, character, duration, integer, decimal, date, finish date, and logical) is determined automatically by Open Plan.
- Variable names should not contain spaces or be identical to field names, function names, or other items used within Open Plan where ambiguity may occur. For example, a variable named “123” could be confused with a numeric value. “act_id” would be confused with the Open Plan field of the same name.
- Variables are not case sensitive.
- The format for a variable definition is: <variable name> = <variable expression>.
- The definition of a variable expression may reference other variables, but referenced variables must have been previously defined.

Here is an example of a CF containing a variable block. Note that the previously defined variable X is included in the definition of variable Y:

```
BEGIN VARIABLES
X = DATEDIFFERENCE(ASDATE, TIMENOW())
Y = IIF(x<=|2d|, IIF(x<=|1d|, 2, 1), -1)
END VARIABLES


IIF(Y>0, “OK”, “Warning”)
```

Calculated Field Functions Reference

Open Plan provides a number of functions that can appear in calculated field expressions. This section lists the available functions followed by a detailed description of each function.

ABS()	GET_NOTE()	MONTH()
BASELINE_FIELD()	GET_PREDS()	NEWLINE()
CDOW()	GET_RELATED()	NUMBER_FORMAT()
CMONTH()	GET_RISKS()	OCCURS()

CTOD()
DATE()
DATEADD()
DATEDIFFERENCE()
DATEFORMAT()
DAY()
DOW()
DURATION()
EVAL()
FAIL_EVALUATE()
FISCALPERIOD()
FORMAT_HEADING_ITEM()
GET_ASSGNS()
GET_CHILDREN()
GET_COSTS()
GET_FIELD()
GET_FIRST_RECORD_ IN_SUMMARY()
GET_SUCCS()
GET_USAGES()
GO_MONTH()
HAS_NOTE()
IIF()
INLIST()
INSTR()
LEFT()
LEN()
LEVEL()
LOCAL()
LOWER()
LTRIM()
MAX()
MID()
MIN()
PARENT()
RECORD_NUMBER()
RIGHT()
ROUND()
SPACE()
SQRT()
STR()
STRTRAN()
STUFF()
SUBSTR()
TIMENOW()
TRIM()
UPPER()
USER_ID()
VAL()
YEAR()

ABS()

Purpose	Returns the absolute value of a numeric variable.
Data type	Decimal or integer
Syntax	ABS(<value>)
Where	<value> is a numeric variable
Example	If y contains the value [4] or [-4], the statement: ABS(y) returns the value [4].

BASELINE_FIELD()

Purpose	Returns a field from the baseline directory table (OPP_BAS) record associated with the first, second, or third selected project baseline. Note that this function may be used with any table whose records can be associated with a project. This includes the Project Directory, Activity, Relationship, Assignment, Cost, Risk, and Subproject tables.
Data Type	Native type of field requested.
Syntax	BASELINE_FIELD(<SelectedBaselineIndex>, <BaselineFieldName>)
Where	▪ SelectedBaselineIndex is a number between 1 and 3 indicating from which of the selected baseline objects the data is to be retrieved (note that “0” is also accepted, and will return the same results as “1”).

- BaselineFieldName is a single or double quote delimited field name indicating the field to be returned. Fields that might be of interest include “BASETYPE” (0 = early, 1 = late, 2 = schedule), “BAS_ID” (baseline name), and “DESCRIPTION."

Example BASELINE_FIELD(1, "DESCRIPTION")

If the baseline selected at index 1 for the current project is “PMB,"the returned string would be “Performance Measurement Baseline."

CDOW()

Purpose Returns the full day of the week (for example, Tuesday).

Data type Character

Syntax CDOW(<date>)

Where <date> is a date variable or user-entered date

Example Assuming that ESDATE is 07OCT04 (Tuesday), the statement:

CDOW(ESDATE)

returns the value **Tuesday**.

CMONTH()

Purpose Returns the full month of the year (for example, October).

Data type Character

Syntax CMONTH(<date>)

Where <date> is a date variable or user-entered date

Example Assuming ESDATE is 07OCT04, the statement:

MONTH(ESDATE)

returns the value **October**.

CTOD()

Purpose The Calendar to Date function converts a character string to a value with a DATE data type. This is useful in calculated field expressions where we want to coerce the result of an operation to type DATE.

Data type Date

Syntax CTOD (<String Expression>)

Where <String Expression> is a quotation-mark delimited string with a valid date format.

Example CTOD(STR(USER_NUM01) + "/" + STR(USER_NUM02) + STR(YEAR(TIMENOW())))

If USER_NUM01 = 12, USER_NUM02 = 31, and Time Now = 1/1/2006, the date result returned will be {12/31/2006}.

DATE()

Purpose Returns the current date.

Data type Date

Syntax DATE ( )

Example Assuming the current date is October 4, 2004, the statement:

DATE( )

returns **04OCT04**.

DATEADD()

Purpose Allows you to add a duration to a date and returns a date.

Data type Date

Syntax DATEADD (<start date>, <duration>, <calendar >)

Where <start date> is a date field or user-entered date

<duration> is a duration field or user-entered value

<calendar name> is the name of the calendar to be used in the calculation. This parameter is optional; if no calendar is specified, Open Plan performs the calculation using the appropriate calendar as follows:

- If a valid calendar is attached to the project, the function uses the calendar named < Default >.
- If no calendar is attached to the project, a 40-hour week is assumed.

> **Note:** If the calculated field or global edit is based on the activity table, you can also specify the fieldname **CLH_ID** to have Open Plan use the calendar assigned to the specific activity. However, if the global edit is based on the assignment table, specify **ID.CLH_ID** instead of **CLH_ID**.

Example Assuming that ESDATE is 04OCT04, the statement:

DATEADD (ESDATE,|2d|, “CAL1”)

returns **05OCT04** based on calculations using the calendar named CAL1.

DATEDIFFERENCE()

Purpose Returns the difference between two dates as a duration.

Data type Duration

Syntax DATEDIFFERENCE (<date1>, <date2>, <calendar>)

Where <date1> is a date field or user-entered date

<date2> is a date field or user-entered date

<calendar name> is the name of the calendar to be used in the calculation. This parameter is optional; if no calendar is specified, Open Plan uses the appropriate calendar as follows:

- If a valid calendar is attached to the project, the function uses the calendar named < Default >.
- If no calendar is attached to the project, a 40-hour week is assumed.

> **Note:** To perform the calculation using the calendar currently assigned to the activity or resource, set the calendar parameter to Calendar. To determine working days based on a specific calendar in the calendar file, set the calendar parameter to the name of the calendar.

Example Assuming that ESDATE is 07OCT04 and EFDATE is 08OCT04, the statement:

DATEDIFFERENCE (ESDATE, EFDATE, “CAL1”)

returns **2d** based on calculations using the calendar named CAL1.

DATEFORMAT()

Purpose Returns a date in a specified format.

Data type Character

Syntax DATEFORMAT (<date field>, <format string>)

Where <date field> is a date field or user-entered date

<format string> is the date format

> **Note:** DATEFORMAT() uses the same date formats as those used when specifying dates on the **Date Scale Preferences** dialog box.

Example Assuming that ESDATE is October 7, 2004, the statement:

DATEFORMAT (ESDATE, “%D%A%Y”)

returns **07OCT04**.

> **Attention:** For a description of date format strings in Open Plan, see **Views and Reports » Tabs and Dialog Boxes » Date Formats Dialog Box** in the *Deltek Open Plan online help*. This information is also available in Date Formats in Open Plan.

DAY()

Purpose Returns a numeric value 1 through 31 for the day of the month.

Data type Integer

Syntax DAY(<date>)

Where	<date> is a date variable or user-entered date
Example	Assuming that ESDATE is 04OCT04, the statement: DAY(ESDATE) returns **4**.

DOW()

Purpose	Returns a numeric value 1 through 7 for the day of the week. The actual number returned depends on settings in the Open Plan .ini file for the first day of the week.
Data type	Integer
Syntax	DOW(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming that ESDATE is 07OCT04 (Tuesday), the statement: DOW(ESDATE) returns **3**.

DURATION()

Purpose	Returns the number of minutes corresponding to a given duration
Data type	Integer
Syntax	DURATION(<Duration Value>)
Where	<Duration Value> is a duration variable or user-entered duration
Example	Assuming an 8-hour workday, the statement: Duration(|2d|) returns **960 (8 hours x 2 days x 60 minutes/hour)**.

EVAL()

Purpose	Evaluates its string expression argument as a dynamic calculated field rather than as a string literal.
Data type	Character
Syntax	EVAL(<exp>)
Where	<exp> evaluates a string expression argument to be calculated as a dynamic calculated field rather than as a string literal.
Example	Assuming that the value of USER_CHR01 is “ESDATE” and USER_CHR02 is “EFDATE” and that ESDATE is 04OCT04 and EFDATE is 06OCT04, the statement: EVAL('CTOD(USER_CHR02) - CTOD(USER_CHR01)') returns a duration of **2d**.

FAIL_EVALUATE()

Purpose	Test for a condition that normally would cause the calculated field to return either a blank value or an invalid result (the latter would keep the remainder of the expression from being evaluated).
Data type	Boolean
Syntax	FAIL_EVALUATE(<String Expression>)
Where	<String Expression> is an expression in string format.
Example	FAIL_EVALUATE("C25") If there is no code file assigned at index 25, this calculated field will return True. Note that if we substituted the expression "C25 IS_EMPTY" as an alternative to using FAIL_EVALUATE, the returned value would be True whether C25 does not exist or C25 exists but is blank.

FISCALPERIOD()

Purpose	This function returns the user-defined label for a specific fiscal reporting period for a specific date.
Data type	Character
Syntax	FISCALPERIOD(<Date>,<Reporting Calendar>)
Where	<Date> is the search date to be located within a fiscal period. < Reporting Calendar> is an optional parameter (defaulting to the project reporting calendar).
Example	FISCALPERIOD(ESDATE) FISCALPERIOD(ESDATE, "REPORTING_CALENDAR_NAME") The function returns the LABEL field from the reporting calendar that includes the supplied date. For example, given the following DATE and LABEL fields: DATE LABEL 01/26/2001 Jan 2001 02/23/2001 Feb 2001 03/23/2001 Mar 2001

FORMAT_HEADING_ITEM()

Purpose	Creates custom grouping heading with summary rows (i.e., the summary type for the grouping row is marked as "Heading").
Data type	Character
Syntax	FORMAT_HEADING_ITEM(<Expression>,<WidthInCharUnits>, <Wordwrap>,<Summarize>)
Where	<Expression> is the character string to be formatted, <WidthInCharUnits> is the width of the expression in character units, <Wordwrap> is the numeric value 0 (no word wrap) or 1 (word wrap active), and <Summarize> is an OPTIONAL argument indicating whether <Expression> (if not a constant) should NOT be summarized (numeric value of 0), or should be summarized (numeric value of 1). The default value is 1.
Example	Given the following calculated field expression:

FORMAT_HEADING_ITEM(C2 + " - ," 20,0,0) +
FORMAT_HEADING_ITEM(C2.DESCRIPTION, 40, 1,0) +
FORMAT_HEADING_ITEM(C2.< Default >,20,1,0)

The output in a spreadsheet grouping on C2, using the above calculated field as the Heading Field would look similar to this:

	Activity ID	Activity Desc.		Orig Dur	Early Start
-	1.2.1 -	1.2.1 SYSTEM ENGINEERING	These activities were delayed due to various technical problems.		21Feb05
	1.16.02	Project Final Review		3d	21Feb05

GET_ASSGNS()

Purpose Returns assignment data for an activity.

Data type Character

Syntax GET_ASSGNS(<fieldname1>[|<fieldname2>…])

Where <fieldname1>, <fieldname2> are names of fields in the Resource Assignment table

> **Note:** Because the GET_ASSGNS() function works within a specified data table, it is important that the table portion of the fieldname be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that the resource assignments for the activity are ENG (a level 1 resource pool) and TECH.MARY (a level 2 resource), the statement:

GET_ASSGNS(“RES_ID|RES_LEVEL”)

It then returns **ENG,1.00;TECH.MARY,2.00**.

GET_CHILDREN()

Purpose Returns a list of fields from the immediate children of the current hierarchical record (activities, resources, and codes).

Data type Character

Syntax GET_CHILDREN(<FieldList>)

Where FieldList is a pipe-delimited string containing the list of fields to be returned. Fields are separated by commas, and records are separated by semi-colons in the returned data.

Example GET_CHILDREN("ACT_ID|DESCRIPTION")

Given an activity "1.01,"with children, "1.01.01" and "1.01.02,"the returned string would be the following:

"1.01.01,First child of 1.01;1.01.02,Second child of 1.01"

GET_COSTS()

Purpose: This function returns cost records for an activity.

Data type: Character

Syntax: GET_COSTS(<fieldname1>[| <fieldname2>…])

Where: <fieldname1>, <fieldname2> are names of fields in the Cost table

> **Note:** Because the GET_COSTS() function works within a specified data table, it is important that the table portion of the fieldname be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example: Assuming that there are two cost records for the activity, that the actual costs are 1,000 and 4,000, and that the actual quantities are 1 and 2, the statement:

GET_COSTS("ACWP_CST|ACWP_QTY")

It then returns **1000.00,1.00;4000.00,2.00.**

GET_FIELD()

Purpose: This function allows users to display data from other tables or data from other records in the same table.

Data type: Character

Syntax: GET_FIELD (<TableType>, <UniqueID>,<FieldName>)

Where:

- TableType = "Activity","Resource","Calendar","B1","B2","B3" (Selected Baseline Activity Tables), "ProjDir","CodeDir","ResDir","CalDir".

  > **Note:** TableType = "Resource" will pull data from the "Resource Data" table.

- UniqueID = a constant, expression, or field name that identifies a look-up key for the table type. Note that the only acceptable table types are those that have unique identifiers (for example, the activity ID on the activity table, or the project name on the Project Directory).
- FieldName = the field in Table TableType whose value we want returned. Note that the field name MUST be enclosed in quotation marks. Otherwise, the VALUE of the field, not the field name itself, will be passed as the argument.

Example: GET_FIELD("C2,"PARENT(C2), "DESCRIPTION")

On the Activity table, if the current Activity has a code 2 value of "1.2.1.3,"the returned string would be "1.2.1. System Engineering."

GET_FIRST_RECORD_IN_SUMMARY()

Purpose	This function returns the value for the requested field. For a grouping summary row, it returns the value of the requested field for the first child record.
Data type	Same data type as input field
Syntax	GET_FIRST_RECORD_IN_SUMMARY (<TableType>, <UniqueID>,<FieldName>)
Where	FieldName = the field in Table TableType whose value we want returned. Note that the field name MUST be enclosed in quotation marks. Otherwise, the VALUE of the field, not the field name itself, will be passed as the argument.
Example	The purpose of the GET_FIRST_RECORD_IN_SUMMARY function is to allow the user to construct a calculated field result in a summary row that uses a break column value rather than a summary value. For example, given the following filter in a barchart (with a visibility setting of "All levels of Rollup and Detail"), with a spreadsheet pane grouped on TOTALFLOAT: TOTALFLOAT > |2d| The filter would evaluate to TRUE for all child rows of groupings where the TOTALFLOAT break-on value is greater than |2d|. However, the filter would evaluate to FALSE on the grouping summary rows since the value of the TOTALFLOAT field on the summary row is undefined (durations cannot be summarized). Even for a numeric or date field (which can be summarized), the filter would have to be written differently to account for a summarized value on a summary row. To work around this problem, the following syntax will give us the result we want: GET_FIRST_RECORD_IN_SUMMARY("TOTALFLOAT") > |2d|

GET_NOTE()

Purpose	Returns an activity, resource, or code note.
Data type	Character
Syntax	GET_NOTE(<category>)
Where	<category> is the name of a note category. This field is optional. If you leave <category> blank, the function assumes the default category.
Example	Assuming that a category <Document> has been set up for an activity table, the statement: GET_NOTE(“Document”) returns the Document category note about the activity.
Example	Assuming that you want to use the default category for an activity note, the statement: GET_NOTE(“”) returns the default category note about the activity.

GET_PREDS()

Purpose	Returns predecessor data for an activity.
Data type	Character
Syntax	GET_PREDS(<fieldname1>[|<fieldname2>…])

Where <fieldname1>, <fieldname2> are names of fields in the Relationship table

> **Note:** Because the GET_PREDS() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that the predecessor to an activity is A100 and that the relationship type is Finish to Start, the statement:

GET_PREDS(“PRED_ACT_ID|REL_TYPE”)

It then returns **A100,Finish to Start**.

GET_RELATED()

Purpose Returns a list of fields for a each member of a child collection related to the main collection. This is basically a more generic version of the more specialized functions GET_PREDS(), GET_SUCCS(), GET_ASSGNS(), etc.

Data type String

Syntax GET_RELATED(<CollectionType>, <FieldList>)

Where < CollectionType > is a quotation-mark delimited identifier.

Main Collection = "Activity"

- “A01”,"A02”,"A03" (Selected Baseline Activity Collection)
- "ASG" (assignment collection)
- "CST" (cost collection)
- "PRD" (predecessor collection)
- "RSK" (risk collection)
- "SUB" (subproject collection)
- "SUC" (successor collection)
- "U01”,"U02”,"U03" (Selected Baseline Usage Collection)
- "USE" (usage collection)

Main Collection = "Baseline Activity"

- "SUB" (subproject collection)
- "USE" or "BSU" (baseline usage)

Main Collection = "Resource Data"

- "PSU" (Project Summary Usage)
- "RSL" (Resource Escalations)

- "SKL" (All skill assignments for the given resource record)
- "SKR" (All skill assignments for the given skill record)

<FieldList> is a pipe-delimited string containing the list of fields to be returned. Fields are separated by commas, and records are separated by semi-colons in the returned data.

Example — Assuming that we are displaying an activity spreadsheet, and that activity “A” is the current activity, the expression below will return the activity ID and early start fields for the record that matches the activity on the first selected baseline.

GET_RELATED(“A01”,“ACT_ID|ESDATE”) returns “A,12Oct04”

GET_RELATED_COUNT()

Purpose — Returns a record count. The two parameters for the function are the related collection identifier and an optional filter. You use this primarily with activity and baseline activity tables.

Data type — String

Syntax — GET_RELATED_COUNT(RELATED_COLLECTION_IDENTIFIER, <FILTER_ON_RELATED_COLLECTION>)

Where — You can use the function with one of the following related collection identifiers:

Main Collection = "Activity"

- “ASG” (assignment collection)
- “CHILD” (child collection)
- “CST” (cost collection)
- “PRD” (predecessor collection)
- “RSK” (risk collection)
- “STP” (step collection)
- “SUB” (subproject collection)
- “SUC” (successor collection)
- “USE” (usage collection)

Main Collection = "Baseline Activity"

- “CHILD” (child collection)
- “USE” (usage collection)

Example — You might want a count of all children for a subproject activity (GET_RELATED_COUNT("CHILD")), or you may want to get a count of all predecessors for a particular activity with a relationship lag of 0 (GET_RELATED_COUNT(PRD", "REL_LAG = |0|").

GET_RISKS()

Purpose — Returns risk data for an activity.

Data type Character

Syntax GET_RISKS(<fieldname1>[| <fieldname2>…]

Where <fieldname1>, <fieldname2> are names of fields in the Risk table

> **Note:** The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks.
>
> The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example The statement:

GET_RISKS(“ESDATE1”)

returns each early start date calculated during risk analysis.

GET_SUCCS()

Purpose Returns successor data for an activity.

Data type Character

Syntax GET_SUCCS(<fieldname1>[|<fieldname2>…])

Where <fieldname1>, <fieldname2> are names of fields in the Relationship table

> **Note:** Because the GET_SUCCS() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that the successor to an activity is A200 and that the relationship type is Finish to Start, the statement:

GET_SUCCS(“SUCC_ACT_ID|REL_TYPE”)

It then returns **A200,Finish to Start**.

GET_USAGES()

Purpose This function returns use records for an activity.

Data type Character

Syntax GET_USAGES(<fieldname1>[| < fieldname2>…]

Where <fieldname1>, <fieldname2> are names of fields in the Use table

> **Note:** Because the GET_USAGES() function works within a specified data table, it is important that the table portion of the fieldname (added when you select Fields via the Fields button) be removed when using this function.
>
> The pipe symbol (|) is used as a separator between the desired fields. The entire expression within the set of () must be enclosed by quotation marks. The text string returned for this function uses commas (,) to separate the different data items within each record and uses semi-colons (;) to separate each record.

Example Assuming that an activity having a single use record shows that 24 units of the resource ENG have been used, the statement:

GET_USAGES("RES_ID|RES_USED")

It then returns **ENG,24.**

GO_MONTH()

Purpose Returns a date that is the specified number of months before or after a specified date.

Data type Date

Syntax GO_MONTH(<date>,<integer>)

Where <date> is a date variable or user-entered date

<integer> is a positive or negative integer

Example Assuming that ESDATE is 04OCT04, the statement:

GO_MONTH(ESDATE,-2)

Returns **04AUG04**.

HAS_NOTE()

Purpose Returns a logical value indicating whether or not a note is attached to an object.

Data type Logical

Syntax HAS_NOTE(<string>)

The string is optional.

Example Assuming that an activity has a note attached, the statement:

HAS_NOTE( )

returns **True**

Example Assuming that an activity has a note in the category "Scope," the statement:

HAS_NOTE("Scope")

returns **True**

IIF()

Purpose Provides a means of defining conditional processing in a report.

Data type	Any
Syntax	IIF(<logicexp>,<iftrue>,<iffalse>)
Where	<logicexp> is a logical expression
	<iftrue> is any valid expression
	<iffalse> is any valid expression
Operation	If <logicexp> evaluates to true, the calculated field or global edit is set to the value of <iftrue>; otherwise, the calculated field or global edit is set to the value of <iffalse>.
Example	Assuming that ESDATE is 19JUN01, the statement:
	IIF(ESDATE>{01JUL01}, “Underway”, “Planned”)
	returns **Planned**.

INLIST()

Purpose	Returns a logical value indicating whether or not a value is included in a list.
Data type	Logical
Syntax	INLIST(<search>,<value1>, <value2>, ...)
Where	<search> is a variable or constant for which the function searches in the list
	<value1> is a variable or constant
	<value2> is a variable or constant

> **Note:** Open Plan interprets a character argument as a comma-delimited list of items.

Example	Assuming that the early start date month is June, the statement:
	INLIST(MONTH(ESDATE),1,4,7,10)
	returns **False**.
Example	Assuming that “CHRIS” is the local portion of a resource code, the statement:
	INLIST(“CHRIS," GET_ASSGNS(“LOCAL(RES_ID)”))
	returns **True** for the activities to which CHRIS is assigned.

INSTR()

Purpose	Returns the position of the first occurrence of one string from within another string. If the string that is sought is not found, this function returns 0.
Data type	Integer
Syntax	INSTR(<start>,<string1>,<string2>)

Where <start> is an integer that sets the starting position for the search

<string1> is the string expression being searched

<string2> is the string expression being sought

> **Note:** The string expressions are case sensitive.

Example Assuming the following string expression to be searched:

SITE COORDINATION AND DESIGN

And the following string being sought:

COORDINATION

The statement:

INSTR(1,"SITE COORDINATION AND DESIGN","COORDINATION")

returns **6**.

LEFT()

Purpose Returns a specified number of leftmost characters in a string.

Data type Character

Syntax LEFT(<string>,<int>)

Where <string> is the string expression being searched

<int> is the number of characters to be returned

> **Note:** The string expressions are case sensitive.

Example The statement:

LEFT("SITE COORDINATION AND DESIGN", 4)

returns **SITE**.

LEN()

Purpose Returns a numeric value that is the length of a variable or constant.

Data type Integer

Syntax LEN(<data>)

Where <data> is a character, date, or numeric variable, or a character or numeric constant

Example The statement:

LEN("Dig Hole")

returns **8**.

Example Assuming the DESCRIPTION field is 30 characters long and contains the string "EXCAVATION," then the statement:

LEN(DESCRIPTION)

returns **10**. Note that it is irrelevant that the field width is 30 characters.

LEVEL()

Purpose	Returns the hierarchical level of an ID or code.
Data type	Integer
Syntax	LEVEL(<ID>)
Where	<ID> is a character variable or constant
Example	Assuming that a code file is assigned to the project in position C1, then the statement: LEVEL(C1) returns the level of C1 in the code structure.

LOCAL()

Purpose	Returns the local portion of an ID or code (that is, the portion of the ID or code that is not shared by other children of the same parent).
Data type	Character
Syntax	LOCAL(<ID>, <level>)
Where	<ID> is a character variable or constant <level> is optional, specifying the level at which you want the local portion
Example	Assuming that a code file is assigned to the project in position C1, the statement: LOCAL(C1) returns the local (rightmost) portion of C1.

LOWER()

Purpose	Converts alphabetic characters from uppercase to lowercase.
Data type	Character
Syntax	LOWER(<string>)
Where	<string> is a character variable or constant
Example	The statement: LOWER(“Dig Hole”) returns **dig hole**.

LTRIM()

Purpose	Trims leading spaces from a string.
Data type	Character
Syntax	LTRIM(<string>)
Where	String is a character string

Example — Assuming that the activity description is "^^^^^^Administration," the statement:

LTRIM(DESCRIPTION)

returns Administration.

> **Note:** In this example, each caret symbol (^) is used to represent a space.

MAX()

Purpose — Returns the maximum value from a list of variables.

Data type — Decimal, integer, character, date, or duration

Syntax — MAX(<value1> [,<value 2>, ...])

Where — <value1>, <value2> are decimals, integers, characters, or dates

> **Note:** Open Plan interprets a character argument as a comma-delimited list of items.

Example — Assuming that value1 is 4, value2 is 25, and value3 is 66, the statement:

MAX(4, 25, 66)

returns **66**.

Example — Assuming that USER_CHR01 contains a comma-delimited list of values 5, 1, 8, and 3, the statement:

MAX(USER_CHR01)

returns **8**

MID()

Purpose — Returns a specified number of characters in a string. If the number of characters to extract is not specified, this function returns all remaining characters in the string.

Data type — Character

Syntax — MID(<string>,<start>,<int>)

Where — <string> is the string expression being searched

<start> is the beginning location of the characters to be returned

<int> is optional; specifying the number of characters to be returned; if omitted, all remaining characters are returned.

> **Note:** The string expression is case sensitive.

Example — The statement:

MID("SITE COORDINATION AND DESIGN,"5, 12)

returns **COORDINATION**

MIN()

Purpose: Returns the minimum value from a list of variables.

Data type: Decimal, integer, character, date, or duration

Syntax: MIN(<value1>,<value 2>, ...)

Where: <value1>, <value2> are decimals, integers, characters, or dates

> **Note:** Open Plan interprets a character argument as a comma-delimited list of items.

Example: Assuming that value1 is 4, value2 is 25, and value3 is 66, the statement:

MIN(4, 25, 66)

returns **4**.

Example: Assuming that USER_CHR01 contains a comma-delimited list of the values 5, 1, 2, and 3, the statement:

MIN(USER_CHR01)

returns **1**.

MONTH()

Purpose: Returns a 2-digit numeric value for the month.

Data type: Integer

Syntax: MONTH(<date>)

Where: <date> is a date variable or user-entered date

Example: Assuming ESDATE is 04JUL04, the statement:

MONTH(ESDATE)

returns **7**.

NEWLINE()

Purpose: Inserts one or more lines in the calculated field expression.

> **Note:** While this function is available for both calculated fields and global edits, it is used in only calculated fields. The effect of the NEWLINE() function is displayed only in spreadsheet columns where wrapping is enabled.

Data type: Integer

Syntax: NEWLINE(<value>)

Where: <value> is an integer representing the number of lines to be added to the expression.

> **Note:** <value> cannot be blank

Example: The expression:

“Early:”+DATEFORMAT(ESDATE,”%D%A%Y”)+NEWLINE(1)+”Late: “+DATEFORMAT(LSDATE,”%D%A%Y”)+NEWLINE(1)+”Sched:”+ DATEFORMAT(SSDATE,”%D%A%Y”)

returns a string similar to the following:

Early:12Apr04

Late:16Apr04

Sched:14Apr04

Notice that the NEWLINE() function has forced the Late and Scheduled dates to be moved to the next line.

NUMBER_FORMAT()

Purpose: Returns a formatted string from a list of parameters.

Data type: Integer

Syntax: NUMBER_FORMAT(<integer>, <string>, <integer>)

Where: <integer> is the number to be formatted

<string> is a set of characters defining how the integer should be formatted. The following characters can be used in combination with one another (except where restrictions apply):

- “$” - displays the native currency symbol. (This character should not be used in conjunction with “%”)
- .”” – displays a decimal point within the integer. The number of decimal places can be indicated in the optional third argument <integer>.
- ,” ” – displays “thousands” separators.
- “%” – displays the resulting integer as a percentage. (This character should not be used in conjunction with “$”)

<int> is optional; specifying the number of decimal places to be returned. Up to 20 decimal places are allowed. If the number is set to 0, the decimal display is overridden.

> **Note:** If there are any unrecognized characters in the format string or if the format string contains both “$” and “%,"the format string will be returned instead of a value.

A blank <string> is interpreted as a DEFAULT format, but the argument cannot be omitted.

Example: The expression:

NUMBER_FORMAT(1000, “$", 2)

Returns **$1,000.00**

OCCURS()

Purpose: Returns the number of times a string occurs within another string.

Data type: Integer

Syntax OCCURS(<string1>,<string2>)

Where <string1> is the string expression to be searched

<string2> is the substring to be searched

> **Note:** The string expressions are case sensitive.

Example The statement:

OCCURS(“Bright light circuits”,"light”)

returns **2**.

PARENT()

Purpose Returns the parent ID of a specified level for an ID or code.

Data type Character

Syntax PARENT(<ID>, <level>)

Where <ID> is a character variable or constant

<level> is optional; specifying the level at which you want the local portion

Example Assuming that a code file is assigned to the project in position C1, then the statement:

PARENT(C1)

returns the ID of the immediate parent of C1.

RECORD_NUMBER()

Purpose Provides a unique number that can be used to identify a particular record.

Data type Integer

Syntax RECORD_NUMBER()

Where <No arguments>

Example This function is useful in general export scripts. For example:

EXPORT csv Sample Export Script

TABLE ACT

FIELD RECORD_NUMBER()

FIELD ACT_ID

RIGHT()

Purpose Returns a specified number of rightmost characters in a string.

Data type Character

Syntax RIGHT(<string>,<int>)

Where <string> is the string expression to be searched

<int> is the number of characters to be returned

Example The statement:

RIGHT("SITE COORDINATION AND DESIGN," 6)

returns **DESIGN**.

ROUND()

Purpose Returns a numeric or duration value rounded to a specified precision.

Data type Decimal, integer, or duration

Syntax ROUND(<value>, <precision>, <type>)

Where <value> is a numeric variable or constant

<precision> is a positive or negative number. If positive, <precision> represents the number of decimal places. A negative precision represents rounding that occurs to the left of the decimal point, i.e., integer rounding (with the decimal point being at precision 0). A –1 represents units of 10, -2 represents units of 100, and –3 represents units of 1000 (10 to the first, second, and third powers, respectively). With a precision of –1, the last two digits of a given number will be rounded to the nearest multiple of 10.

You can also modify the precision of a duration using one of the following codes:

- t (minutes)
- h (hours)
- d (days)
- w (weeks)
- m (months)

Note: The code indicating the duration precision is a string and must be placed within quotation marks.

<type> indicates how Open Plan should round. Valid values are "up" (rounds all decimals up), "down" (rounds all decimals down), or the field can be empty (rounds up values =>.5, and rounds down values <.5) This parameter is optional.

Example The statement:

ROUND(458.9738, 2, "up")

returns **458.98**.

Example The statement:

ROUND(orig_dur / 3.0, "d")

returns the original duration divided by 3.0 and rounded up to the nearest day.

Example The statement:

ROUND(4589, -2)

returns **4600**.

SPACE()

Purpose	Returns a specified number of space characters.
Data type	Character
Syntax	SPACE(<int>)
Where	<int> is the number of space characters to be returned
Example	The statement
	SPACE(6)
	Returns six space characters.

SQRT()

Purpose	Returns the square root of the absolute value of a numeric value.
Data type	Decimal or integer
Syntax	SQRT(<value>)
Where	<value> is a numeric variable or constant
Example	The statement:
	SQRT(4)
	returns **2**.

STR()

Purpose	Converts a numeric to a character variable.
Data type	Character
Syntax1	STR(<value>,<length>,<decimal>)
Where	<value> is a decimal, integer, or duration
	<length> is a numeric or constant representing the length of the resultant character string including the decimal point and decimal places. This parameter is optional.
	<decimal> is a numeric variable or constant representing the number of decimal places. This parameter is optional.
Syntax2	STR(<date field>)
Where	<date field> is a date field
Example	The statement:
	STR(1.2345,3,1)
	returns **1.2**.
Example	Assuming that ESDATE is October 5, 2004, the statement:
	STR(ESDATE)
	Returns **05OCT04**

> **Note:** The string returned when using STR() with a date field will use the project's current date format.

STRTRAN()

Purpose: Replaces one substring with another substring and returns the revised string.

Data type: Character

Syntax: STRTRAN(<string1>,<string2>,<string3>)

Where: <string1> is the string expression to be searched

<string2> is the substring to be searched

<string3> is optional; is the replacement string; if omitted, a null string is assumed and <string2> is simply deleted

> **Note:** The string expressions are case sensitive.

Example: The statement:

STRTRAN("SITE MANAGEMENT","MANAGEMENT","SPECIFICATION")

returns **SITE SPECIFICATION**.

STUFF()

Purpose: Replaces a specified number of characters with another substring and returns the revised string.

Data type: Character

Syntax: STUFF(<string1>,<start>,<int>,<string2>

Where: <string1> is the string expression being searched

<start> is the position in the string expression where the replacement should begin

<int> is the number of characters to be removed before a replacement string is inserted; if 0, no characters are removed

<string2> is optional; specifies the replacement string; if omitted, a null string is assumed and the characters specified by <int1> and <int2> are simply deleted

Example: The statement:

STUFF("SITE MANAGEMENT," 6,10, "SPECIFICATION")

returns **SITE SPECIFICATION**.

SUBSTR()

Purpose: Copies a specified part of a character string.

Data type: Character

Syntax	SUBSTR(<string>, <start>, <length>)
Where	<string> is a character string
	<start> is a numeric variable or constant representing the first character to be copied
	<length> is a numeric variable or constant representing the length of the character string to be copied. The parameter is optional.
Example	The statement:
	SUBSTR(“ABCDEF,"3, 2)
	returns **CD**.
Example	Assuming the ID = “P100,"the statement:
	SUBSTR(ID, 1, 2)
	returns **P1**.

TIMENOW()

Purpose	Returns the currently set Time Now date.
Data type	Date
Syntax	TIMENOW( )
Example	Assuming the currently set Time Now date is October 7, 2004, the statement:
	TIMENOW( )
	returns **07OCT04**.

TRIM()

Purpose	Removes trailing blanks from a character variable or constant.
Data type	Character
Syntax	TRIM(<string>)
Where	<string> is a character variable or constant
Example	Assuming that the activity description is “Administration^^^^^^^^^^,"the statement:
	TRIM(DESCRIPTION)+”!!!”
	returns **Administration!!!**.

> **Note:** In this example, each caret symbol (^) is used to represent a space.

UPPER()

Purpose	Converts alphabetic characters from lowercase to uppercase.
Data type	Character
Syntax	UPPER(<string>)
Where	<string> is a character variable or constant

Example	The statement: UPPER(“Dig Hole”) returns **DIG HOLE**.

USER_ID()

Purpose	Returns name of current user.
Data type	Character
Syntax	USER_ID(<string>)
Where	(No arguments)
Example	USER_ID() returns "SYSADMIN."

VAL()

Purpose	Converts a character string to a numeric value. It can also be used to convert durations to minutes.
Data type	Decimal or integer
Syntax	VAL(<string>)
Where	<string> is a character or duration
Operation	If <string> is not a valid numeric value, VAL converts the string starting at the leftmost character until an invalid numeric is encountered. Integer zero is returned if the string does not start with a valid numeric value.
Example	The statement: VAL(“1678”) returns **1678**.
Example	Assuming that the original duration of an activity is 100 days, the statement: VAL(ORIG_DUR) returns **48000** based on conversion preferences set for the project of 8 hours per work day.

YEAR()

Purpose	Returns a 4-digit numeric value for the year.
Data type	Integer
Syntax	YEAR(<date>)
Where	<date> is a date variable or user-entered date
Example	Assuming ESDATE is 04JUL04, the statement: YEAR(ESDATE) returns **2004**.

Examples of Custom Calculated Fields

The following are examples of custom calculated fields.

_COMPARE_2_DATES

Expression	SUBSTR(ESDATE, 1, 8) = SUBSTR(LSDATE, 1, 8) Or DATEFORMAT(ESDATE, “%C%M%D”) = DATEFORMAT(LSDATE,”%C%M%D”)
Table	Activity
Data type	Logical
Purpose	Compares two date fields without regard to the time portion of the date string and returns **True** if dates are the same. This example uses the early start and late start dates.

_DISPLAY_MIXED_DUR_IN_DAYS

Expression	STR((VAL(ORIG_DUR) /60.0 /8), 6, 2) + “ days”
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to days, and displays the result labeled as **days**. The calculation is based on eight hours per day.

_DISPLAY_MIXED_DUR_IN_HOURS

Expression	STR((VAL(ORIG_DUR) /60.0), 6, 2) + “ hours”
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to hours, and displays the result labeled as **hours**.

_DISPLAY_MIXED_DUR_IN_MINUTES

Expression	STR((VAL(ORIG_DUR)), 8, 0) + “ minutes”
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to minutes, and displays the result labeled as **minutes**.

_DISPLAY_MIXED_DUR_IN_WEEKS

Expression	STR((VAL(ORIG_DUR) /60.0 /40), 6, 2) + “ weeks”
Table	Activity
Data type	Character

Purpose	Converts duration expressed in any unit to weeks, and displays the result labeled as **weeks**. This calculation is based on 40 hours per week.

_DISPLAY_MIXED_DUR_IN_MONTHS

Expression	STR((VAL(ORIG_DUR) /60.0 /160), 6, 2) + " months"
Table	Activity
Data type	Character
Purpose	Converts duration expressed in any unit to months, and displays the result labeled as **months**. This calculation is based on 160 hours per month.

_FORMATTED_DATE

Expression	DATEFORMAT(ESDATE,"%A %D, %C")
Table	Activity
Data type	Character
Purpose	Displays the early start date in month abbreviation, day of month, four-digit year format. For example, this calculated field might return a date such as **Oct 10, 2004**. For a complete list of standard date formats, search for DATEFORMAT() in the Open Plan help system.

_INDENT_A_FIELD

Expression	IIF(LEVEL(ACT_ID) > 1, space((LEVEL(ACT_ID) - 1) * 3) + DESCRIPTION, DESCRIPTION)
Table	Activity
Data type	Character
Purpose	Indents any field based on the level of the activity ID. This example will indent the activity description 3 spaces for each ID level after the first.

_NUMERIC_TO_DURATION

Expression	|1d| * USER_NUM01
Table	Activity
Data type	Duration
Purpose	Converts any numeric field to the duration data type. This example uses User Numeric Field 1 and days for the duration units.

_START_HOURS_AT_0000

Expression	DATEFORMAT(ESDATE, '%D%A%C 00:00')
Table	Activity
Data type	Date
Purpose	Changes the time portion of a date field to 00:00. This example uses the early start date.

_END_HOURS_AT_2400

Expression	DATEFORMAT(EFDATE, '%D%A%C 24:00')
Table	Activity
Data type	Finish Date
Purpose	Changes the time portion of a date field to 24:00. This example uses the early finish date.

_ID_ANY_LEVEL

Expression	IIF(LEVEL(ACT_ID) >= 3, LOCAL(PARENT(ACT_ID, 3)), “”)
Table	Activity (or any table with a hierarchical field)
Data type	Character
Purpose	Returns the value of the specified level of the hierarchical ID field used. This example will return the value of the third level of the activity. If the activity ID has less than three levels, it returns nothing. If you want something other than the third level, change the numbers representing

Temporary Calculated Fields

In addition to allowing you to name and save calculated fields for future use, Open Plan allows you to create and use temporary calculated fields in the following circumstances:

- Export scripts
- Spreadsheet **Add**, **Insert**, and **Edit Column** dialog boxes
- Multi-table define links
- When defining the box layouts for the network and hierarchy views
- When defining box attributes for the network view
- OLE automation

These temporary calculated fields are not displayed in the **Calculated Fields** dialog box. They are not saved as part of the project. If you save the view while a temporary calculated field is in effect, Open Plan saves the view with the definition applied.

To define a temporary calculated field in a network view, complete the following steps:

1. Display the network view to which you want to apply the temporary calculated field.
2. Take one of the following actions:
   - On the **View** tab, click **Preferences**.
   - Right-click an empty area of the view, and click **Preferences** on the context menu.
3. On the **Preferences** dialog box, click the **Box Attribute** or the **Box Layout** tab.
4. Enter the temporary calculated field in the **Criterion** or **Field** column, and click **OK**.

   Open Plan displays the results of the calculated field in the network view.

To define a temporary calculated field in a spreadsheet view, complete the following steps:

1. Display the spreadsheet view to which you want to apply the temporary calculated field.
2. Right-click the column heading, and click **Add**, **Insert**, or **Edit** column.

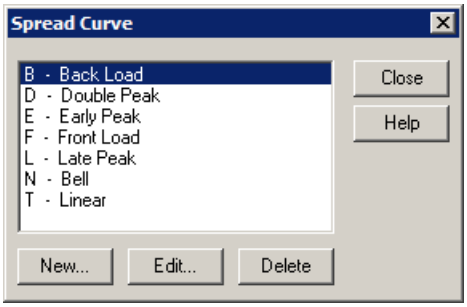
   Open Plan displays the appropriate **Column** dialog box.
3. Enter the calculated field in the **Field Name** field and a column title in the **Title** field, and click **OK**.

   Open Plan displays the results of the calculated field in the selected column.

Spread Curves

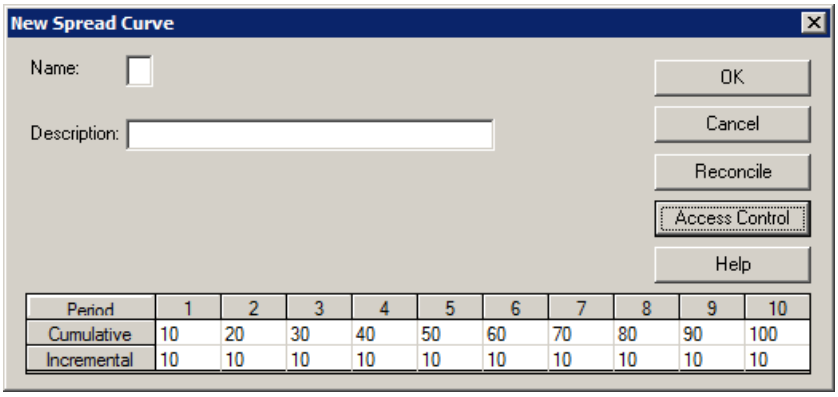
Spread curves allow you to define the manner in which a resource assignment is distributed over the duration of an activity. When assigning a resource to an activity, you can use the **Curve** field on the **Resources** tab of the **Activity Details** dialog box to select a predefined spread curve.

When you click **Spread Curves** on the **Tools** tab, Open Plan displays the following dialog box:



This dialog box lists the spread curves defined for Open Plan. Using this dialog box, you can create a new spread curve, edit a spread curve, or delete a spread curve.

Clicking **New** on the **Spread Curves** dialog box displays the **New Spread Curve** dialog box:



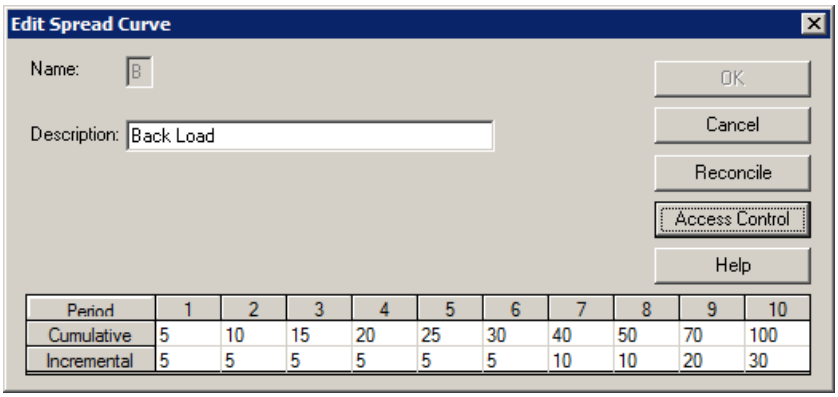
This dialog box features the following controls:

- **Name** — Open Plan uses the (single) character entered in this field to identify the spread curve.

  If you enter a character that is already in use for another spread curve, Open Plan issues a warning when you click **OK**.
- **Description** — This field allows you to specify a more descriptive name for the spread curve. Once defined, the spread curve is identified by both its Name and its Description. You cannot leave this field blank. You must enter a description for the spread curve.

- **Cumulative** — As you enter values in the grid, Open Plan recalculates the values and displays the current total in column 10 of the **Cumulative** row.
- **Incremental** — The values in the **Incremental** row allow you to define the percentage to be applied to each specific period.
- **Reconcile** — If you click this button, Open Plan ensures that the value in column 10 of the **Cumulative** row equals 100%. If necessary, Open Plan redistributes the values in the remaining cells in such a manner as to honor your inputs to the extent that it is possible.

When you select an existing spread curve from the Spread Curve dialog box and click Edit, Open Plan displays the Edit Spread Curve dialog box:



Period	1	2	3	4	5	6	7	8	9	10
Cumulative	5	10	15	20	25	30	40	50	70	100
Incremental	5	5	5	5	5	5	10	10	20	30

This dialog box is the same as the **New Spread Curve** dialog box except that the **Name** field is disabled. You can edit only the description and percentage values in the **Cumulative** and **Incremental** rows.

Note: Be aware that modifying or deleting a spread curve that is in use by another project will affect the project the next time resource scheduling is performed or a baseline is updated.

To create a new spread curve, complete the following steps:

1. On the **Tools** tab, click **Spread Curves**.
2. On the **Spread Curve** dialog box, click **New**.
3. In the **Name** field of the **New Spread Curve** dialog box, enter a single character to serve as the name.
4. In the **Description** field, enter a longer description for the spread curve.
5. In the grid at the bottom of the dialog box, enter the appropriate values in any combination of cells in the **Cumulative** and **Incremental** rows.

   The total in the **Cumulative** row is displayed in column 10. This value must equal 100%.
6. Click **Reconcile** to ensure that the values in each column of the Cumulative row are correct.
7. Click **OK**.

To edit a spread curve, complete the following steps:

1. On the **Tools** tab, click **Spread Curve**.
2. From the **Spread Curve** dialog box, select the spread curve to edit.

3. Click Edit.
4. In the **Edit Spread Curve** dialog box, enter a new description and percentage values as desired.

   The total in the **Cumulative** row is displayed in column 10. This value must equal 100%.
5. Click **Reconcile** to ensure that the values in each column of the Cumulative row are correct.
6. Click **OK**.

To delete a spread curve, complete the following steps:

1. On the tools tab, click **Spread Curves**.
2. From the **Spread Curve** dialog box, select the spread curve to delete.
3. Click **Delete**.

Spread Curve Profiles

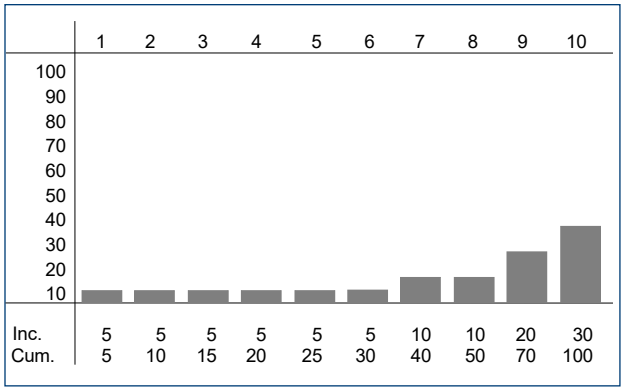
In Open Plan, resource and cost assignment profiles can be defined by spread curves that describe how the assignment is spread through the duration of an activity. Open Plan provides the following standard profiles:

- Back load (B)
- Double peak (D)
- Early peak (E)
- Front load (F)
- Late peak (L)
- Bell (Normal) (N)
- Linear (T)

All assignment profiles are described by defining values for 10 evenly spaced percentile points, which are then mapped to the activity duration. The following sections discuss each of these assignment profiles in more detail.

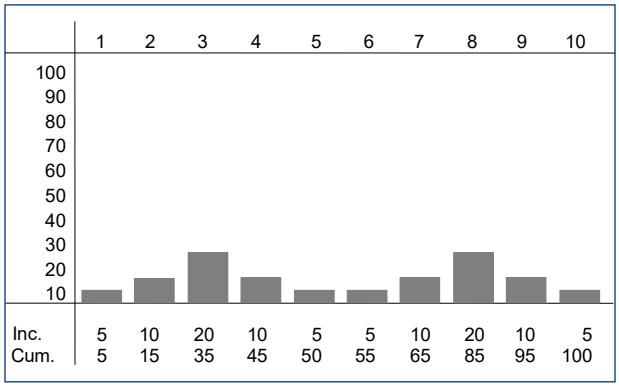
Back Load Profile

The back load assignment profile describes an assignment in which most of the resource effort is allocated toward the end of the activity duration. The curve code for a back load assignment profile is B.



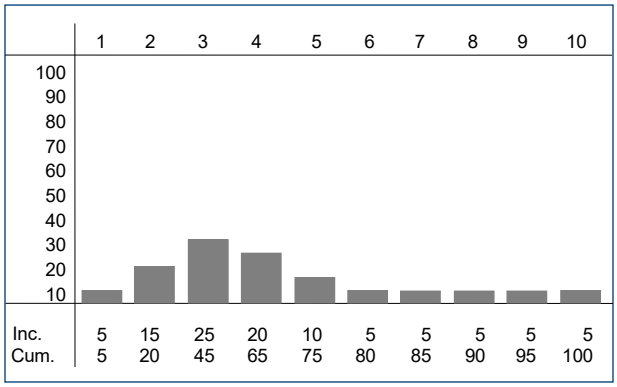
Double Peak Profile

The double peak assignment profile describes an assignment in which most of the resource effort is allocated to two peaks — one near the beginning of the activity duration, and one near the end. The curve code for a double peak assignment profile is D.



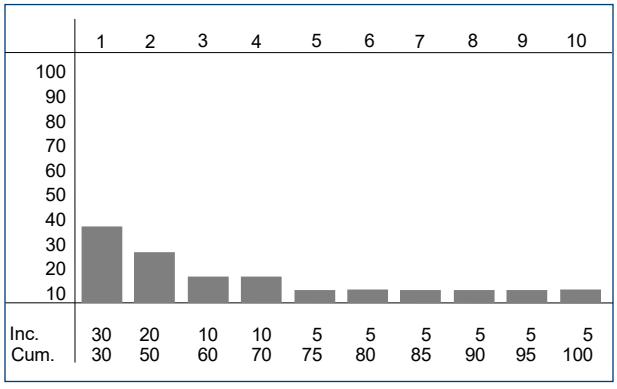
Early Peak Profile

The early peak assignment profile describes an assignment in which most of the resource effort is allocated near the beginning of the activity duration. The curve code for an early peak assignment profile is E.



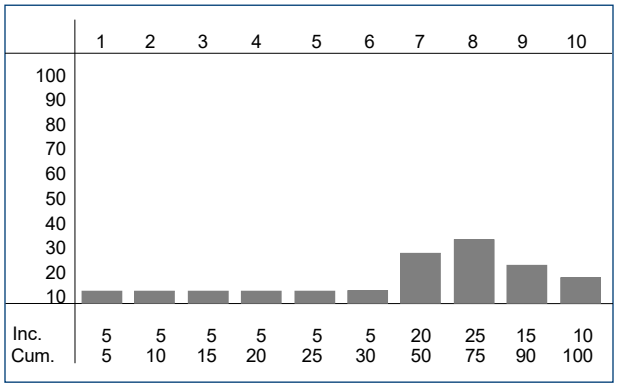
Front Load Profile

The front load assignment profile describes an assignment in which most of the resource effort is allocated at the beginning of the activity duration. The curve code for a front load assignment profile is F.



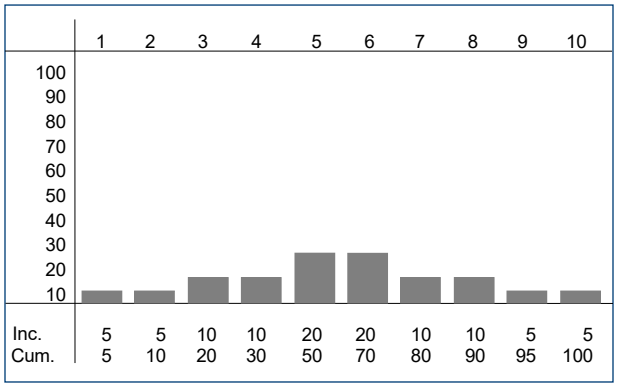
Late Peak Profile

The late peak assignment profile describes an assignment in which most of the resource effort is allocated near the end of the activity duration. The curve code for a late peak assignment profile is L.



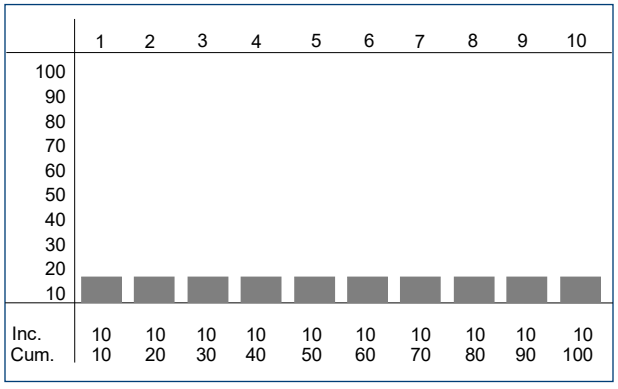
Bell (Normal) Profile

The bell (normal) assignment profile describes an assignment in which a large proportion of the resource effort is allocated near the middle of the activity duration. The resource effort starts with a minimal effort, gradually increases to the middle of the duration, and then tapers off toward the end. The curve code for a bell (normal) assignment profile is N.



Linear Profile

The linear assignment profile describes an assignment in which the resource effort is allocated as evenly as possible throughout the activity duration. The curve code for a linear assignment profile is T.



Frequently Asked Questions

This topic is designed to provide answers to some frequently asked questions about Open Plan.

Spreadsheet Views

Question	Answer
Why does Open Plan Professional sometimes prevent me from defining a grouping in a spreadsheet view?	If you are doing a grouping across tables (for example, you want to display both activity and resource information), then grouping is not allowed in a single table view. You can use a Multi-Table view and grouping across tables.
I have grouping turned on in a spreadsheet view and a blank row is inserted. If I put something in that blank row and then click somewhere outside of that row, the row I typed in disappears. What's happening?	The row you typed in did not disappear. It was moved to the appropriate location based on the grouping criteria.
I applied a filter to my spreadsheet, and it worked fine. Then I added a new row that should have been filtered out. Even when I clicked away from the row, the new row stayed. Why?	When a filter is applied to a view, the view compares all existing records to the filter. As you add new records, however, Open Plan does not apply the filter to these records. Keeping this information visible allows you to review newly-entered data and make any changes. To see the effect of the filter on any new data, re-apply the filter to the view.
I saved a multi-table spreadsheet view with a link in it as a template. Now I can't get the view to open.	If there is a case where a link has no columns in it, then the spreadsheet view will not display. One possible reason is that the link definition may not be valid for this particular project. For example, the link included either the baseline activity or baseline usage table, but the project in question doesn't have a baseline or no baseline is selected.

Barchart Views

Question	Answer
Why does my barchart report start at the wrong date?	You need to set the project dates you wish to display in the **Print Options** dialog box.
Why are my bars printing solid black?	Check the settings in your Windows printer configuration. The **Control Panel/Printers/Setup/Options** dialog box is specific to your output device. If your printer supports a **Dithering** option, make sure that it is set to either **Fine** or **LineArt**.

Question	Answer
I don't understand why the label for my reporting calendar is showing up at the end of the period. How is this label associated with the period?	When using reporting calendars, each interval to be marked is defined by a pair of dates. If the caption is positioned at the center or right of the interval being marked, it corresponds to the date at the end of the period; otherwise, it corresponds to the date at the beginning. This is also the case with regular date ribbons (that is, without a reporting calendar).
I created a new bar. Why is my new bar not appearing?	Check the visibility setting in the **Bar Set Preferences** dialog box. If this seems to be correct, check that you do not have a filter set that excludes activities that use the bar. Note that if a filter excludes an activity, it also excludes its children.
How can I increase the number of lines shown in the barchart view?	You can reduce the font size on the **Spreadsheet Preferences** dialog box. Then you can reduce the row height in the **Bar Types** dialog box. This dialog box is accessed from the **Bar Attributes** tab of the **Bar Set Preferences** dialog box.
How do I roll up on codes in the barchart view?	Assume that you want to roll up to Level 2 of the code file stored in the C1 field. Display an activity barchart view and follow this procedure: 1. From the **View** tab, click **Preferences** from the **Spreadsheet** group. 2. On the Options tab of the Spreadsheet Preferences dialog box, select Enable Groupings. 3. Display the **Grouping** tab, and set up a grouping heading using the following controls: ▪ Break On — C1 ▪ Break Level — Enter the level at which you want the grouping to break. For example, enter 2 to break at level 2. ▪ Background — Select a color to distinguish the rolled up rows from the detail levels. ▪ Heading — Select Yes. ▪ Summary — Select Top. ▪ Heading Field — Leave as <Default> 4. On the **View** tab, click **Bar Sets** and select the current bar set. 5. Display the **Bar Attributes** tab of the **Bar Set Preferences** dialog box, and set up a new bar using the following controls: ▪ Criterion — Leave this field blank.

Question	Answer
	▪ Bar Type — Early Dates ▪ Pattern — Your choice ▪ Outline — Your choice ▪ Fill — Your choice ▪ Visibility — Grouping Summary Only ▪ Label — Summary
Where can I see activities and resources listed on one table?	Use an activity/resource barchart view or use a multi-table spreadsheet/barchart view and define the link appropriately.

Histogram Views

Question	Answer
Why is the **Select Resource** option dimmed out in my histogram view?	A resource file is not attached to the project. Assign an existing resource file to your project using one of the following procedures: ▪ In the **Open Plan Explorer**, open the **Resources** folder located in the **Open Plan Library** folder. Drag the icon representing the resource file onto your project folder. ▪ Select the project folder and click **Properties** on the project name. On the **Files** tab, click the ellipsis to the right of the **Resource Definition File** field. Then select the resource file from the list or create a new one.
How exactly is the data plotted in histogram views?	When the histogram shows cumulative data, which is done as an S-curve rather than a histogram, the plotted points are at the boundaries of the intervals and represent the data at those points. By contrast, histogram bars represent the incremental amount occurring between the start and the finish of the interval. If the cross-table is present, the data displayed in a particular column represents the values plotted at the end of that interval. For more information, see Histogram Views.
Why is a histogram view truncating my data? I am trying to show 20 years of dates using hourly intervals.	Histogram views have a default limit of displaying approximately 100000 intervals. (You can modify this default limit by changing a setting within Open Plan.) To change the setting, follow this procedure: 1. In a histogram view, display the **Options** dialog box by clicking the **File** menu **» Preferences » Application**.

Question	Answer
	2. On the **Advanced** tab, select **HistogramArrayLimit** from the list. 3. The default value is set to 100000. Enter a new value for the appropriate number of intervals. 4. Click **OK**. The default limit of 100000 intervals allow you to represent approximately 8 years in days and approximately 4 months in hours. We recommend that you choose an appropriate date scale interval for the length of the project, typically weeks for a long project. Or you can use a reporting calendar, which can combine different intervals.

Resources

Question	Answer
Why do I sometimes show overloaded resources after resource-limited resource scheduling?	Check the scheduling interval specified on the **Advanced** tab in the **Resource Scheduling** dialog box. This should correspond to the intervals defined in the calendars used by the project. For example, even if you defined all activity durations in days, you may have an activity or resource calendar that breaks the workday into two 4-hour shifts. Moreover, scheduling intervals are defined as starting at 12:00 midnight. (Thus, 6-hour scheduling intervals start at 12:00AM, 6:00AM, 12:00PM, and 6:00PM.) If the pattern of working intervals defined in a calendar does not match up with the pattern of scheduling intervals, Open Plan may calculate a resource overload in certain situations. By reducing the scheduling interval to a smaller unit, the calculations will yield the correct results. It is also possible to overload resources during resource-limited scheduling in the following situations: ▪ When you have assigned resources to a subproject ▪ When you have assigned resources to a hammock ▪ When you have assigned the **Immediate** attribute to an activity
Why aren't my resources being progressed?	The option to automatically progress resources based on activity progress may not be enabled. To make sure this option is enabled for each resource you want to progress, use one of the following procedures: ▪ Display the **Resource Details** dialog box for each resource.

Question	Answer
	▪ Add a column to a resource/activity spreadsheet view that displays the **Progress Based on Activity Progress** field from the **Resource** table.

Calculated Fields

Question	Answer
I am trying to define a calculated field that calculates early start + 10 days by using ESDATE + |10d|, but I keep getting a date which is only 3 days later than the early start date. Why?	Durations used in calculations use the project definition of what a day or week or month is. So if a day is defined as 8 hours, then ESDATE + |10d| will add 80 hours to the date, which is a little over 3 days. To add 10 days, use 240 hours. The same is true if you use the DATEADD and DATEDIFFERENCE functions without a calendar. As an alternative, try DATEADD(esdate, |10d|, calendar).
I am testing a value in a calculated field, but the test never seems to be true.	Check to see what field is being tested. For example, a code field actually contains a character value, not a numeric. Therefore, in order to test for the value 1, you might use a calculation such as IIF (C1 = "1", 1, 2) and not IIF(C1=1, 1, 2).
What is the maximum number of characters that can be returned by a calculated field?	For a calculated field returning a character result, the maximum length is approximately 32,765 characters.

Durations

Question	Answer
When I enter values in duration fields with a time unit (for example, 1M), I do not always get the results I expect. Why is this?	In duration fields, the use of days, weeks, and months is merely a data entry convenience and should not be taken too literally. These periods are exactly as defined in the **Preferences** tab of the **Project Properties** dialog box. By default, a month is 160 hours; if you use this default, entering 1M for an activity duration is exactly the same as entering 160H. If you subsequently alter the definition of a month, then any duration entered in months will be changed. In this case, if you change the definition of month from 160 to 168, the duration will still appear as months, but the activity scheduled will take 168 hours instead of 160 hours.
Why does Open Plan sometimes show durations for a summary activity such as a subproject or a hammock in minutes even though I have defined	This is usually due to the use of activity calendars that do not match the project definition of a day (defined on the **Preferences** tab of the **Project Properties** dialog box). Although Open Plan allows you to mix calendar and duration units within a project, on any individual activity it is desirable

Question	Answer
the durations of all of the detail activities in days?	that they be consistent. For example, if durations are to be input in days for a particular set of activities, then the calendar on those activities should define all working days as the same length, and this length should match the project definition of a day.
Why did a 3-day activity that is progressed as 75% complete show one day remaining?	If the original duration is measured in days, then progress will normally be rounded to the nearest day. This can easily be overridden by typing in a remaining or elapsed duration in different units. If you want to define a 3-day activity (with each day consisting of 8 hours) as *exactly* 75% complete, enter 6 hours for the remaining duration.

Miscellaneous

Question	Answer
I'm tired of typing in the entire word in lists when the choice is right there in the list: Do I have to type the entire word each time to select it?	You need to type only enough characters to uniquely identify the choice before pressing the **Enter** key.
How do I assign multiple calendars to my project?	While only one calendar file can be assigned to a project at any one time, this calendar file can contain any number of individual calendars. Once you have stored calendars in the calendar file, you can assign specific calendars to resources, activities, and relationship lags.
How do I view a session log?	By default, if errors occur when running a process (resource scheduling, time analysis, etc.), Open Plan will prompt you to view the session log. To view a session log, click **Open Log** on the **Processes** tab, and click the appropriate session log tab. If you would like Open Plan always to prompt you to view the session log after running a process, clear the appropriate option on the **Project** tab of the **Options** dialog box. (Click the **File** menu **» Preferences » Application**.)

Appendix A: Create/Update Baseline Option Flags

In version 8.0 of Open Plan Professional, more options were added to the CreateBaselineWithOptionFlags() and UpdateBaselineWithOptionFlags() methods. In order to be backwards compatible, these options are only active when the flag **Baseline_Use_Intfc_V2** is set. Otherwise, they are ignored.

Attention: For information about the methods, see:

- CreateBaselineWithOptionFlags() method
1. UpdateBaselineWithOptionFlags() method

The following table lists the names of each flag, a description of its usage, whether it is used when creating or updating the baseline, and in which version of the interface it is valid.

Flag	Description	Used when Creating/Updating Baseline	Version	Value
Baseline_Include_In_Progress_Activities	Set this flag to include in-progress activities in update logic.	Create/Update	Version 1 & 2	1
Baseline_Include_Completed_Activities	Set this flag to include completed activities in update logic.	Create/Update	Version 1 & 2	2
Baseline_Existing_Replaced_By_Progressed_Dates	If the flags 'Baseline_Use_Project_Start' and 'Baseline_Update_Data' are set, the 'Baseline_Existing_Replaced_By_Progressed_Dates' flag controls whether or not dates in the baseline activity record are modified. When clear, dates in the baseline activity records are preserved. When set, dates in the baseline activity records are replaced with dates from the associated activity record.	Create/Update	Version 1 Only	4
Baseline_Rollup_Data	Set this flag to roll up all dates, budget cost, and labor quantity data in the baseline to the parent levels.	Create/Update	Version 1 & 2	8
Baseline_Delete_Deleted_Activities	Set this flag to remove any activities from the baseline that no longer exist in the current project.	Update	Version 1 & 2	16
Baseline_Use_External_Baselines	Set this flag for Open Plan to use baseline information from external subprojects. Only valid when adding Baseline of the same name as the selected master baseline.	Create	Version 1 & 2	32
Baseline_Use_Remaining_For_Planned_Activities	Set this flag to update baseline activity resource requirements from the remaining resource requirements instead of the budget resource requirements for activities that are not in-progress or complete.	Create/Update	Version 1 & 2	64

Flag	Description	Used when Creating/Updating Baseline	Version	Value
	This flag is not available when the baseline is based on Schedule Dates.			
Baseline_Use_Project_Start	Set this flag to create baseline dates by performing a time analysis from the project start date. This flag is not available when the baseline is based on Schedule Dates.	Create	Version 1 & 2	128
Baseline_Update_Data	When set, activities already in the baseline are updated. In version 1, all data (Codes, UDFs, and Descriptions) and Resource Quantities are updated when this flag is set. Baseline Activity dates may or may not be set, depending on the value of 'Baseline_Existing_Replaced_By_Progressed_Dates' (See description below) In version 2, only the data (Codes, UDFs, and Descriptions) is updated. In Version 1: This flag is only valid in Open Plan 3.5, and earlier, and is only enabled if the Baseline_Use_Intfc_V2 flag is NOT set. In Version 2: This flag is only valid in Open Plan 8.0 or higher, and is only enabled if the Baseline_Use_Intfc_V2 flag is set.	Update	Version 1 & 2	256
Baseline_Replace_Dates	Set this flag to instruct Open Plan to replace the baseline dates of activities that already exist in the baseline. This flag is only valid in Open Plan 8.0 or higher, and is only enabled if the Baseline_Use_Intfc_V2 flag is set.	Update	Version 2 Only	512
Baseline_Update_Resources	Set this flag to instruct Open Plan to update baseline activity resource usage. When baseline dates are to be replaced, the resources will be updated. When not replacing the baseline dates, and if resources have changed, the current quantity will be evenly spread across the original baseline dates or a subset of the original baseline dates, depending on actual cost record dates and other factors. This flag is only valid in Open Plan 8.0 or higher, and is only enabled if the Baseline_Use_Intfc_V2 flag is set.	Update	Version 2 Only	1024
Baseline_Update_External_Subprojects	This flag is ignored if the **Use Baseline of Same Name on External Subprojects** option was not selected when this baseline was created. If this flag is set, in order for the external subproject baselines to be updated, you must have rights to update the external baseline. These rights are granted in the Baseline Access Control dialog box for each	Update	Version 2 Only	2048

Flag	Description	Used when Creating/Updating Baseline	Version	Value
	of the external baselines. The Update in the Master field must be set to Yes for each external baseline to be updated. This flag is only valid in Open Plan 8.0 or higher, and is only enabled if the Baseline_Use_Intfc_V2 flag is set.			
Baseline_Use_Intfc_V2	When set, enables Version-2 only flags and associated logic. When clear, Version 2 flags are ignored.	Update	Version 2 Only	4096

Appendix B: Deltek Open Plan Data Structure

Attention: For more information about the table relationships, see the *Deltek Open Plan Entity Relationships Diagram* PDF.

System Data

Configuration Data

Table OPP_ALR - Email Advisory Definitions

The Email Alerts table contains Open Plan email alerts definitions. System-defined items are stored with a blank USR_ID field. The data dictionary table identifier for table OPP_ALR is ALR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ADDR_FIELD	None	The email address to which an email advisory message is to be sent.	TEXT	VARCHAR2(32)	NOT NULL
ADDR_FMT	None	Flag indicating whether the email addresses are embedded in the field or listed in a field.	INTE	NUMBER(10, 0)	NOT NULL
ALR_ID	None	The unique name of an email advisory definition.	TEXT	VARCHAR2(60)	NOT NULL
ALR_UID	None	The unique identifier of an email advisory definition.	GUID	VARCHAR2(22)	NOT NULL
COMBINE	None	Flag indicating that Open Plan will combine all the messages generated by the advisory into a single email that is sent to each addressee.	INTE	NUMBER(10, 0)	NOT NULL
FILTER	None	The name of a filter assigned to an email advisory definition.	TEXT	VARCHAR2(1024)	NULL
LASTUPDATE	None	The date and time that the email advisory definition was last updated.	DATE	DATE	NOT NULL
MESSAGE	None	The body of message that will be sent by the email advisory definition.	MEMO	LONG	NULL
OWNER_ID	None	The user ID of the owner of the email advisory definition.	TEXT	VARCHAR2(20)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUBJECT	None	The subject line for emails generated by an email advisory definition.	TEXT	VARCHAR2(100)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
USR_ID	None	If empty, indicates that the email advisory definition is shared by all users.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_BGE - Batch Global Edit Definitions

The Batch Global Edit Definitions table contains batch global edit definitions. The data dictionary table identifier for table OPP_BGE is BGE.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BGE_ID	None	The name of a batch global edit definition.	TEXT	VARCHAR2(60)	NOT NULL
BGE_UID	None	The unique identifier of a batch global edit definition record.	GUID	VARCHAR2(22)	NOT NULL
GLOBALEDITS	None	The global edit expressions contained in the batch global edit definition.	MEMO	LONG	NOT NULL
LASTUPDATE	None	The last date and time that the batch global edit definition record was updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The user ID of the owner of the batch global edit definition.	TEXT	VARCHAR2(20)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TABLE_TYPE	None	The table type to which the batch global edit applies.	TEXT	VARCHAR2(30)	NOT NULL
USR_ID	None	If empty, indicates that the batch global definition is shared by all users.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_CLC - Calculated Fields, Filters & Global Edits

The Calculated Fields, Filters & Global Edits table contains Open Plan calculated field, filter, and global edit definitions. All of these items may be referred to as calculated fields. System-defined items are stored with a blank USR_ID field. The data dictionary table identifier for table OPP_CLC is CLC.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLC_ID	None	The name of a calculated field, filter or global edit, unique within each table type for which calculated fields, filters, and global edits may be defined.	TEXT	VARCHAR2(60)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLC_UID	None	The unique identifier of a calculated field record.	GUID	VARCHAR2(22)	NOT NULL
EXPRESSION	None	A calculated field, filter or global edit expression.	TEXT	VARCHAR2(2000)	NULL
FILTER	None	The filter used by a global edit expression.	TEXT	VARCHAR2(1024)	NULL
FLAGS	None	Contains a value that identifies a record as a calculated field or a filter.	INTE	NUMBER(10, 0)	NOT NULL
GEDATA	None	The field that is updated by a global edit expression. The presence of a value in this field identifies a record in this table as a global edit.	TEXT	VARCHAR2(60)	NULL
LASTUPDATE	None	The date and time that the calculated field, filter or global edit was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The owner of the calculated field, filter or global edit.	TEXT	VARCHAR2(20)	NULL
SCALE	None	The number of decimal places for a numeric calculated field or global edit expression.	INTE	NUMBER(10, 0)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TABLE_TYPE	None	The table type upon which the calculated field, filter or global edit is based. Valid values are: ACT (Activity), BSA (Baseline Activity), BSU (Baseline Usage), CDR (Code Data), PRJ (Project Directory), PSU (Project Summary Usage), ASG (Resource Assignment), REL (Relationship), AVL (Resource Availability), CST (Resource Cost), RES (Resource Data), RSL (Resource Escalation), USE (Resource Usage, RSK (Risk Detail), and SUB (Subproject). Note that PRJ and SUB are not valid for global edits.	TEXT	VARCHAR2(30)	NOT NULL
TYPE	None	The data type of the calculated field, filter or global edit expression. Valid values are: BOOL (Logical), DATE (Date), DBLE (Decimal), DURA (Duration), FDAT (Finish Date), INTE (Integer), or TEXT (Character). Note that filter expressions will always be of type BOOL.	TEXT	VARCHAR2(4)	NOT NULL
USR_ID	None	If empty, indicates that the calculated field, filter, or global edit is shared by all users.	TEXT	VARCHAR2(20)	NULL

Table OPP_CTX - Crosstable Export Definitions

The Crosstable Export Definitions table stores crosstable export definitions.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CTX_ID	None	The unique name of a crosstable export definition.	None	VARCHAR2(60)	NOT NULL
CTX_UID	None	The unique identifier for a crosstable export definition record.	None	VARCHAR2(22)	NOT NULL
LASTUPDATE	None	The date and time that the crosstable export definition was last updated.	None	DATE	NOT NULL
OWNER_ID	None	The owner of a crosstable export definition.	None	VARCHAR2(20)	NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	None	NUMBER(10, 0)	NOT NULL
SETTINGS	None	The various settings that make up a crosstable export definition.	None	LONG	NOT NULL
USR_ID	None	If empty, indicates that the crosstable export definition is shared by all users.	None	VARCHAR2(20)	NULL

Table OPP_RCD - Reporting Calendars

The Reporting Calendar Details table contains reporting calendar definitions. Reporting calendars are system-defined items. The data dictionary table identifier for table OPP_RCD is RCD.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CUT_DATE	None	A date period in a reporting calendar.	FDAT	DATE	NULL
CUT_LABEL	None	The label for a date period in a reporting calendar.	TEXT	VARCHAR2(20)	NULL
DIR_UID	None	The unique identifier that links a reporting calendar detail record to a unique reporting calendar object directory record.	GUID	VARCHAR2(22)	NOT NULL
RCD_UID	None	The unique identifier for a reporting calendar record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Table OPP_RLP - Rollup Definitions

The Rollup Definitions table stores rollup definitions. The data dictionary table identifier for table OPP_RLP is RLP.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LASTUPDATE	None	The date and time that the roll-up definition was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The owner of a rollup definition.	TEXT	VARCHAR2(20)	NOT NULL
RLP_ID	None	The name of a rollup definition, unique within each table type for which rollups may be defined.	TEXT	VARCHAR2(60)	NOT NULL
RLP_UID	None	The unique identifier for a rollup definition record.	GUID	VARCHAR2(22)	NOT NULL
ROLLUPS	None	The rollup fields contained in the rollup definition.	MEMO	LONG	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TABLE_TYPE	None	The data type for which the roll-up definition is specified. Valid values are: ACT (Activity), RES (Resource data), and CDR (Code data).	TEXT	VARCHAR2(30)	NOT NULL
USR_ID	None	The user ID of the user who last updated the record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_SCA - Code Structure Associations

The Code Structure Association table stores the mapping of all breakdown structures to other folder-level objects such as projects and resource files. The data dictionary table identifier for table OPP_SCA is SCA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
COD_NUMBER	None	The position of the code associated with a project or resource file.	INTE	NUMBER(10, 0)	NULL
COD_UID	None	The unique identifier of the code file that is associated.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	None	The unique identifier for a project object or resource file object directory record to which the code file is associated.	GUID	VARCHAR2(22)	NULL
PROMPT_TEXT	None	The prompt that is displayed for this code file instead of the code field name.	TEXT	VARCHAR2(20)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SCA_ID	None	The field name used for the associated code file, C1 through C90.	TEXT	VARCHAR2(59)	NOT NULL
SCA_UID	None	A unique identifier for a code structure association record.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The data type to which the code file is assigned. Valid values are ACT (Activity), RES (Resource Data), and PRJ (Project Level).	TEXT	VARCHAR2(30)	NULL

Table OPP_SPD - Spread Curves

The Spread Curves table contains Open Plan spread curves definitions. These spread curves are used on resource assignments. The data dictionary table identifier for table OPP_SPD is SPD.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The description of a spread curve.	TEXT	VARCHAR2(60)	NOT NULL
LASTUPDATE	None	The date and time that the spread curve was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The owner of a sort definition.	TEXT	VARCHAR2(20)	NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SP_F1	None	The first value in a spread curve.	INTE	NUMBER(3, 0)	NOT NULL
SP_F2	None	The second value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F3	None	The third value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F4	None	The fourth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F5	None	The fifth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F6	None	The sixth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F7	None	The seventh value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F8	None	The eighth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F9	None	The ninth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SP_F10	None	The tenth value in a spread curve.	INTE	NUMBER(3, 0)	NULL
SPD_ID	None	The unique name of a spread curve definition.	INTE	VARCHAR2(1)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SPD_UID	None	The unique identifier for a spread curve record.	GUID	VARCHAR(22)	NOT NULL
USR_ID	None	The user ID of the user who last updated the record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_SRT - Sort Definitions

The Sort Definitions table contains Open Plan sort definitions. The data dictionary table identifier for table OPP_SRT is SRT.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None	The sort expression.	TEXT	VARCHAR2(200)	NOT NULL
DESCRIPTION	None	The description of a sort expression.	TEXT	VARCHAR2(60)	NOT NULL
LASTUPDATE	None	The date and time that the sort definition was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The owner of a sort definition.	TEXT	VARCHAR2(20)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SRT_ID	None	The name of a sort, unique within each table type for which sorts may be defined.	TEXT	VARCHAR2(60)	NOT NULL
SRT_UID	None	The unique identifier for a sort definition record.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The table type to which the sort is applied.	TEXT	VARCHAR2(30)	NOT NULL
USR_ID	None	The user ID of the user who last updated the record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_UFV- User Favorites

The Use Favorites table stores use favorites, such as views. The data dictionary identifier for the OPP_UVFV table is UFV.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
UFV_UID	None	The unique identifier for a User Favorite record.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
USR_ID	None	The user associated with the favorite.	TEXT	NVARCHAR2(20)	NOT NULL
OBJ_UID	None	The unique identifier for the Favorite (for example, a view).	GUID	VARCHAR2(22)	NULL
OBJECT_TYPE	None	The object type of the Favorite (for example,“VUE”).	TEXT	VARCHAR2(30)	NOT NULL
TABLE_TYPE	None	The table type of the Favorite (for example, “ACT”).	TEXT	VARCHAR2(30)	NOT NULL

Table WST_AUDIT_LOG – Audit Log

The Audit Log table stores limited audit information pertaining to user activity in the EPM products.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
EVENT_DATE	None	Date / Time of event occurrence	None	DATE	NOT NULL
EVENT_ID	None	Event ID number 257 = Using Standard Authentication 258 = Using Windows Authentication 259 = User login 260 = Bad user name or password 265 = User not authorized for product 266 = Logins are disabled for product 267 = Product license count exceeded. 269 = User logoff 270 = Product terminated. Too many failed login attempts	None	NUMBER(10,0)	NOT NULL
EVENT_TYPE	None	Event Type 1 = Error 2 = Warning 3 = Information 4 = Debug	None	NUMBER(10,0)	NOT NULL
DATA	None	Event Data (For future use)	None	LONG	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
MACHINE_ID	None	Computer that generated the event	None	VARCHAR(15)	NOT NULL
PRD_UID	None	Product ID of the application that generated the event	None	Number(10,0)	NOT NULL
ROW_UID	None	Row Unique Identifier	None	VARCHAR(22)	NOT NULL
USR_ID	None	User ID of user that generated the event	None	VARCHAR(20)	NOT NULL

Table WST_DCT - Data Dictionary

The data dictionary table defines the columns for tables within Open Plan. The structure of this table itself should never be modified. If the structure of any Open Plan tables defined in the data dictionary is modified, appropriate entries or adjustments must be made to this table. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
COL_FLAGS	None		None	NUMBER(10, 0)	NOT NULL
DEL_FOREIGN_ACTION	None		None	VARCHAR(50)	NULL
FKEY_FLD_NAME	None		None	VARCHAR2(30)	NULL
FKEY_REQUIRED	None		None	NUMBER(10, 0)	NOT NULL
FKEY_TABLE	None		None	VARCHAR2(30)	NULL
FKEY_VIRTUAL	None		None	NUMBER(10, 0)	NOT NULL
FLD_NAME	None	The name of an Open Plan field.	None	VARCHAR2(30)	NOT NULL
LENGTH	None	The number of characters required for the field.	None	NUMBER(10, 0)	NOT NULL
SCALE	None	The number of decimal places required for the field.	None	NUMBER(10, 0)	NOT NULL
STRING_ID	None	The identifier of the field's string resources.	None	NUMBER(10, 0)	NOT NULL
SYS_NAME	None		None	VARCHAR2(30)	NULL
TABLE_TYPE	None	The three-character identifier of the table in which the field exists.	None	VARCHAR2(30)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
TYPE	None	The Open Plan data type of the field.	None	VARCHAR2(4)	NOT NULL
USR_NAME	None	A user-defined name to be displayed for this field, overriding the name determined by the STRING_ID	None	VARCHAR2(60)	NULL

Table WST_ENUM – PPM Enumeration Types

The PPM Enumeration Types table stores all enumeration types associated with fields in PPM tables.

This table is not used by Open Plan.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ENUM_CODE	None	Enumeration code	None	VARCHAR(4)	NOT NULL
ENUM_ORDER	None	Sort order	None	NUMBER(10,0)	NULL
FLD_NAME	None	Field name of the table	None	VARCHAR(30)	NOT NULL
STRING_ID	None	Language resource id	None	VARCHAR(30)	NOT NULL
TABLE_TYPE	None	Unique table type	None	VARCHAR(30)	NOT NULL

Table WST_LCK - Object Locks

The Locked Objects table maintains a list of items that are currently open within Open Plan. The data dictionary table identifier for table OPP_LCK is LCK.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CONTENT	None	Not used by Open Plan	TEXT	VARCHAR2(22)	NULL
CONTEXT_ID	None	Not used by Open Plan	TEXT	VARCHAR2(200)	NULL
DIR_UID	None	The unique identifier for the locked directory object. This value is not displayed in Open Plan. The object's name (DIR_ID) is displayed instead.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	None	The date and time that the lock record was last updated.	DATE	DATE	NOT NULL
LCK_UID	None	The unique identifier for an object lock record. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LOCKMODE	None	The mode in which the directory object is locked: E (Exclusive), S (Shared), R (Read Only), B (checked out to the briefcase data source), or I (checked out to the main data source).	TEXT	VARCHAR(1)	NOT NULL
MACHINE_ID	None	The machine ID of a user associated with an object lock record.	TEXT	VARCHAR2(15)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.		NUMBER(10, 0)	NOT NULL
ULI_UID	None	The identifier for the user's login, unique for each session even if for the same user.		VARCHAR(22)	NOT NULL
USR_ID	None	The ID of the user that has the object locked.	TEXT	VARCHAR2(20)	NOT NULL

Table WST_LICN - Licenses

The Licenses table contains the list of licenses installed.

This table is not used by Open Plan.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LICN_KEY	None	License Key	None	VARCHAR(255)	NOT NULL
LICN_UID	None	Unique License ID	None	VARCHAR(22)	NOT NULL
PRD_UID	None	Unique Product ID	None	NUMBER(10,0)	NOT NULL

Table WST_LTYP – License Types

The License Types table contains the list of license types.

This table is not used by Open Plan.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LTYP	None	Unique ID of License Type	None	VARCHAR(8)	NOT NULL
PRD_UID	None	Unique Product ID	None	NUMBER(10,0)	NULL
TYPE_NAME	None	Type Name	None	VARCHAR(50)	NULL

Table WST_MSG - User Messages

The User Messages table stores messages that are to be sent to users via Open Plan. Once the message has been sent, the record is removed from this table. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
MESSAGETEXT	None	The text of the Open Plan message.	None	VARCHAR2(250)	NOT NULL
MESSAGETIME	None	The date and time that an Open Plan message is to be sent.	None	DATE	NOT NULL
MSG_UID	None	The unique identifier for a user messages record.	None	VARCHAR2(22)	NOT NULL
ULI_UID	None	The identifier for the user's login, unique for each session even if for the same user.	None	VARCHAR2(22)	NOT NULL
USR_ID	None	The ID of the user to whom the message will be sent.	None	VARCHAR2(20)	NOT NULL

Table WST_TAB - Table Names

The Table Names table provides a mapping between the 3-character data dictionary table identifier and the actual database table name. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
COTYPE	None	For Deltek use only.	None	VARCHAR2(4)	NULL
DATA_OBJ_TYPE_DESCRIP	None			VARCHAR (60)	
FLAGS	None	For Deltek use only.	None	NUMBER(10, 0)	NOT NULL
PRIMARY_KEY	None	Not used by Open Plan.	None	VARCHAR2(100)	NULL
PRD_UID	None	Deltek EPM Product ID	None	NUMBER(10,0)	NULL
TABLE_NAME	None	Names of the tables defined in the Open Plan data dictionary (WST_DCT).	None	VARCHAR2(30)	NOT NULL
TABLE_TYPE	None	Table types defined in the Open Plan data dictionary (WST_DCT).	None	VARCHAR2(30)	NOT NULL
VERSION	None		None	VARCHAR2(4)	NOT NULL

Table WST_UPD - User Preference Defaults

The User Preference Defaults table stores the default values for user preference information used within Open Plan. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CAT_VALUE	None	The value of the user preference default.	None	VARCHAR2(255)	NULL
CATEGORY	None	The category for a user preference default record.	None	VARCHAR2(50)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL

Table WST_UPF - User Preference Settings

The User Preferences table stores all user preference information used within Open Plan. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CAT_VALUE	None	The value of the user preference setting.	None	VARCHAR2(255)	NULL
CATEGORY	None	The category for a user preference setting.	None	VARCHAR2(50)	NOT NULL
DIR_UID	None	The unique identifier for a project object directory record.	None	VARCHAR2(22)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL
USR_ID	None	The user to whom the user preference setting applies.	None	VARCHAR2(20)	NOT NULL

Table WST_CFG – Configuration Options

The Configurations Options table stores options used by Open Plan and may be shared by other products as well, such as Cobra and PM Compass. Options stored for MSP import and export have a CFG_TYPE (see table below) value of “MSP Integration”. Other types may be defined in the future for options related to other operations in Open Plan.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CFG_UID	None	The unique identifier for the options set.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CFG_ID	None	The name of the options set.	TEXT	VARCHAR2(60)	NOT NULL
CFG_TYPE		Type of import or export.	TEXT	VARCHAR2(30)	NOT NULL
OWNER_ID	None	The owner of the options set.	TEXT	VARCHAR2(20)	NOT NULL
USR_ID	None	The user ID of the user who last updated the record.	TEXT	VARCHAR(20)	NOT NULL
LASTUPDATE	None	The date and time that the code record was last updated.	DATE	DATE	NOT NULL
DESCRIPTION	None	The description of the options set.	TEXT	VARCHAR(254)	NOT NULL
DATA	None	The binary field used to store the options in the form of an XML schema.	MEMO	LONG	NULL

Directory Data

Table OPP_PRJ - Project Directory Details

The Project Directory Details table contains detail information on projects that are defined in the Object Directory table. The data dictionary table identifier for table OPP_PRJ is PRJ.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_CLH_ID	Default Activity Calendar	The name of the default activity calendar for the project.	TEXT	VARCHAR2(60)	NULL
ACTDATEOPT	TA - Actual Date Option	The value of the time analysis actual date option.	INTE	NUMBER(10, 0)	NULL
ACTTYPE	Default Activity Type	The default activity type for a project. Valid values are: ALAP, ASAP, Discontinuous, Effort Driven, External Subproject, Finish Milestone, Hammock, Subproject, and Start Milestone, which are stored in the database as L, N, D, R, E, F, H, P and S , respectively.	ACTT	VARCHAR2(4)	NULL
ACWP_LAB	ACWP Labor	The actual cost of work performed for labor resources assigned a project's activities.	DBLE	NUMBER(15, 2)	NULL
ACWP_MAT	ACWP Material	The actual cost of work performed for material resources assigned a project's activities.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACWP_ODC	ACWP Other Direct Cost	The actual cost of work performed for other resources assigned a project's activities.	DBLE	NUMBER(15, 2)	NULL
ACWP_QTY	Labor Actual Units	The sum of the quantities of units for the work performed by labor resources assigned to the project's activities.	DBLE	NUMBER(15, 2)	NULL
ACWP_SUB	ACWP Subcontract	The actual cost of work performed for subcontract resources assigned a project's activities.	DBLE	NUMBER(15, 2)	NULL
AFDATE	Actual Finish	A project's actual finish date.	FDAT	DATE	NULL
ANCILMODE	Ancillary File Open Mode	A project's default open mode for its ancillary files. Valid values are: Exclusive, Shared, or Read Only, which are stored in the database as E, S, or R, respectively.	TEXT	VARCHAR2(4)	NULL
ASDATE	Actual Start	A project's actual start date.	DATE	DATE	NULL
ASG_LEVEL_TYPE	Default Resource Curve	Default resource spread curve for new resource assignments.	TEXT	VARCHAR(4)	NULL
ASG_LEVEL_VALUE	Default Resource Level	Default resource level for new resource assignments.	ASGL	VARCHAR(4)	NULL
AUTOANAL	TA - Auto Time Analysis Option	Flag which indicates whether automatic time analysis is on or off.	INTE	NUMBER(10, 0)	NULL
AUTOPROGACT	Auto Progress Activities Option	Flag which indicates whether activities should be progressed during automatic progressing or not.	INTE	NUMBER(10, 0)	NULL
AUTOPROGBASE	Auto Progress Based On Option	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL
AUTOPROGCFB	Auto Progress Complete If Finished	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL
AUTOPROGEND	Auto Progress End Date	End date for progress calculations.	DATE	DATE	NULL
AUTOPROGFILT	Auto Progress Filter	Filter for progress calculations.	TEXT	VARCHAR2(50)	NULL
AUTOPROGPPC	Auto Progress PPC Option	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
AUTOPROGPSB	Auto Progress Complete If Started	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL
AUTOPROGRES	Auto Progress Resources Option	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL
AUTOPROGSTRT	Auto Progress Start Date	Start date for progress calculations.	DATE	DATE	NULL
AUTOPROGTYPE	Auto Progress Type	Flag for progress calculation option.	INTE	NUMBER(10, 0)	NULL
BAC_LAB	BAC Labor	The budget at completion for labor resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BAC_MAT	BAC Material	The budget at completion for material resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BAC_ODC	BAC Other Direct Cost	The budget at completion for other resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BAC_QTY	Labor Budget Units	The total number of labor resource units rolled up to the project level.	DBLE	NUMBER(15, 2)	NULL
BAC_SUB	BAC Subcontract	The budget at completion for subcontract resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWP_LAB	BCWP Labor	The budgeted cost of work performed for labor resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWP_MAT	BCWP Material	The budgeted cost of work performed for material resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWP_ODC	BCWP Other Direct Cost	The budgeted cost of work performed for other resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWP_QTY	Labor Earned Units	The summed earned value of labor resource units budgeted for the project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWP_SUB	BCWP Subcontract	The budgeted cost of work performed for subcontract resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWS_LAB	BCWS Labor	The budgeted cost of work scheduled for labor resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BCWS_MAT	BCWS Material	The budgeted cost of work scheduled for material resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWS_ODC	BCWS Other Direct Cost	The budgeted cost of work scheduled for other resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BCWS_QTY	Labor Scheduled Units	The budgeted quantity of labor for the work scheduled for the project from the baseline start date until time now.	DBLE	NUMBER(15, 2)	NULL
BCWS_SUB	BCWS Subcontract	The budgeted cost of work scheduled for subcontract resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
BFDATE	Baseline Finish	The baseline finish date for a project.	FDAT	DATE	NULL
BSDATE	Baseline Start	The baseline start date for a project.	DATE	DATE	NULL
CALACTCST	Cost - Calculate Actual Cost	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CALBUDCST	Cost - Calculate Budgeted Cost	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CALCCOSTBASE	Cost - Calculate Cost Based On	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CALCSTESC	Cost - Calculate Escalated Cost	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CALEVCST	Cost - Calculate Earned Value	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CALREMCST	Cost - Calculate Remaining Cost	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
CLD_UID	Calendar File	The unique identifier that links a project record to a unique calendar object directory record for the calendar file assigned to the project. This field is not displayed in Open Plan – the calendar file's name (DIR_ID) is displayed instead.	GUID	VARCHAR2(22)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
COBRA_PROGRAM	Cobra Project	This field links the name of the Cobra project associated with the IPMDAR export of an Open Plan project so that the Cobra project name is selected by default when the IPMDAR export runs.	TEXT	VARCHAR2(22)	NULL
CST_ROLLUP	Cost – Rollup Calculated Cost	Flag for cost calculation option.	INTE	NUMBER(10, 0)	NULL
DEFDURUNIT	Default Duration Unit	The default duration unit for a project.	INTE	NUMBER(10, 0)	NULL
DEFACTDUR	Default Activity Duration	The default duration assigned to a new activity in a project	INTE	NUMBER(10, 0)	NULL
DEFENDHR	Default Finish Hour	The hours in the default end time for a project.	INTE	NUMBER(10, 0)	NULL
DEFENDMN	Default Finish Minute	The minutes in the default end time for a project.	INTE	NUMBER(10, 0)	NULL
DEFSTARTHR	Default Start Hour	The hours in the default start time for a project.	INTE	NUMBER(10, 0)	NULL
DEFSTARTMN	Default Start Minute	The minutes in the default start time for a project.	INTE	NUMBER(10, 0)	NULL
DFORMAT	Current Date Format	Current date format for the project.	TEXT	VARCHAR2(50)	NULL
DIR_UID	Name	The unique identifier that links a project record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
EFDATE	Early Finish Date	The early finish date for a project.	FDAT	DATE	NULL
ESDATE	Early Start Date	The early start date for a project.	DATE	DATE	NULL
ETC_LAB	ETC Labor	The estimate to complete for labor resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
ETC_MAT	ETC Material	The estimate to complete for material resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
ETC_ODC	ETC Other Direct Cost	The estimate to complete for other resources assigned to a project's activities.	DBLE	NUMBER(15, 2)	NULL
ETC_QTY	Labor Remaining Units	The estimated labor remaining quantity for a project.	DBLE	NUMBER(15, 2)	NULL

<table>
<tr><th>Field Name</th><th>Field Label</th><th>Description</th><th>Open Plan Data Type</th><th>Oracle Data Type</th><th>Nulls</th></tr>
<tr><td>ETC_SUB</td><td>ETC Subcontract</td><td>The estimate to complete for subcontract resources assigned to a project's activities.</td><td>DBLE</td><td>NUMBER(15, 2)</td><td>NULL</td></tr>
<tr><td>EVT</td><td>Earned Value Technique</td><td>The default earned value technique for a project.<table><tr><th>Valid Values</th><th>Stored in the Open Plan Database as…</th></tr><tr><td>Level of Effort</td><td>A</td></tr><tr><td>Percent Complete</td><td>C</td></tr><tr><td>50-50</td><td>E</td></tr><tr><td>0-100</td><td>F</td></tr><tr><td>100-0</td><td>G</td></tr><tr><td>User-Defined Percentage</td><td>H</td></tr><tr><td>Planning Package</td><td>K</td></tr><tr><td>Resource % Complete</td><td>L</td></tr><tr><td>Steps</td><td>S</td></tr></table></td><td>EVTE</td><td>VARCHAR2(2)</td><td>NULL</td></tr>
<tr><td>HARDZERO</td><td>ETC Subcontract</td><td>Flag for resource scheduling option.</td><td>INTE</td><td>NUMBER(10, 0)</td><td>NULL</td></tr>
<tr><td>LFDATE</td><td>Late Finish Date</td><td>The project's late finish date, calculated by time analysis.</td><td>FDAT</td><td>DATE</td><td>NULL</td></tr>
<tr><td>LSDATE</td><td>Late Start Date</td><td>The late start date for a project.</td><td>DATE</td><td>DATE</td><td>NULL</td></tr>
<tr><td>MEAN_EF</td><td>Mean Early Finish</td><td>The mean early finish date for the project.</td><td>FDAT</td><td>DATE</td><td>NULL</td></tr>
<tr><td>MINCALCDU</td><td>Min. Calculated Duration Unit</td><td>The minimum calculated duration unit for the project.</td><td>INTE</td><td>NUMBER(10, 0)</td><td>NULL</td></tr>
<tr><td>MINTOTFT</td><td>Min. Total Float</td><td>The minimum total float for the project</td><td>DURA</td><td>VARCHAR2(15)</td><td>NULL</td></tr>
</table>

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
MNPERDAY	Minutes per Day	Minutes per day for the project.	INTE	NUMBER(10, 0)	NULL
MNPERMON	Minutes per Month	Minutes per month for the project.	INTE	NUMBER(10, 0)	NULL
MNPERWK	Minutes per Week	Minutes per week for the project.	INTE	NUMBER(10, 0)	NULL
MSPIMPORT	MSP Import Options	Stores the settings specified for MSP Import/Export.	TEXT	VARCHAR2(20)	NULL
MSP_GROUP_OPTS_ID	None	Stores the name of the options group set (CFG_ID from the WST_CFG table) used for a project.	TEXT	VARCHAR2(60)	NULL
MSP_IMPEXP_OPTS	None	Stores the new project-specific import/export options in the form on an XML schema.	TEXT	VARCHAR2(2000)	NULL
MULTIEND	TA – Multiple-End Option	Flag for time analysis option.	INTE	NUMBER(10, 0)	NULL
NRISKSIMULS	Risk - Number of Simulations	Flag for risk analysis option.	INTE	NUMBER(10, 0)	NULL
OPCLIENT	Client	The name of the client.	TEXT	VARCHAR2(60)	NULL
OPCOMPANY	Company	The name of the company.	TEXT	VARCHAR2(60)	NULL
OPMANAGER	Project Manager	The name of the project manager.	TEXT	VARCHAR2(60)	NULL
OPSTAT	Project Status	The computed status of a project. Valid value are Planned, In Progress and Complete. These values are stored in the database as 0, 1, and 2, respectively.	ACTS	VARCHAR2(4)	NULL
OUTOFSEQ	TA - Out-of-Sequence Option	Flag for time analysis option.	INTE	NUMBER(10, 0)	NULL
PCOMPLETE	Percent Complete	The project’s percent complete.	INTE	NUMBER(10, 0)	NULL
PM_EMAIL	Project Manager E-mail	The project manager's email address.	TEXT	VARCHAR2(60)	NULL
PRIORITY1	RS - Priority 1 Option	Flag for resource scheduling option.	TEXT	VARCHAR2(60)	NULL
PRIORITY2	RS - Priority 2 Option	Flag for resource scheduling option.	TEXT	VARCHAR2(60)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PRIORITY3	RS - Priority 3 Option	Flag for resource scheduling option.	TEXT	VARCHAR2(60)	NULL
PRJ_FLAG	Project Flag		INTE	NUMBER(10, 0)	NULL
PROGPRIO	RS - In Progress Priority Option	Flag for resource scheduling option.	INT	NUMBER(10, 0)	NULL
PROJSTATUS	Project Phase	The project status. Valid values are: Proposed, Open, and Closed. These values are stored in the Open Plan database as P, O, and C, respectively.	PRJS	VARCHAR2(4)	NULL
PROJTYPE	Project Type		TEXT	VARCHAR2(20)	NULL
RCL_ID	Reporting Calendar	The name of the default reporting calendar for the project.	TEXT	VARCHAR2(20)	NULL
RDS_UID	Resource File	The unique identifier that links a project record to a unique resource object directory record for the resource file assigned to the project.	GUID	VARCHAR2(22)	NULL
REFDATE	Reporting Reference Date	Reporting reference date for the project.	DATE	DATE	NULL
REL_CLH_ID	Relationship Calendar	The name of the default relationship calendar for the project.	TEXT	VARCHAR2(60)	NULL
RISKSEED	Risk - Fixed Seed Option	Flag for risk analysis option.	INTE	NUMBER(10, 0)	NULL
RS_ACTDATE	RS - Ignore Actual Dates	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_ALTPRTY	RS - Use Alternate Resource	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_CONUSE	RS - Consider Usage on Higher	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_OVLLATE	RS - Force Overloaded Activities to Late Dates	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_PRIORTY	Project Priority	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RS_REPROF	RS - Limit Reprofiling to Original Level	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_SUMDATE	Project Summary Usage Date Option	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
RS_SUMMARY	Create Project Summary Usage	Flag for resource scheduling option. If True, project summary usage information will be saved when the project is saved.	INTE	NUMBER(10, 0)	NULL
RSK_CALSD	Risk - Use Activity Calendar	Flag for risk analysis option.	INTE	NUMBER(10, 0)	NULL
SCHMETHOD	RS - Scheduling Method Option	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
SDEV_EF	Std. Deviation of Early Finish	The early finish standard deviation for the project	DURA	VARCHAR2(10)	NULL
SFDATE	Scheduled Finish Date	The scheduled finish date for a project.	FDAT	DATE	NULL
SMOOTHING	RS - Smoothing Option	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
SSDATE	Schedule Start Date	The schedule start date for a project.	DATE	DATE	NULL
STARTDATE	Start Date	The start date for a project.	DATE	DATE	NULL
STARTPC	User Defined EVT Split %	If the User-Defined Percentage is selected as the default earned value technique for the project, this field stores the percentage that is earned at the start of the activity.	INTE	NUMBER(10, 0)	NULL
STARTVIEW	Startup View	The view that is opened when a project is opened.	TEXT	VARCHAR2(20)	NULL
STATDATE	Time Now	The time now date for a project.	DATE	DATE	NULL
TA_BEFORE_RK	Risk - Time Analyze Before Risk Analysis	Flag for risk analysis option.	INTE	NUMBER(10, 0)	NULL
TA_SUBEND	TA - Subproject End Activity	Flag for time analysis option.	INTE	NUMBER(10, 0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
TA_SUMMARY	TA - Show Summary Dates Option	Flag for time analysis option.	INTE	NUMBER(10, 0)	NULL
TARGCOST	Project Target Cost	Target cost for the project.	DBLE	NUMBER(15, 2)	NULL
TFDATE	Target Finish Date	The target finish date for a project.	FDAT	DATE	NULL
TFTYPE	Target Finish Type	The target finish type for a project.	TARG	VARCHAR2(4)	NULL
TIMEUNIT	RS - Scheduling Interval	Flag for resource scheduling option.	INTE	NUMBER(10, 0)	NULL
TOTACT	Total Activities	The total number of activities in the project.	INTE	NUMBER(10, 0)	NULL
TOTACTCOM	Total Completed Activities	The number of completed activities in the project.	INTE	NUMBER(10, 0)	NULL
TOTACTPRG	Total Activities in Progress	The number of in-progress activities in the project.	INTE	NUMBER(10, 0)	NULL
TOTRELSHP	Total Relationships	The number of relationships in the project.	INTE	NUMBER(10, 0)	NULL
TOTRESO	Total Resource Assignments	The number of resource assignments in the project.	INTE	NUMBER(10, 0)	NULL
TSDATE	Target Start Date	A project's target start date.	DATE	DATE	NULL
TSTYPE	Target Start Type	A project's target start type.	TARG	VARCHAR2(4)	NULL

Table OPP_RDS - Resource Structure Directory Details

The Resource Structure Directory Details table stores information used to define the structure of a resource structure. The data dictionary table identifier for table OPP_RDS is RDS.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ANCILMODE	None	A resource file's default open mode for its ancillary files. Valid values are: Exclusive, Shared, or Read Only, which are stored in the database as E, S, or R, respectively.	TEXT	VARCHAR2(4)	NULL
CLD_UID	None	The unique identifier that links a resource file record to a unique calendar object directory record for	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		the calendar file assigned to the resource file.			
DIR_UID	None	The unique identifier that links a resource structure record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
STARTVIEW	None	The view that is opened when a resource file is opened.	TEXT	VARCHAR2(20)	NULL

Table WST_COD - Code Structure Directory Details

The Code Structure Directory Details table stores information used to define the structure of a code breakdown structure. The data dictionary table identifier for table WST_COD is COD.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None		None	VARCHAR2(60)	NOT NULL
DIR_UID	None	The unique identifier that links a code structure record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
PROMPT_TEXT	None	The default prompt text that is displayed when the code file is attached to a project or resource file.	TEXT	VARCHAR2(20)	NULL
STARTVIEW	None	The view that is opened when a code breakdown structure is opened.	TEXT	VARCHAR2(20)	NULL
TH_FLAGS	None		TEXT	VARCHAR2(10)	NULL

Table WST_DIR - Object Directory

The Object Directory table is the master directory for all project-related data items. The data dictionary table identifier for table WST_DIR is DIR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The description of a directory object.	TEXT	VARCHAR2(125)	NULL
DIR_ID	None	The name of a project, code, calendar, resource, or view object. In Open Plan, these objects are often referred to as "files".	TEXT	VARCHAR2(22)	NOT NULL
DIR_UID	None	The unique identifier for an object directory record.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	None	The date and time that the directory object record was last updated.	DATE	DATE	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
OPENMODE	None	The mode in which the object is to be opened: Exclusive, Shared, or Read Only. The values are stored in the database as E, S, and R, respectively.	TEXT	VARCHAR2(1)	NOT NULL
OWNER_ID	None	The user ID of the object's owner.	TEXT	VARCHAR2(20)	NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TABLE_TYPE	None	The data type of the directory object. Valid values are: CLD (calendar file), COD (code file), PRJ (project file), RCL (reporting calendar), RDS (resource file), and VUE (view file).	TEXT	VARCHAR2(30)	NOT NULL
USR_ID	None	The user ID of the last user to update the directory object record.	TEXT	VARCHAR2(20)	NOT NULL

Security Data

Table WST_ACL - Object Access Rights

The Object Access Control table stores access control information for items in the object folder table. The data dictionary table identifier for table WST_ACL is ACL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACL_UID	None	The unique identifier for an access control record	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	None	The unique identifier for the directory object record to which the access control record applies.	GUID	VARCHAR2(22)	NOT NULL
GRP_ID	None	The unique identifier of a group.	TEXT	VARCHAR2(20)	NULL
READONLY	None	Determines whether the user, group, or role combination is restricted to opening the directory object in read only mode.	INTE	NUMBER(10, 0)	NOT NULL
ROL_ID	None	The unique identifier of a role.	TEXT	VARCHAR2(20)	NULL
USR_ID	None	The unique identifier of a user.	TEXT	VARCHAR2(20)	NULL

Table WST_APPSERVERS

The WST_APPSERVERS table stores the application servers.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
APPSERVER	None	App Sesrver		NVARCHAR2(80)	NOT NULL

Table WST_GRP - Group Definitions

The Group Definitions table contains definitions of valid groups. The data dictionary table identifier for table WST_GRP is GRP.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CTRL_COUNTRY	None			NVARCHAR(40)	
DESCRIPTION	None	The description of a group.	TEXT	VARCHAR2(60)	NULL
GRP_ID	None	The unique identifier of a group.	TEXT	VARCHAR2(20)	NOT NULL
MANAGER	None	The group's manager.	TEXT	VARCHAR2(20)	NULL
ROL_ID	None	The default role for a group.	TEXT	VARCHAR2(20)	NULL

Table WST_LIC - License Exceptions

The License Exceptions table contains information about license exceptions. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LIC_USERS	None		None	NUMBER(10, 0)	NOT NULL
LICENSE	None		None	VARCHAR2(32)	NOT NULL
LOGGED_USERS	None	The number of logged in users when the exception occurred.	None	NUMBER(10, 0)	NOT NULL
LOGINTIME	None	A user's last login date and time.	None	DATE	NOT NULL
MACHINE_ID	None	The machine ID of a currently logged in user.	None	VARCHAR2(20)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL
USR_ID	None	The unique identifier for a user.	None	VARCHAR2(20)	NOT NULL

Table WST_PFA - Product Function Group Access Rights

Access rights to a product function group are assigned for each role that the administrator has defined. These access rights are stored in the Product Function Group Access Rights table. Records in this table represent the intersection of a Role and a Product Function group and the specified access rights.

Each function group has 2 security attributes that can be controlled. The first attribute controls the visible state of securable objects in the function group as Visible or Not Visible. The second attribute controls the enabled state of securable objects in the function group as Enabled or Not Enabled.

The access attributes are stored in a single field using the following definitions:

FLAG_VISIBLE Mask = 0x00000001L

0 = Not Visible

1 = Visible

FLAG_ENABLED Mask = 0x00000002L

0 = Not Enabled

1 = Enabled

These flags are independent so that a user may be granted rights to execute a command or modify a data element even if it is not displayed to the user because of the state of the visibility flag. Data in the access rights table is stored in a sparse format such that records exist only for items that do not have rights granted to them. If an item is specified as both visible and enabled, then a record does not exist in the access rights table. If an item is specified as not visible and/or not enabled, then records are placed in the access rights table. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
GRP_ID	None	The unique identifier of a group.	None	VARCHAR2(20)	NULL
PFG_UID	None	The unique identifier of a product function group record.	None	VARCHAR2(22)	NOT NULL
RIGHTS	None	The rights specified by the product function group access record.	None	NUMBER(10, 0)	NOT NULL
ROL_ID	None	The unique identifier of a role.	None	VARCHAR2(20)	NULL

Table WST_PFD - Project Function Group Detail

The Product Function Group Details table stores the associations of Securable objects to function groups. This table is used by the Security Runtime module to determine the securable objects that are controlled by a particular function group. It is not referenced by the Security administration module. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
OBJ_UID	None		None	VARCHAR2(60)	NOT NULL
PFG_UID	None	The unique identifier of a product function group record.	None	VARCHAR2(22)	NOT NULL

Table WST_PFG - Product Function Group Definitions

The Product Function Group table stores a collection of one or more Securable objects that are controlled as a single item for purposes of defining access control. For instance, the user might define a function group that contains a single command. This group can then be set to define whether or not a user can execute that command. Another function group might be made up of a group of data elements that are secured as a group such as the fields that make up Activity Costs. Another function group could be defined to control the ability to update Activity Progress. This function group might contain both command and data elements to prevent the user from both displaying the progress dialog as well as changing the data elements through a spreadsheet.

As implied by their title, product function groups are defined by each product. Product function groups are also defined as applying to either a Group or a Role based on the value of the TYPE field. A TYPE field value of 0 represents a Group, a value of 1 represents a Role. Function groups defined for groups are displayed as part of the Group administration module by product. Function groups defined for Roles are displayed as part of the Role administration module by product. Product function groups are stored in the Product Function Group table by product ID. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The description of the product function group.	None	VARCHAR2(100)	NULL
PFG_ID	None		None	VARCHAR2(24)	NULL
PFG_UID	None	The unique identifier of a product function group record.	None	VARCHAR2(22)	NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL
TYPE	None		None	NUMBER(10, 0)	NULL

Table WST_PRD - Installed Deltek Products

The Installed Products table stores product identifiers that are used to determine security information for each specific product. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The name of the installed product.	None	VARCHAR2(60)	NOT NULL
FLAG	None		None	NUMBER(10, 0)	NULL
INSTALLDATE	None	The date and time the product was installed.	None	DATE	NOT NULL
INSTALLEDVER_MAJ	None	The major version number (the portion of the version number to the left of the decimal place).	None	NUMBER(10, 0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
INSTALLEDVER_MIN	None	The minor version number (the portion of the version number to the right of the decimal place).	None	NUMBER(10, 0)	NULL
INSTALLERVER_SP	None	The Service Pack number	None	NUMBER(10,0)	NULL
INSTALLERVER_REV	None	The Revision number	None	NUMBER(10,0)	NULL
LICENSE	None	The license key for the installed product.	None	VARCHAR2(32)	NULL
MESSAGETEXT	None	A message that is to be displayed to each user upon logging into the product.	None	VARCHAR2(250)	NULL
MSGEXPIRE	None	The expiration date of the login message.	None	DATE	NULL
PPM_ADMIN_COMPAT_VER	None		None	INT (Default 0)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL

Table WST_PSO - Securable Object Definition

The Securable Object Definition table stores items that products define and recognize as being able to participate in the overall security model. Securable objects can be broken down into 2 distinct groups: Data elements and Commands. Data elements represent is the persistent properties of an object that are typically stored as fields on a database table. Commands are operations that can be executed by the user - these are most analogous to the menu options available in each product. Securable objects are stored on the table by Product ID. The securable object table enumerates all items that the product exposes as securable. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
OBJ_UID	None		None	VARCHAR2(60)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	None	NUMBER(10, 0)	NOT NULL

Table WST_TAB_PRD_ACTION

This stores the registered product-specific actions (stored procedures) that are invoked when a foreign record is update or deleted.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PRD_UID	None	PRD UID		INT	NOT NULL
TABLE_TYPE	None	TABLE TYPE		VARCHAR(30)	NOT NULL
ACTION_TYPE	None	ACTION TYPE		VARCHAR(50)	NOT NULL
EXEC_PROC	None	EXEC PROC		VARCHAR(255)	NOT NULL

Table WST_ROL - Role Definitions

Roles are used to specify access rights against product function groups. The Security Role Definition table contains the definitions of the Security roles. The data dictionary table identifier for table WST_ROL is ROL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The description of a role.	TEXT	VARCHAR2(60)	NULL
MANAGER	None	The role's manager.	TEXT	VARCHAR2(20)	NULL
ROL_ID	None	The unique identifier of a role.	TEXT	VARCHAR2(20)	NOT NULL

Table WST_ULI - Logged In Users

The currently Logged In Users table contains a list of all logged in users as well as the machine ID of the workstation from which the user logged in. The product number and password are also tracked for license-checking purposes. The data dictionary table identifier for table WST_ULI is ULI.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
FLAGS	None			INT (Default 0)	NOT NULL
LICENSEREVOKED	None	Deprecated, no longer used by Open Plan.	None	NUMBER(10, 0)	NOT NULL
LOGINTIME	None	A user's last login date and time.	DATE	DATE	NOT NULL
MACHINE_ID	None	The machine ID of a currently logged in user.	TEXT	VARCHAR2(15)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	INTE	NUMBER(10, 0)	NOT NULL
ULI_UID	None	The unique identifier for a user login record.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
USR_ID	None	The login name of a currently logged in user. Records may remain in this table if a user exits the software abnormally, but they will be removed the next time the user logs in and exits normally on the same machine.	TEXT	VARCHAR2(20)	NOT NULL

Table WST_UPA - User Product Access Control

The User Product Access Control table stores information about which products user are permitted to run. The data dictionary table identifier for table WST_UPA is UPA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
FLAGS	None		INTE	NUMBER(10, 0)	NOT NULL
PRD_UID	None	The unique identifier for a Deltek product record.	INTE	NUMBER(10, 0)	NOT NULL
USR_ID	None	The unique identifier of a user.	INTE	VARCHAR2(20)	NOT NULL
LICN_UID	None	Not used by Open Plan.		VARCHAR2(22)	NULL

Table WST_USG - User-group Assignment

The User-Group Assignment table contains the associations of users to groups. The data dictionary table identifier for table WST_USG is USG.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
GRP_ID	None	The unique identifier of a group.	TEXT	VARCHAR2(20)	NOT NULL
USG_UID	None	The unique identifier for a user-group association record.	GUID	VARCHAR2(22)	NULL
USR_ID	None	The unique identifier of a user.	TEXT	VARCHAR2(20)	NOT NULL

Table WST_USR - User Definitions

The User Definitions table contains definitions of valid users. The data dictionary table identifier for table WST_USR is USR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_COMPARE	None		TEXT	VARCHAR2(4)	NULL
ACT_FIELD	None		TEXT	VARCHAR2(30)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_RULE	None		INTE	NUMBER(10, 0)	NULL
ACT_VALUE	None		TEXT	VARCHAR2(30)	NULL
ACTIVE	None			INT (Default 1)	NOT NULL
ADDRESS1	None	The user's address.	TEXT	VARCHAR2(60)	NULL
ADDRESS2	None	Second line in the user's address.	TEXT	VARCHAR2(60)	NULL
ALTMANAGER	None	The user's alternate manager.	TEXT	VARCHAR2(20)	NULL
AUTHTYPE	None	Authentication Type		INT (Default 0)	NOT NULL
CITY	None	The user's city.	TEXT	VARCHAR2(40)	NULL
COMMENTS	None	Comments for the user record.	TEXT	VARCHAR2(255)	NULL
COUNTRY	None	The user's country.	TEXT	VARCHAR2(40)	NULL
DELEGATE				NVARCHAR (20)	
DEPARTMENT	None	The user's department.	TEXT	VARCHAR2(60)	NULL
DESCRIPTION	None	A description of the user's account.	TEXT	VARCHAR2(60)	NULL
DOMAIN	None	The users Windows Active Directory Domain name	TEXT	VARCHAR2(80)	NULL
EMAIL	None	The user's email address.	TEXT	VARCHAR2(60)	NULL
FAILED_LOGINS	None			INT (Default 0)	NOT NULL
FAX	None	The user's fax number.	TEXT	VARCHAR2(20)	NULL
FILTER	None		TEXT	VARCHAR2(10)	NULL
FIRST_NAME	None	The user's first name.	TEXT	VARCHAR2(50)	NULL
ISOUTOFOFFICE	None			NVARCHAR (1), DEFAULT 'N'	NOT NULL
LAST_NAME	None	The user's last name.	TEXT	VARCHAR2(50)	NULL
LOCATION	None	The user's location.	TEXT	VARCHAR2(60)	NULL
MANAGER	None	The user's manager.	TEXT	VARCHAR2(20)	NULL
PASSWD	None	The user's encrypted password.	TEXT	VARCHAR2(178)	NULL
PWD_LASTUPDATE	None	The last date of password change.		DATE (Default, current date)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PHONE	None	The user's phone number.	TEXT	VARCHAR2(20)	NULL
ROL_ID	None	The users default Role.	TEXT	VARCHAR(20)	NULL
STATE	None	The user's state.	TEXT	VARCHAR2(40)	NULL
USR_ID	None	The user's ID.	TEXT	VARCHAR2(20)	NOT NULL
ZIP	None	The user's zip code	TEXT	VARCHAR2(20)	NULL

Table WST_USR_CTRL_COUNTRY

The USR_CTRL_COUNTRY table stores the countries that are allowed to be controlled by users.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CTRL_COUNTRY	None	CTRL COUNTRY		NVARCHAR(40)	NOT NULL

View Data

Table OPP_BRS - Bar Set Definitions

The Bar Set Definition table stores bar set definitions used by barchart views. The data dictionary table identifier for table OPP_BRS is BRS.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BRS_ID	None	The unique name of a bar set.	TEXT	VARCHAR2(32)	NOT NULL
BRS_UID	None	The unique identifier of a bar set.	GUID	VARCHAR2(22)	NOT NULL
DATA	None	The binary data used by a bar set.		LONG RAW	NULL
DATALEN	None	Length of the data in the DATA field	INTE	NUMBER(10, 0)	NULL
DESCRIPTION	None	The description of a bar set.	TEXT	VARCHAR(60)	NULL
LASTUPDATE	None	The date and time that the bar set record was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The user ID of the bar set's owner.	TEXT	VARCHAR2(20)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TYPE	None	Flag used to determine the type of bar set, Activity or Multi-table. These	INTE	NUMBER(10, 0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		values are stored in the Open Plan database as 0 and 1, respectively.			
USR_ID	None	The user ID of the last user to update the bar set.	TEXT	VARCHAR2(20)	NOT NULL
VERSION	None		INTE	NUMBER(10, 0)	NULL

Table OPP_EMF - Metafiles

The Metafile Data table stores the list metafile names that are used by title blocks and bar charts. The data dictionary table identifier for table OPP_EMF is EMF.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None	The binary field used to store enhanced metafiles.		LONG RAW	NULL
DATALEN	None	Length of the data in the DATA field.	INTE	NUMBER(10, 0)	NULL
DESCRIPTION	None	The description of a metafile.	TEXT	VARCHAR2(60)	NULL
EMF_ID	None	The name of a metafile.	TEXT	VARCHAR2(20)	NOT NULL
SHAPE	None	The shape of the metafile. Used to help categorize metafiles into manageable groups.	TEXT	VARCHAR2(32)	NULL

Table OPP_VLD - View Layout Data

The View Layout table stores information that results from the intersection of a view and the data that is being viewed (for example, the box placements on a network drawing or the collapse/expand settings of a grouped spreadsheet).

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None	The view layout data.	None	LONG RAW	NOT NULL
DIR_UID	None	The unique identifier of the directory object for which the view layout data is being stored.	None	VARCHAR2(22)	NOT NULL
VUE_UID	None	The unique identifier of the view for which the view layout data is being stored.	None	VARCHAR2(22)	NOT NULL

Table OPP_VTB - View Title Blocks

The Title Block table stores information relating to title block definitions used by views. The data dictionary table identifier for table OPP_VTB is VTB.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None	The binary data used by a title block.	None	LONG RAW	NULL
DATALEN	None	The length of the data in the DATA field.	INTE	NUMBER(10, 0)	NULL
DESCRIPTION	None	The description of a view title block.	TEXT	VARCHAR2(60)	NULL
LASTUPDATE	None	The date and time that the title block record was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The user ID of the title block's owner.	TEXT	VARCHAR2(20)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency.	INTE	NUMBER(10, 0)	NOT NULL
USR_ID	None	The last user to update the title block.	TEXT	VARCHAR2(20)	NOT NULL
VTB_ID	None	The unique name of a title block.	TEXT	VARCHAR2(20)	NOT NULL
VTB_UID	None	The unique identifier of a title block.	GUID	VARCHAR(22)	NOT NULL

Table OPP_VUE - View Data

The View Data table stores sort, filter, and title block properties for each defined view. The data dictionary table identifier for table OPP_VUE is VUE.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BRS_ID	None	The name of a bar set assigned to a view.	TEXT	VARCHAR2(20)	NULL
DATA	None	The binary data used by a view.	None	LONG RAW	NULL
DATA_TYPE	None		TEXT	VARCHAR2(4)	NULL
DATALEN	None	Length of the data in the DATA field.	INTE	NUMBER(10, 0)	NULL
DIR_UID	None	The unique identifier that links a view data record to a unique view object directory record.	GUID	VARCHAR2(22)	NOT NULL
ETYPE	None	The type of view. Valid values are B (Barchart), C (Code View), H (Resource Histogram), K (Risk Histogram), M (Multi-table Spreadsheet), N (Network View), R (Resource View), S (Spreadsheet), X (Multi-table Barchart), and Y (Calendar View)	VIEW	VARCHAR2(4)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
FLT_ID	None	The filter name or expression applied to a view.	TEXT	VARCHAR2(1024)	NULL
SRT_ID	None	The sort name or expression applied to a view.	TEXT	VARCHAR2(1024)	NULL
VERSION	None		INTE	NUMBER(10, 0)	NULL
VTB_ID	None	The name of a title block assigned to a view.	TEXT	VARCHAR2(20)	NULL

User Data

Baseline Data

Table OPP_BAS - Project Baseline Directory

The Project Baseline Directory stores information used to define baselines that have been created on a project. The data dictionary table identifier for table OPP_BAS is BAS.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_ID	None	The name of a baseline, unique within a project.	TEXT	VARCHAR2(10)	NOT NULL
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL
BASETYPE	None	A flag indicating the baseline type (Early, Late, or Scheduled). These values are stored in the Open Plan database as 0, 1, 2, respectively.	INTE	NUMBER(10, 0)	NOT NULL
DESCRIPTION	Descriptio n	The description of a baseline.	TEXT	VARCHAR2(60)	NULL
DIR_UID	None	The unique identifier that links a baseline record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
FILTER	None	The filter used when updating a baseline.	TEXT	VARCHAR2(1024)	NULL
FLAGS	None	Specifies the options that were selected on the baseline update dialog.	INTE	NUMBER(10, 0)	NULL
LASTUPDATE	None	The date and time that the baseline was last updated.	DATE	DATE	NOT NULL
OWNER_ID	None	The user ID of the baseline's owner.	TEXT	VARCHAR2(20)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SOURCE_DIR_ID	None	Used to store the name of the separate source project that was used to create and/or update a baseline in OP.	TEXT	NVARCHAR(22)	NULL
USR_ID	None		TEXT	VARCHAR2(20)	NOT NULL

Table OPP_BCR - Baseline Activity Calculated Results

The Baseline Activity Calculated Results table contains data that is calculated by the scheduling and cost calculation modules. The data dictionary table identifier for table OPP_BCR is BCR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACTIVEINDEX	Probability of Being Active	Indicates the percentage of times the activity was active in a conditional branch during risk analysis.	DBLE	NUMBER(15,2)	NULL
ACWP_LAB	ACWP Labor	The actual cost of work performed for labor resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ACWP_MAT	ACWP Material	The actual cost of work performed for material resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ACWP_ODC	ACWP Other Direct Cost	The actual cost of work performed for other resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ACWP_QTY	Labor Actual Units	The quantity of units for the work performed by labor resources assigned to the activity.	DBLE	NUMBER(15, 2)	NULL
ACWP_SUB	ACWP Subcontract	The actual cost of work performed for subcontract resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BAC_LAB	BAC Labor	The budget at completion for labor resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BAC_MAT	BAC Material	The budget at completion for material resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BAC_ODC	BAC Other Direct Cost	The budget at completion for other resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAC_QTY	Labor Budget Units	The total number of labor resource units budgeted for the activity in the baseline.	DBLE	NUMBER(15, 2)	NULL
BAC_SUB	BAC Subcontract	The budget at completion for subcontract resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL
BCWP_LAB	BCWP Labor	The budgeted cost of work performed for labor resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWP_MAT	BCWP Material	The budgeted cost of work performed for material resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWP_ODC	BCWP Other Direct Cost	The budgeted cost of work performed for other resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWP_QTY	Labor Earned Units	The earned value of labor resource units budgeted for the activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWP_SUB	BCWP Subcontract	The budgeted cost of work performed for subcontract resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWS_LAB	BCWS Labor	The budgeted cost of work scheduled for labor resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWS_MAT	BCWS Material	The budgeted cost of work scheduled for material resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWS_ODC	BCWS Other Direct Cost	The budgeted cost of work scheduled for other resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
BCWS_QTY	Labor Scheduled Units	The budgeted quantity of labor for the work scheduled for the project from the baseline start date until time now.	DBLE	NUMBER(15, 2)	NULL
BCWS_SUB	BCWS Subcontract	The budgeted cost of work scheduled for subcontract resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BSA_UID	Activity ID	The unique identifier of a baseline activity record.	GUID	VARCHAR2(22)	NOT NULL
COMP_RS_C	Sched. Actions	Indicates if an activity in a baseline was split, stretched, or reprofiled. Valid values are: Normal, Splittable, Stretchable, Reprofilable, and Immediate. These values are stored in the Open Plan database as null, P, T, R, and I, respectively.	RSCL	VARCHAR2(4)	NULL
COMPSTAT	Computed Status	The computed status of an activity in a baseline. Valid value are Planned, In Progress and Complete. These values are stored in the database as 0, 1, and 2, respectively.	ACTS	VARCHAR2(4)	NULL
CRITICAL	Critical Flag	The critical flag for an activity in a baseline. Valid values are: Not Critical, Critical, Most Critical, and Controlling Critical. These values are stored in the Open Plan database as 0, 1, 2, and 3, respectively.	CRIT	VARCHAR2(4)	NULL
CRITINDEX	Probability of Being Critical	The criticality index for an activity in a baseline.	INTE	NUMBER(10, 0)	NULL
DELAYRES_UID	Delaying Res.	The unique identifier of the delaying resource for an activity in a baseline.	GUID	VARCHAR2(22)	NULL
DIR_UID	Name	The unique identifier that links the baseline activity calculated results record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
EFDATE	Early Finish	The early finish date for an activity in a baseline.	FDAT	DATE	NULL
ESDATE	Early Start	The early start date for an activity in a baseline.	DATE	DATE	NULL
ETC_LAB	ETC Labor	The estimate to complete for labor resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ETC_MAT	ETC Material	The estimate to complete for material resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ETC_ODC	ETC Other Direct Cost	The estimate to complete for other resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ETC_QTY	Labor Remaining Units	The estimated labor remaining quantity for an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
ETC_SUB	ETC Subcontract	The estimate to complete for subcontract resources assigned to an activity in a baseline.	DBLE	NUMBER(15, 2)	NULL
FEDATE	Earliest Feasible Start	The earliest feasible start date for an activity in a baseline.	DATE	DATE	NULL
FINFREEFLT	Finish Free Float	The finish free float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
FINTOTFLT	Finish Total Float	The finish total float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
FREEFLOAT	Free Float	The free float of an activity in a baseline.	DURA	VARCHAR2(10)	NULL
LFDATE	Late Finish	The late finish date for an activity in a baseline.	FDAT	DATE	NULL
LOGICFLAG	Activity Logic Flag	The activity logic flag for an activity in a baseline. Valid values are: Start Activity, None, End Activity, Start and Finish Activity, and Isolated, which are stored in the database as S, null, F, SF, and I, respectively.	LOGI	VARCHAR2(4)	NULL
LSDATE	Late Start	The late start date for an activity in a baseline.	DATE	DATE	NULL
MEAN_EF	Mean Early Finish	The mean early finish date for an activity in a baseline.	FDAT	DATE	NULL
MEAN_ES	Mean Early Start	The mean early start date for an activity in a baseline.	DATE	DATE	NULL
MEAN_FF	Mean Free Float	The mean free float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
MEAN_LF	Mean Late Finish	The mean late finish date for an activity in a baseline.	FDAT	DATE	NULL
MEAN_LS	Mean Late Start	The mean late start date for an activity in a baseline.	DATE	DATE	NULL
MEAD_OD	Mean Activity of Duration	The mean value of activity duration.	DURA	VARCHAR2(10)	NULL
MEAN_TF	Mean Total Float	The mean total float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
MEAN_TN_EF	Mean Duration of Time Now to Early Finish	The mean value of duration between Time Now and the Early Finish date.	DURA	VARCHAR2(10)	NULL
OUTOFSEQ	Out of Sequence	Flag on the activity record indicating if the activity has been progressed out of sequence.	BOOL	VARCHAR2(4)	NULL
REM_DUR	Computed Remaining Dur.	The remaining portion of the original duration for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
RES_DATE	First Usage	The date upon which an activity in a baseline first used a resource	DATE	DATE	NULL
RS_FLOAT	Sched. Float	The scheduled float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
SCHED_DUR	Sched. Duration	The scheduled duration for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
SDEV_EF	Std. Deviation of Early Finish	The early finish standard deviation for an activity in a baseline.	FDAT	VARCHAR2(10)	NULL
SDEV_ES	Std. Deviation of Early Start	The early start standard deviation for an activity in a baseline.	DATE	VARCHAR2(10)	NULL
SDEV_FF	Std. Deviation of Free Float	The free float standard deviation for an activity in a baseline	DURA	VARCHAR2(10)	NULL
SDEV_LF	Std. Deviation of Late Finish	The late finish standard deviation for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
SDEV_LS	Std. Deviation of Late Start	The late start standard deviation for an activity in a baseline.	DATE	VARCHAR2(10)	NULL
SDEV_OD	Std. Deviation of Duration	The standard deviation of duration for an activity calculated by Risk Analysis.	DURA	VARCHAR2(10)	NULL
SDEV_TF	Std. Deviation of Total Float	The total float standard deviation for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
SDEV_TN_EF	Std. Deviation of Time Now to Early Finish	The standard deviation of span from Time Now to the Early Finish date for an activity calculated by Risk Analysis.	DURA	VARCHAR2(10)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SFDATE	Sched. Finish	The scheduled finish date for an activity in a baseline.	FDAT	DATE	NULL
SSDATE	Sched. Start	The scheduled start date for an activity in a baseline.	DATE	DATE	NULL
SSINDEX	Sensitivity Index	Indicates the schedule sensitivity index as defined by PMBOK and calculated by risk analysis.	DBLE	NUMBER(15,2)	NULL
TOTALFLOAT	Total Float	The total float for an activity in a baseline.	DURA	VARCHAR2(10)	NULL

Table OPP_BSA - Baseline Activity Details

The Baseline Activity Details table contains information copied from the Activity Details table for baseline comparison purposes. The data dictionary table identifier for table OPP_BSA is BSA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_PROBABILITY	Probability of Occurrence		DBLE	NUMBER(15,2)	NULL
ACT_TYPE	Activity Type	The activity type stored in a baseline for an activity. Valid values are: ALAP, ASAP, Discontinuous, Effort Driven, External Subproject, Finish Milestone, Hammock, Subproject, and Start Milestone, which are stored in the database as L, N, D, R, E, F, H, P and S , respectively. In multi-projects where not all external subprojects are opened, the types Foreign Project, Foreign Subproject, and Foreign Activity may be present, but these types are not stored in the database. If exported, the short forms for Foreign Project, Foreign Subproject, and Foreign Activity are Z, Y and G, respectively.	ACTT	VARCHAR2(4)	NULL
ACT_UID	Activity ID	The unique identifier that links an activity record to a baseline activity records.	GUID	VARCHAR2(22)	NOT NULL
AFDATE	Actual Finish	The actual finish date stored in a baseline for an activity.	FDAT	DATE	NULL
ASDATE	Actual Start	The actual start date stored in a baseline for an activity.	DATE	DATE	NULL
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BFDATE	Actual Start	The baseline finish date stored in a baseline for an activity.	FDAT	DATE	NULL
BSA_ID	Baseline Activity ID	The activity ID of an baseline activity record.	TEXT	VARCHAR2(59)	NOT NULL
BSA_UID	None	The unique identifier of a baseline activity record.	GUID	VARCHAR2(22)	NOT NULL
BSDATE	Baseline Start	The baseline finish date stored in a baseline for an activity.	DATE	DATE	NULL
CLH_UID	Calendar ID	The unique identifier that links a baseline activity record to a unique calendar detail record.	GUID	VARCHAR2(59)	NULL
DESCRIPTION	Activity Desc.	The activity description stored in a baseline for an activity.	TEXT	VARCHAR2(125)	NULL
DHIGH	Pessimistic Duration	The maximum duration for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
DLOW	Optimistic Duration	The minimum duration for an activity in a baseline.	CURA	VARCHAR2(10)	NULL
DSHAPE	Duration Distribution Type	The duration distribution curve for an activity in a baseline.	DIST	VARCHAR2(4)	NULL

<table>
<tr><th>Field Name</th><th>Field Label</th><th>Description</th><th>Open Plan Data Type</th><th>Oracle Data Type</th><th>Nulls</th></tr>
<tr><td>EVT</td><td>Earned Value Technique</td><td>The earned value technique assigned to an activity in a baseline.<table><tr><th>Valid Values</th><th>Stored in the Open Plan Database as…</th></tr><tr><td>Level of Effort</td><td>A</td></tr><tr><td>Percent Complete</td><td>C</td></tr><tr><td>50-50</td><td>E</td></tr><tr><td>0-100</td><td>F</td></tr><tr><td>100-0</td><td>G</td></tr><tr><td>User-Defined Percentage</td><td>H</td></tr><tr><td>Planning Package</td><td>K</td></tr><tr><td>Resource % Complete</td><td>L</td></tr><tr><td>Steps</td><td>S</td></tr></table></td><td>EVTE</td><td>VARCHAR2(2)</td><td>NULL</td></tr>
<tr><td>MAXDUR</td><td>Max. Duration</td><td>The maximum duration of a stretchable or reprofilable activity in a baseline.</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
<tr><td>MAXSPLITS</td><td>Max. Number of Splits</td><td>The maximum number of splits possible for a splittable activity in a baseline.</td><td>INTE</td><td>NUMBER(10, 0)</td><td>NULL</td></tr>
<tr><td>MINSPLITD</td><td>Min. Split Length</td><td>The minimum size of a split for a splittable activity in a baseline.</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
<tr><td>OPKEY</td><td>Key Activity</td><td>The state of the key activity flag for an activity in a baseline.</td><td>BOOL</td><td>VARCHAR2(4)</td><td>NULL</td></tr>
<tr><td>ORIG_DUR</td><td>Original Duration</td><td>The original duration for an activity in a baseline.</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
<tr><td>PPC</td><td>Physical % Complete</td><td>The physical percent complete for an activity in a baseline.</td><td>DBLE</td><td>NUMBER(15, 2)</td><td>NULL</td></tr>
<tr><td>PRIORITY</td><td>Priority</td><td>Resource scheduling priority for an activity in a baseline.</td><td>INTE</td><td>NUMBER(10, 0)</td><td>NULL</td></tr>
</table>

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PROGTYPE	Progress Type	The progress type used by an activity in a baseline. Valid values are: Planned, Remaining Duration, Percent Complete, Elapsed Duration, and Complete. These values are stored in the Open Plan database as null, R, P, E, and C, respectively.	PROG	VARCHAR2(4)	NULL
PROGVALUE	Progress Value	The progress value for an activity in a baseline.	DURA	VARCHAR2(10)	NULL
RS_SUPRESS	Suppress Requirements	The state of the suppress resources flag for an activity in a baseline.	BOOL	VARCHAR2(4)	NULL
RSCLASS	R/S Type	The resource scheduling options for an activity in a baseline. Valid values are: Normal, Splittable, Stretchable, Reprofilable, and Immediate. These values are stored in the Open Plan database as null, P, T, R, and I, respectively.	RSCL	VARCHAR2(4)	NULL
SEP_ASG	Separate Assignments	The state of the Separate Assignments flag for an activity in a baseline.	BOOL	VARCHAR2(4)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
STARTPC	User Defined EVT Split %	When using the "User-defined percentage" earned value technique, this field contains the percentage earned at the start of the activity in a baseline.	INTE	NUMBER(10, 0)	NULL
TARGFTYPE	Target Finish Type	The target finish type for an activity in a baseline. Valid values are: None, Not Earlier Than, Not Later Than, On Target, and Fixed Target. These values are stored in the Open Plan database as null, NE, NL, ON, and FX, respectively.	TARG	VARCHAR2(4)	NULL
TARGSTYPE	Target Start Type	The target start type for an activity in a baseline. Valid values are: None, Not Earlier Than, Not Later Than, On Target, and Fixed Target. These values are stored in the Open Plan database as null, NE, NL, ON, and FX, respectively.	TARG	VARCHAR2(4)	NULL
TFDATE	Target Finish	The target finish date for an activity in a baseline.	FDAT	DATE	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
TSDATE	Target Start	The target start date for an activity in a baseline.	DATE	DATE	NULL
XFDATE	Expected Finish	The expected finish stored in a baseline for an activity	FDAT	DATE	NULL

Table OPP_BSC - Baseline Access Control List

The Baseline Access Control table stores access control list information for items in the baseline directory table. The data dictionary table identifier for table OPP_BSC is BSC.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACCESS_ID	None		TEXT	VARCHAR2(20)	NOT NULL
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL
BSC_UID	None	The unique identifier for a baseline access control record.	GUID	VARCHAR2(22)	NOT NULL
ID_TYPE	None		TEXT	VARCHAR2(2)	NOT NULL
RIGHTS	None		TEXT	VARCHAR2(5)	NOT NULL

Table OPP_BSD - Baseline Subproject Associations

The Baseline Association table contains information that associates baselines created in a master project with its external subprojects so that the baseline can be viewed and edited in the external subproject. The data dictionary table identifier for table OPP_BSD is BSD.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL
BASESEL	None	Indicates if a baseline is selected or not. The value is 1 if selected and 0 if not.	INTE	NUMBER(10, 0)	NULL
BSD_UID	None	The unique identifier of a baseline description record.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	None	The unique identifier that links a baseline record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL

Table OPP_BSU - Baseline Resource Usage

The Baseline Resource Usage table contains information copied from the resource usage table for baseline comparison purposes. The data dictionary table identifier for table OPP_BSU is BSU.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ALT_RES_UID	Alternate Res. ID	The unique identifier that links an assignment record for an activity stored in a baseline to the resource record representing the resource assignment alternate.	GUID	VARCHAR2(22)	NULL
ASG_UID	Early Finish	The unique identifier for a resource assignment in a baseline.	GUID	VARCHAR2(22)	NULL
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NOT NULL
BSA_UID	None	The unique identifier of a baseline activity record.	GUID	VARCHAR2(22)	NOT NULL
BSU_UID	None	The unique identifier of a baseline resource usage record.	GUID	VARCHAR2(22)	NOT NULL
CLH_UID	None	The unique identifier of the resource calendar for a baseline usage record.	GUID	VARCHAR2(22)	NULL
CST_CLASS	Cost Class	The cost class for a resource assignment in a baseline.	TEXT	VARCHAR2(20)	NULL
DIR_UID	None	The unique identifier that links a baseline usage record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
PALLOC_UID	Project Allocation	The unique identifier of the project directory object for which this resource was reserved.	GUID	VARCHAR2(22)	NULL
RDS_UID	None	The unique identifier that links a baseline usage record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
RES_CST	Resource Cost	The cost for resource usage in a baseline.	DBLE	NUMBER(15, 2)	NULL
RES_ESC	Resource Escalated Cost	The escalated cost for resource usage in a baseline.	DBLE	NUMBER(15, 2)	NULL
RES_UID		A unique identifier that links a baseline usage record to a resource file.	GUID	VARCHAR2(22)	NOT NULL
RES_SKL_UID		The unique identifier that links the assignment record to a resource skill record.	GUID	VARCHAR(22)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RES_USED	Res. Used	The total quantity of resource used between the start and finish dates of the baseline record.	DBLE	NUMBER(15, 2)	NULL
RFDATE	Finish Date	The end of the period during which the resource was used in the baseline.	FDAT	DATE	NULL
RSDATE	Start Date	The beginning of the period during which the resource was used in the baseline.	DATE	DATE	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Calendar Data

Table OPP_CLH - Calendar Headers

The Calendar Header table stores calendar name and description information for individual calendar definitions within a calendar file. The data dictionary table identifier for table OPP_CLH is CLH.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLH_ID	None	The name of a calendar within a calendar file, unique within each file.	HIEI	VARCHAR2(59)	NOT NULL
CLH_UID	None	A unique identifier for a calendar header record. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
DESCRIPTION	None	The description of a calendar within a calendar file.	TEXT	VARCHAR2(125)	NULL
DIR_UID	None	The unique identifier that links a calendar header record to a unique calendar object directory record.	GUID	VARCHAR2(22)	NOT NULL
INTEGRATION_ID	None	The unique identifier used to match an Open Plan calendar to its equivalent in the third-party integration product (for example, Primavera).	INTE	NUMBER(10,0)	NULL
LASTUPDATE	None	The date and time that the calendar was last updated.	DATE	DATE	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUPPRESS	None		BOOL	VARCHAR2(4)	NULL
USR_ID	None	The user ID of the user who last updated the record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_CLR - Calendar Details

The Calendar Detail table stores the working and non-working information for calendars defined in the calendar header table. The data dictionary table identifier for table OPP_CLR is CLR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLH_UID	None	The unique identifier that links a calendar detail record to the calendar header table.	GUID	VARCHAR2(22)	NOT NULL
CLR_UID	None	A unique identifier for a calendar detail record.	GUID	VARCHAR2(22)	NOT NULL
DATESPEC	None	A string that identifies a type of rule for a calendar. Valid values are: Sunday through Saturday (standard days); a date with no year (recurring holidays); a specific date.	DSPC	VARCHAR2(10)	NOT NULL
DIR_UID	None	The unique identifier that links a calendar detail record to a unique calendar object directory record.	GUID	VARCHAR2(22)	NOT NULL
OPFINISH	None	A valid time later than the OPSTART time.	SHFT	VARCHAR2(6)	NOT NULL
OPSTART	None	A valid time.	SHFT	VARCHAR2(6)	NOT NULL
OPWORK	None	Flag indicating whether the specified date period is working or non-working.	BOOL	VARCHAR2(4)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Code Data

Table OPP_CRA - Code Assignments

The Code Assignments table stores code values assigned to projects, activities, resources, and codes. The COD_NUMBER field identifies the code number to which the CDR_UID value is associated. This table is a sparse storage mechanism that contains only the records that have been assigned. The data dictionary table identifier for table OPP_CRA is CRA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID		The unique identifier that links a baseline record to a code assignment.	GUID	VARCHAR2(22)	NULL
DIR_UID		The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
FIELD_NAME			TEXT	VARCHAR2(60)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
FIELD_VALUE			CODE	VARCHAR2(22)	NOT NULL
FK_UID		The unique ID that links a code record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID			GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	File Type		TEXT	VARCHAR2(30)	NOT NULL

Table WST_CDR - Code Breakdown Definition

The Code Breakdown Definitions table contains the code elements that make up a code breakdown structure. The data dictionary table identifier for table WST_CDR is CDR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BARCOLOR	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
BARSTYLE	None	Not used by Open Plan.	None	VARCHAR2(6)	NULL
BOXCOLOR	None	Not used by Open Plan.	None	VARCHAR2(6)	NULL
BOXSTYLE	None	Not used by Open Plan.	None	VARCHAR2(6)	NULL
CA_PF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CA_PU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CA_VF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CA_VU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CC_PF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CC_PU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CC_VF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CC_VU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CDR_ID	Code ID	A unique code within a code breakdown structure.	None	VARCHAR2(59)	NULL
CDR_UID	None	The unique identifier of a code record.	GUID	VARCHAR2(22)	NOT NULL
CP_PF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CP_PU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CP_VF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
CP_VU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
DESCRIPTION	Code Description	The description of a record in a code breakdown structure.	TEXT	VARCHAR2(125)	NULL
DIR_UID	None	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
INTEGRATION_ID	None	The unique identifier used to match an Open Plan code to its equivalent in the third-party integration product (for example, Primavera).	INTE	NUMBER(10,0)	NULL
LASTUPDATE	Last Updated	The date and time that the code record was last updated.	DATE	DATE	NOT NULL
NUMCHILD	None	Not used by Open Plan.	None	NUMBER(10, 0)	NULL
PARENT	None	Not used by Open Plan.	None	VARCHAR2(59)	NULL
POSITION_NUM	Child Position		INTE	NUMBER(10, 0)	NULL
SC_PF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SC_PU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SC_VF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SC_VU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SP_PF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SP_PU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SP_VF	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SP_VU	None	Not used by Open Plan.	None	NUMBER(15, 2)	NULL
SUPPRESS	Suppress		INTE	VARCHAR2(4)	NULL
TAG	None	Not used by Open Plan.	None	VARCHAR2(20)	NULL
USR_ID	Updated By	The user ID of the last user to update the code.	TEXT	VARCHAR2(20)	NOT NULL

Explorer Data

Table OPP_EXF - Explorer Folders

The Explorer Folders table stores the data associated with folders assigned to the Open Plan Explorer. The data dictionary table identifier for table OPP_EXF is EXF.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DIR_UID	None	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NULL
EXF_ID	None	The unformatted name of the Open Plan Explorer Folder.	HIEI	VARCHAR2(60)	NOT NULL
EXF_UID	None	The unique identifier of an Open Plan Explorer folder. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None		TEXT	VARCHAR2(30)	NULL
USR_ID	None	The user to whom the Open Plan Explorer folder belongs.	TEXT	VARCHAR2(22)	NULL

Table OPP_EXI - Explorer Folder Items

The Explorer Folder Items table stores the data associated with folder items assigned to the Open Plan Explorer. The data dictionary table identifier for table OPP_EXI is EXI.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DIR_UID	None	For rows where the table type is VUE, specifies the unique identifier of the object directory item to which the view is assigned.	GUID	VARCHAR2(22)	NULL
EXF_UID	None	The unique identifier of an Open Plan Explorer folder. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NULL
EXI_UID	None	The unique identifier of an Open Plan Explorer folder item. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ITEM_TYPE	None	The type of item. Valid values are: CLD (Calendar file), COD (Code file), PJT (Project template), PRJ (Project file), RDS (Resource file), and VUE (View).	TEXT	VARCHAR2(4)	NULL
ITEM_UID	None	The unique identifier for an Open Plan Explorer item.	GUID	VARCHAR2(22)	NULL
USR_ID	None	The user to whom the item belongs. If blank, the item refers to the Projects or Open Plan Library folders.	TEXT	VARCHAR2(20)	NULL

Table OPP_EXL - Explorer Listview Fields

The Explorer Listview fields table contains the fields selectable for display in the list view of the Open Plan Explorer. This table is not defined in the Open Plan data dictionary, and its structure must not be changed.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CUSTOM_NAME	None		None	VARCHAR2(50)	NULL
FIELD_NAME	None	A field displayed in the Explorer folder's listview.	None	VARCHAR2(60)	NOT NULL
FIELD_ORDER	None	The field's position in the listview.	None	NUMBER(10, 0)	NOT NULL
GROUP_FIELD	None	The field upon which the listview items are grouped.	None	NUMBER(10, 0)	NULL
SORT_FIELD	None	The fields by which with list view items are sorted.	None	NUMBER(10, 0)	NULL
TABLE_NAME	None		None	VARCHAR2(4)	NOT NULL
USR_ID	None		None	VARCHAR2(20)	NULL
WIDTH	None	The width of the field's column in the listview.	None	NUMBER(10, 0)	NOT NULL

Table OPP_EXS - Explorer Shortcut Items

The Explorer Shortcut Items table stores the data associated with shortcuts assigned to the Open Plan Explorer. The data dictionary table identifier for table OPP_EXS is EXS.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DATA	None	The definition of the shortcut.	TEXT	VARCHAR2(512)	NOT NULL
DESCRIPTION	None	The shortcut's description.	TEXT	VARCHAR2(60)	NOT NULL
EXS_UID	None	The unique identifier of an Open Plan Explorer shortcut. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL

Project Data

Table OPP_ACR - Activity Calculated Results

The Activity Calculated Results table contains data that is calculated by the scheduling and cost calculation modules. The data dictionary table identifier for table OPP_ACR is ACR.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	None	The unique identifier for an activity record in the database.	GUID	VARCHAR2(22)	NOT NULL
ACTIVEINDEX	Probability of Being Active	Indicates the percentage of times the activity was active in a conditional branch during risk analysis.	DBLE	NUMBER(15,2)	NULL
ACWP_LAB	ACWP Labor	The actual cost of work performed for labor resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ACWP_MAT	ACWP Material	The actual cost of work performed for material resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ACWP_ODC	ACWP Other Direct Cost	The actual cost of work performed for other resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ACWP_QTY	Labor Actual Units	The quantity of units for the work performed by labor resources assigned to the activity.	DBLE	NUMBER(15, 2)	NULL
ACWP_SUB	ACWP Subcontract	The actual cost of work performed for subcontract resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BAC_LAB	BAC Labor	The budget at completion for labor resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BAC_MAT	BAC Material	The budget at completion for material resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BAC_ODC	BAC Other Direct Cost	The budget at completion for other resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BAC_QTY	Labor Budget Units	The total number of labor resource units budgeted for the activity.	DBLE	NUMBER(15, 2)	NULL
BAC_SUB	BAC Subcontract	The budget at completion for subcontract resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWP_LAB	BCWP Labor	The budgeted cost of work performed for labor resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BCWP_MAT	BCWP Material	The budgeted cost of work performed for material resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWP_ODC	BCWP Other Direct Cost	The budgeted cost of work performed for other resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWP_QTY	Labor Earned Units	The earned value of labor resource units budgeted for the activity.	DBLE	NUMBER(15, 2)	NULL
BCWP_SUB	BCWP Subcontract	The budgeted cost of work performed for subcontract resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWS_LAB	BCWS Labor	The budgeted cost of work scheduled for labor resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWS_MAT	BCWS Material	The budgeted cost of work scheduled for material resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWS_ODC	BCWS Other Direct Cost	The budgeted cost of work scheduled for other resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
BCWS_QTY	Labor Scheduled Units	The budgeted quantity of labor for the work scheduled for the project from the baseline start date until time now.	DBLE	NUMBER(15, 2)	NULL
BCWS_SUB	BCWS Subcontract	The budgeted cost of work scheduled for subcontract resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
COMP_RS_C	Sched. Actions	Calculated by resource scheduling. Indicates if the activity was split, stretched, or reprofiled. Valid values are: Normal, Splittable, Stretchable, Reprofilable, and Immediate. These values are stored in the Open Plan database as null, P, T, R, and I, respectively.	RSCL	VARCHAR2(4)	NULL
COMPSTAT	Computed Status	The computed status of an activity as determined by time analysis. Valid value are Planned, In Progress and Complete. These values are stored in the database as 0, 1, and 2, respectively.	ACTS	VARCHAR2(4)	NULL
CRITICAL	Critical Flag	The critical flag for an activity. Calculated by time analysis. Valid values are: Not Critical, Critical, Most Critical, and Controlling Critical. These values are stored in	CRIT	VARCHAR2(4)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		the Open Plan database as 0, 1, 2, and 3, respectively.			
CRITINDEX	Probability of Being Critical	The criticality index for an activity, calculated by risk analysis. Stores the number of times the activity fell on the critical path, expressed as a percentage.	INTE	NUMBER(10, 0)	NULL
DELAYRES_UID	Delaying Res.	The unique identifier of the resource that caused the activity to be delayed during resource scheduling.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier that links the activity calculated results record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
EFDATE	Early Finish	The early finish date calculated by time analysis for an activity.	FDAT	DATE	NULL
ESDATE	Early Start	The early start date calculated by time analysis for an activity.	DATE	DATE	NULL
ETC_LAB	ETC Labor	The estimate to complete for labor resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ETC_MAT	ETC Material	The estimate to complete for material resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ETC_ODC	ETC Other Direct Cost	The estimate to complete for other resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
ETC_QTY	Labor Remaining Units	The estimated labor remaining quantity for an activity.	DBLE	NUMBER(15, 2)	NULL
ETC_SUB	ETC Subcontract	The estimate to complete for subcontract resources assigned to an activity.	DBLE	NUMBER(15, 2)	NULL
FEDATE	Earliest Feasible Start	The earliest feasible start date for an activity, calculated by resource scheduling.	DATE	DATE	NULL
FINFREEFLT	Finish Free Float	The finish free float calculated by time analysis for an activity.	DURA	VARCHAR2(10)	NULL
FINTOTFLT	Finish Total Float	The finish total float calculated by time analysis for an activity.	DURA	VARCHAR2(10)	NULL
FREEFLOAT	Free Float	The free float calculated by time analysis for an activity.	DURA	VARCHAR2(10)	NULL
LFDATE	Late Finish	The late finish date calculated by time analysis for an activity.	FDAT	DATE	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LOGICFLAG	Activity Logic Flag	The activity logic flag for an activity. Indicates how this activity is related to others in the network. Valid values are: Start Activity, None, End Activity, Start and Finish Activity, and Isolated, which are stored in the database as S, null, F, SF, and I, respectively.	LOGI	VARCHAR2(4)	NULL
LSDATE	Late Start	The late start date calculated by time analysis for an activity.	DATE	DATE	NULL
MEAN_EF	Mean Early Finish	The mean early finish date calculated by risk analysis for an activity.	FDAT	DATE	NULL
MEAN_ES	Mean Early Start	The mean early start date calculated by risk analysis for an activity.	DATE	DATE	NULL
MEAN_FF	Mean Free Float	The mean free float calculated by risk analysis for an activity.	DURA	VARCHAR2(10)	NULL
MEAN_LF	Mean Late Finish	The mean late finish date calculated by risk analysis for an activity.	FDAT	DATE	NULL
MEAN_LS	Mean Late Start	The mean late start date calculated by risk analysis for an activity.	DATE	DATE	NULL
MEAD_OD	Mean Activity of Duration	The mean value of activity duration.	DURA	VARCHAR2(10)	NULL
MEAN_TF	Mean Total Float	The mean total float calculated by risk analysis for an activity.	DURA	VARCHAR2(10)	NULL
MEAN_TN_EF	Mean Duration of Time Now to Early Finish	The mean value of duration between Time Now and the Early Finish date.	DURA	VARCHAR2(10)	NULL
OUTOFSEQ	Out of Sequence	Flag on the activity record indicating if the activity has been progressed out of sequence.	BOOL	VARCHAR2(4)	NULL
REM_DUR	Computed Remaining Dur.	The remaining portion of the original duration for an activity, calculated by time analysis. Based on progress information supplied by the user or calculated during progress calculations.	DURA	VARCHAR2(10)	NULL
RES_DATE	First Usage	The date upon which an activity first used a resource, calculated by resource scheduling.	DATE	DATE	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RS_FLOAT	Sched. Float	The activity's scheduled float, calculated by resource scheduling	DURA	VARCHAR2(10)	NULL
SCHED_DUR	Sched. Duration	The activity duration calculated by resource scheduling.	DURA	VARCHAR2(10)	NULL
SDEV_EF	Std. Deviation of Early Finish	The early finish standard deviation for an activity calculated by risk analysis.	FDAT	VARCHAR2(10)	NULL
SDEV_ES	Std. Deviation of Early Start	The early start standard deviation for an activity calculated by risk analysis.	DATE	VARCHAR2(10)	NULL
SDEV_FF	Std. Deviation of Free Float	The free float standard deviation for an activity calculated by risk analysis.	DURA	VARCHAR2(10)	NULL
SDEV_LF	Std. Deviation of Late Finish	The late finish standard deviation for an activity calculated by risk analysis.	FDAT	VARCHAR2(10)	NULL
SDEV_LS	Std. Deviation of Late Start	The late start standard deviation for an activity calculated by risk analysis.	DATE	VARCHAR2(10)	NULL
SDEV_OD	Std. Deviation of Duration	The standard deviation of duration for an activity calculated by Risk Analysis.	DURA	VARCHAR2(10)	NULL
SDEV_TF	Std. Deviation of Total Float	The total float standard deviation for an activity calculated by risk analysis.	DURA	VARCHAR2(10)	NULL
SDEV_TN_EF	Std. Deviation of Time Now to Early Finish	The standard deviation of span from Time Now to the Early Finish date for an activity calculated by Risk Analysis.	DURA	VARCHAR2(10)	NULL
SFDATE	Sched. Finish	The scheduled finish date calculated by resource scheduling for an activity.	FDAT	DATE	NULL
SSDATE	Sched. Start	The scheduled start date calculated by resource scheduling for an activity.	DATE	DATE	NULL
SSINDEX	Sensitivity Index	Indicates the schedule sensitivity index as defined by PMBOK and calculated by risk analysis.	DBLE	NUMBER(15,2)	NULL
TOTALFLOAT	Total Float	The total float calculated by time analysis for an activity.	DURA	VARCHAR2(10)	NULL

Table OPP_ACT - Activity Details

The Activity Details table contains information that defines an activity and its scheduling constraints. The data dictionary table identifier for table OPP_ACT is ACT.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_ID	Activity ID	An activity ID, unique within a project.	HEIR	VARCHAR2(59)	NOT NULL
ACT_PROBABILITY	Probability of Occurrence		DBLE	NUMBER(15,2)	NULL
ACT_TYPE	Activity Type	The activity type stored in a baseline for an activity. Valid values are: ALAP, ASAP, Discontinuous, Effort Driven, External Subproject, Finish Milestone, Hammock, Subproject, and Start Milestone, which are stored in the database as L, N, D, R, E, F, H, P and S , respectively. In multi-projects where not all external subprojects are opened, the types Foreign Project, Foreign Subproject, and Foreign Activity may be present, but these types are not stored in the database. If exported, the short forms for Foreign Project, Foreign Subproject, and Foreign Activity are Z, Y and G, respectively.	ACTT	VARCHAR2(4)	NULL
ACT_UID	None	The unique identifier for an activity record in the database.	GUID	VARCHAR2(22)	NOT NULL
AFDATE	Actual Finish	The actual finish date for an activity. Value may be user-entered or calculated by progress calculations.	FDAT	DATE	NULL
ASDATE	Actual Start	The actual start date for an activity. Value may be user-entered or calculated by progress calculations.	DATE	DATE	NULL
BFDATE	Baseline Finish	The baseline finish date for an activity, based on the last baseline selection.	FDAT	DATE	NULL
BSDATE	Baseline Start	The baseline start date for an activity, based on the last baseline selection.	DATE	DATE	NULL
CLH_UID	Calendar ID	The unique identifier that links an activity record to a unique calendar record.	GUID	VARCHAR2(22)	NOT NULL
DESCRIPTION	Activity Desc.	The description of an activity.	TEXT	VARCHAR2(125)	NULL

<table>
<tr><th>Field Name</th><th>Field Label</th><th>Description</th><th>Open Plan Data Type</th><th>Oracle Data Type</th><th>Nulls</th></tr>
<tr><td>DHIGH</td><td>Pessimistic Duration</td><td>The maximum duration used by risk analysis for an activity.</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
<tr><td>DIR_UID</td><td>Project File</td><td>The unique identifier that links an activity record to a unique project object directory record.</td><td>GUID</td><td>VARCHAR2(22)</td><td>NOT NULL</td></tr>
<tr><td>DLOW</td><td>Optimistic Duration</td><td>The minimum duration used by risk analysis for an activity.</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
<tr><td>DSHAPE</td><td>Duration Distribution Type</td><td>The duration distribution curve used by risk analysis for an activity. Valid values are: None, Uniform, Normal, Beta, and Triangular. These values are stored in the Open Plan database as null, U, N, B, and T, respectively.</td><td>DIST</td><td>VARCHAR2(4)</td><td>NULL</td></tr>
<tr><td>EVT</td><td>Earned Value Technique</td><td>The earned value technique assigned to an activity.<table><tr><th>Valid Values</th><th>Stored in the Open Plan Database as…</th></tr><tr><td>Level of Effort</td><td>A</td></tr><tr><td>Percent Complete</td><td>C</td></tr><tr><td>50-50</td><td>E</td></tr><tr><td>0-100</td><td>F</td></tr><tr><td>100-0</td><td>G</td></tr><tr><td>User-Defined Percentage</td><td>H</td></tr><tr><td>Planning Package</td><td>K</td></tr><tr><td>Resource % Complete</td><td>L</td></tr><tr><td>Steps</td><td>S</td></tr></table></td><td>EVTE</td><td>VARCHAR2(2)</td><td>NULL</td></tr>
<tr><td>LASTUPDATE</td><td>Last Updated</td><td>The date and time that the activity was last updated.</td><td>DATE</td><td>DATE</td><td>NOT NULL</td></tr>
<tr><td>MAXDUR</td><td>Max. Duration</td><td>For a stretchable or reprofilable activity, the maximum duration</td><td>DURA</td><td>VARCHAR2(10)</td><td>NULL</td></tr>
</table>

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		that resource scheduling can use for this activity.			
MAXSPLITS	Max. Number of Splits	For a splittable activity, the maximum number of times that resource scheduling can split the activity.	INTE	NUMBER(10, 0)	NULL
MINSPLITD	Min. Split Length	For a splittable activity, the minimum period of time that resource scheduling can use when splitting the activity.	DURA	VARCHAR2(10)	NULL
MSPUNIQUEID	MSP Unique ID	The MSP unique ID field for activities imported from Microsoft Project.	INTE	NUMBER(10, 0)	NULL
OPKEY	Key Activity	Controls whether risk analysis is to provide a detailed analysis of the activity.	BOOL	VARCHAR2(4)	NULL
ORIG_DUR	Original Duration	The original duration for an activity. This value is user-entered, except for hammocks and subprojects, where the value is calculated by time analysis, and effort driven activities, where the value is calculated based on the amount of resource assigned to the activity.	DURA	VARCHAR2(10)	NULL
PPC	Physical % Complete	The physical percent complete for an activity. Value may be user-entered or calculated by progress calculations.	DBLE	NUMBER(15, 2)	NULL
PRIORITY	Priority	The relative priority used by resource scheduling of this activity compared to others.	INTE	NUMBER(10, 0)	NULL
PROGTYPE	Progress Type	The progress type used by an activity. Controls the behavior of the activity in conjunction with the actual date information. Valid values are: Planned, Remaining Duration, Percent Complete, Elapsed Duration, and Complete. These values are stored in the Open Plan database as null, R, P, E, and C, respectively.	PROG	VARCHAR2(4)	NULL
PROGVALUE	Progress Value	The progress value for an activity. The interpretation of Progress Value depends on the value of Progress Type. For example, a progress value of 100 could represent 100% complete or 100d remaining duration.	DURA	VARCHAR2(10)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PWAGUID	Project Server Task Guid	The Globally Unique Identifier (GUID) for activities imported from Microsoft Project Server.	TEXT	VARCHAR2(36)	NULL
RS_SUPRESS	Suppress Requirements	If set to true, this activity and its resource requirements are ignored during resource scheduling.	BOOL	VARCHAR2(4)	NULL
RSCLASS	R/S Type	The resource scheduling options for an activity when the activity cannot be scheduled without conflict. Valid values are: Normal, Splittable, Stretchable, Reprofilable, and Immediate. These values are stored in the Open Plan database as null, P, T, R, and I, respectively.	RSCL	VARCHAR2(4)	NULL
SEP_ASG	Separate Assignments	Controls whether Open Plan should consider multiple assignments of members of the same pool or skill as separate assignments.	BOOL	VARCHAR2(4)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SOURCE_BASELINE	None	When a baseline is selected, the Baseline Start and Baseline Fields on the activity table are populated with the corresponding values from those fields for the selected baseline. If you save the project with the baseline selected, those values will also be saved. Subsequently, if the activity table is displayed (e.g., in a spreadsheet) without a baseline selected, you will see the Baseline Start and Finish dates that were previously stored. The SOURCE_BASELINE field shows you the name of the baseline that was the source for those values.	TEXT	VARCHAR2(1)	NULL
STARTPC	User Defined EVT Split %	When using the "User-defined percentage" earned value technique, this field contains the percentage earned at the start of the activity. The remainder is earned upon completion of the activity.	INTE	NUMBER(10, 0)	NULL
TARGFTYPE	Target Finish Type	The target finish type for an activity. Valid values are: None, Not Earlier Than, Not Later Than, On Target, and Fixed Target. These values are stored in the	TARG	VARCHAR2(4)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		Open Plan database as null, NE, NL, ON, and FX, respectively.			
TARGSTYPE	Target Start Type	The target start type for an activity. Valid values are: None, Not Earlier Than, Not Later Than, On Target, and Fixed Target. These values are stored in the Open Plan database as null, NE, NL, ON, and FX, respectively.	TARG	VARCHAR2(4)	NULL
TFDATE	Target Finish	The target finish date for an activity.	FDAT	DATE	NULL
TSDATE	Target Start	The target start date for an activity.	DATE	DATE	NULL
USR_ID	Updated By	The user ID of the last user to update the activity record.	TEXT	VARCHAR2(10)	NOT NULL
XFDATE	Expected Finish	The expected finish date for an activity.	FDAT	DATE	NULL

Table OPP_ASG - Resource Assignments

The resource assignments table stores activity resource assignments that are used for resource scheduling and cost calculations. The data dictionary table identifier for table OPP_ASG is ASG.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID	The unique identifier for an activity record in the database.	GUID	VARCHAR2(22)	NOT NULL
ALT_RES_UID	Alternate Res. ID	The unique identifier that links the assignment record to the resource record representing the resource assignment alternate.	GUID	VARCHAR2(22)	NOT NULL
ASG_UID	None	The unique identifier for a resource assignment.	GUID	VARCHAR2(22)	NOT NULL
CST_CLASS	Cost Class	The cost class for the assignment. This information is used during Cobra integration.	TEXT	VARCHAR2(20)	NULL
DIR_UID	Project File	The unique identifier that links a resource assignment record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
INTEGRATION_ID	None	The unique identifier used to match an Open Plan resource assignment to its equivalent in the third-party integration product (for example, Primavera).	INTE	NUMBER(10,0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
LASTUPDATE	Last Updated	The date and time that the resource assignment was last updated.	DATE	DATE	NOT NULL
LEV_TYPE	Res. Curve	The spread curve selected for the resource assignment. A null value means that the resource is a level-per-time unit instead of a total. Valid values for this field come from the spread curves defined via the Open Plan Tools Menu, Spread Curves.	TEXT	VARCHAR2(1)	NULL
PPC	Physical % Complete	The physical percent complete for a resource assignment.	DBLE	NUMBER(15, 2)	NULL
RDS_UID		The unique identifier that links a resource assignment record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
REMAINING	Remaining Requirement	The remaining resource requirement for a resource assignment.	DBLE	NUMBER(15, 2)	NULL
RES_LEVEL	Res. Level	The amount of resource required by the resource assignment.	DBLE	NUMBER(15, 2)	NULL
RES_OFFSET	Res. Offset	The number of time units that must elapse between the start of an activity and the time when the resource starts working.	DURA	VARCHAR2(10)	NULL
RES_PERIOD	Res. Period	The total time for which the resource is required.	DURA	VARCHAR2(10)	NULL
RES_UID		The unique identifier that links the assignment record to a resource record.	GUID	VARCHAR2(22)	NOT NULL
RES_SKL_UID		The unique identifier that links the assignment record to a resource skill record.	GUID	VARCHAR(22)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUPPRESS	Suppress Flag	Flag to suppress a resource's requirement during resource scheduling.	BOOL	VARCHAR2(4)	NULL
USR_ID	Updated By	The user ID of the last user to update the resource assignment record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_CST - Resource Actuals

The Resource Actuals table stores actual quantities and actual costs that are incurred by resources against activities. The RDS_UID and RES_UID fields are allowed to be blank to support activity level costs. The data dictionary table identifier for table OPP_CST is CST.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID	The unique identifier for an activity.	GUID	VARCHAR2(22)	NOT NULL
ACWP_CST	ACWP Cost	The actual cost of work performed for an individual cost record.	DBLE	NUMBER(15, 2)	NULL
ACWP_ESC	ACWP Escalated Cost	The escalated actual cost of work performed for an individual cost record.	DBLE	NUMBER(15, 2)	NULL
ACWP_QTY	ACWP Quantity	The quantity of units for the work performed by labor resources for a cost record.	DBLE	NUMBER(15, 2)	NULL
BCWP_CST	BCWP Cost	The budgeted cost of work performed for an individual cost record.	DBLE	NUMBER(15, 2)	NULL
BCWP_ESC	BCWP Escalated Cost	The escalated budgeted cost of work performed for an individual cost record.	DBLE	NUMBER(15, 2)	NULL
BCWP_QTY	BCWP Quantity	The earned value of labor resource units budgeted for the activity.	DBLE	NUMBER(15, 2)	NULL
CST_UID	None	The unique identifier for a cost record.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	Project File	The unique identifier that links a resource cost record to a unique project object directory record.	GUID	VARCHAR2(22)	NOT NULL
END_DATE	Period End Date	The end date for the resource cost period.	FDAT	DATE	NULL
RDS_UID	Resource File	The unique identifier that links a cost record to a unique resource directory object record.	GUID	VARCHAR2(22)	NULL
RES_UID		The unique identifier that links a cost record to a unique resource record.	GUID	VARCHAR2(22)	NULL
RES_SKL_UID		The unique identifier that links the assignment record to a resource skill record.	GUID	VARCHAR(22)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
START_DATE	Period Start Date	The start date for a resource cost period.	DATE	DATE	NULL

Table OPP_IRL - Inter-project Relationships

The Inter-project Relationships table contains the definition and scheduling constraint information for relationships between activities in different projects. The data dictionary table identifier for table OPP_IRL is IRL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLH_UID	Calendar ID	The unique identifier that links the relationship record to a unique calendar detail record.	GUID	VARCHAR2(22)	NOT NULL
IRL_UID		The unique identifier for an inter-project relationship record. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	Last Updated	The date and time that the inter-project relationship record was last updated.	DATE	DATE	NOT NULL
PRED_ACT_UID	Predecessor	The unique identifier of the predecessor activity.	GUID	VARCHAR2(22)	NOT NULL
PRED_DIR_UID	None	The unique identifier of the predecessor activity's project.	GUID	VARCHAR2(22)	NOT NULL
REL_FF	Relationship Free Float	The relationship's free float.	DURA	VARCHAR2(10)	NULL
REL_LAG	Relationship Lag	The relationship's lag.	DURA	VARCHAR2(10)	NOT NULL
REL_PROBABILITY	Relationship Branch Probability	Probability of a relationship being active during a risk analysis simulation pass.	DBLE	NUMBER(15,2)	NULL
REL_TF	Relationship Total Float	The relationship's total float.	DURA	VARCHAR2(10)	NULL
REL_TYPE	Relationship Type	The relationship type. Valid values are: Finish to Start, Start to Start, Start to Finish, and Finish to Finish. These values are stored in the Open Plan database as FS, SS, SF, and FF, respectively.	RELT	VARCHAR2(4)	NOT NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUCC_ACT_UID	Successor	The unique identifier of the successor activity.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SUCC_DIR_UID	None	The unique identifier of the successor activity's project.	GUID	VARCHAR2(22)	NOT NULL
USR_ID	Updated By	The user ID of the last user to update the inter-project relationship record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_REL - Activity Relationships

The Activity Relationships table contains the definition and scheduling constraint information for relationships between activities in the same project. The data dictionary table identifier for table OPP_REL is REL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLH_UID	Calendar ID	The unique identifier that links the relationship record to a unique calendar detail record.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	Last Updated	The date and time that the relationship record was last updated.	DATE	DATE	NOT NULL
PRED_ACT_UID	Predecessor	The unique identifier of the predecessor activity.	GUID	VARCHAR2(22)	NOT NULL
REL_FF	Relationship Free Float	The relationship's free float.	DURA	VARCHAR2(10)	NULL
REL_LAG	Relationship Lag	The relationship's lag.	DURA	VARCHAR2(10)	NOT NULL
REL_PROBABILITY	Relationship Branch Probability	Probability of a relationship being active during a risk analysis simulation pass.	DBLE	NUMBER(15,2)	NULL
REL_TF	Relationship Total Float	The relationship's total float.	DURA	VARCHAR2(10)	NULL
REL_TYPE	Relationship Type	The relationship type. Valid values are: Finish to Start, Start to Start, Start to Finish, and Finish to Finish. These values are stored in the Open Plan database as FS, SS, SF, and FF, respectively.	RELT	VARCHAR2(4)	NOT NULL
REL_UID		The unique identifier for an activity relationship record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
SUCC_ACT_UID	Successor	The unique identifier of the successor activity.	GUID	VARCHAR2(22)	NOT NULL
USR_ID	Updated By	The user ID of the user who last updated the relationship record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_RSK - Risk Key Activities

The risk key activities table contains information generated by the risk analysis module for activities that are marked as key activities. The data dictionary table identifier for table OPP_RSK is RSK.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID	The unique identifier for an activity record in the database.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
EFDATE1	First Early Finish	The first date in a set of early finish observations in a group.	FDAT	DATE	NULL
EFDATE2	Last Early Finish	The last date in a set of early finish observations in a group.	FDAT	DATE	NULL
EFDATE_COUNT	Early Finish Count	The number of early finish observations in the group.	INTE	NUMBER(10,0)	NULL
ESDATE1	First Early Start	The first date in a set of early start observations in a group.	DATE	DATE	NULL
ESDATE2	Last Early Start	The last date in a set of early start observations in a group.	DATE	DATE	NULL
ESDATE_COUNT	Early Start Count	The number of early start observations in the group.	INTE	NUMBER(10,0)	NULL
LFDATE1	First Late Finish	The first date in a set of late finish observations in a group.	FDAT	DATE	NULL
LFDATE2	Last Late Finish	The last date in a set of late finish observations in a group.	FDAT	DATE	NULL
LFDATE_COUNT	Late Finish Count	The number of late finish observations in the group.	INTE	NUMBER(10,0)	NULL
LSDATE1	First Late Start	The first date in a set of late start observations in a group.	DATE	DATE	NULL
LSDATE2	Last Late Start	The last date in a set of late start observations in a group.	DATE	DATE	NULL
LSDATE_COUNT	Late Start Count	The number of late start observations in the group.	INTE	NUMBER(10,0)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RSK_UID	None	The unique identifier for a risk activity record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Table OPP_STP – Activity Steps

The Activity Steps table stores progress-related information for individual steps assigned to activities. The data dictionary table identifier for table OPP_STP is STP.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID	The unique identifier for an activity record in the database	GUID	VARCHAR2(22)	NOT NULL
COMPLETED	Completed	Indicates whether step has been completed.	TEXT	VARCHAR2(1)	NULL
DESCRIPTION	Description	Description of step	MEMO	LONG	NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
INTEGRATION_ID	Integration ID	Object (unique) ID of the P6 step	INTE	NUMBER(10,0)	DEFAULT (0) NOT NULL
LASTUPDATE	None	The date and time that the step record was last updated.	DATE	DATE	NOT NULL
PERCENT_COMPLETE	% Complete	The % complete value for the step	DBLE	NUMBER(10,2)	NULL
STP_ID	Name	The step name (must be unique under a given activity)	TEXT	NVARCHAR2(59)	NOT NULL
STP_NUM	None	The step number (indicates the order in which the step appears). The number must be unique under a given activity.	INTE	NUMBER(10,0)	NOT NULL
STP_UID	None	The OP unique identifier of an activity step record	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrecy in the shared mode	INTE	NUMBER(10,0)	NOT NULL
WEIGHT	Weight	The weight given to a particular step (the weight % for a step may be calculated	DBLE	NUMBER(10,2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		by dividing the step's weight by the sum of all step weights for the activity).			

Table OPP_SUB - Subproject Summary Information

The Subproject Summary Information table contains calculated summary data for subproject activities as well as the external subproject name for external subproject activities. The data dictionary table identifier for table OPP_SUB is SUB.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID	The unique identifier for an activity record in the database.	GUID	VARCHAR2(22)	NOT NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
S_EFDATE	Early Finish	The subproject's early finish date.	FDAT	DATE	NULL
S_ESDATE	Early Start	The subproject's early start date.	DATE	DATE	NULL
S_LFDATE	Late Finish	The subproject's late finish date.	FDAT	DATE	NULL
S_LSDATE	Late Start	The subproject's late start date.	DATE	DATE	NULL
S_SFDATE	Sched. Finish	The subproject's scheduled finish date.	FDAT	DATE	NULL
S_SSDATE	Sched. Start	The subproject's scheduled start date.	DATE	DATE	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUBPCOMP	Sub. % Complete	The subproject's percent complete.	DBLE	NUMBER(15, 2)	NULL
SUBPRJ_UID	External Subproject	The unique identifier that identifies an external subproject within a project.	GUID	VARCHAR2(22)	NOT NULL

Table OPP_USE - Resource Usage

The Resource Usage table contains resource usage information that is generated by the resource scheduling module. The data dictionary table identifier for table OPP_USE is USE.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_UID	Activity ID.	The unique identifier for an activity.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ALT_RES_UID	Alternate Res. ID	The unique identifier that links the resource usage record to the resource record representing the resource assignment alternate.	GUID	VARCHAR2(22)	NULL
ASG_UID	None	The unique identifier for a resource assignment.	GUID	VARCHAR2(22)	NOT NULL
CLH_UID	Calendar ID	The unique identifier of the resource calendar for a resource usage record.	GUID	VARCHAR2(22)	NOT NULL
CST_CLASS	Cost Class	The cost class for the resource usage. This information is used during Cobra integration.	TEXT	VARCHAR2(20)	NULL
DIR_UID	Project File	The unique identifier for a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
PALLOC_UID	Project Allocation	The unique identifier of the project directory object for which this resource was reserved.	GUID	VARCHAR2(22)	NULL
RDS_UID	Resource File	The unique identifier that links a resource usage record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
RES_CST	Resource Cost	The cost of the resource used during the period defined by the record's start and end dates.	DBLE	NUMBER(15, 2)	NULL
RES_ESC	Resource Escalated Cost	The escalated cost of the resource used during the period defined by the record's start and end dates.	DBLE	NUMBER(15, 2)	NULL
RES_UID		The unique identifier for a resource.	GUID	VARCHAR2(22)	NOT NULL
RES_SKL_UID		The unique identifier that links the assignment record to a resource skill record.	GUID	VARCHAR(22)	NULL
RES_USED	Res. Used	The amount of resource used during the period defined by the record's start and end dates.	DBLE	NUMBER(15, 2)	NULL
RFDATE	Finish Date	The end of the period during which the resource was used.	FDAT	DATE	NULL
RSDATE	Start Date	The date the resource usage begins.	DATE	DATE	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
USE_UID	None	The unique identifier of a resource usage record.	GUID	VARCHAR2(22)	NOT NULL

Resource Data

Table OPP_AVL - Resource Availability

The Resource Availability table contains information describing the amount of resource quantity that is available for a specific time period. The data dictionary table identifier for table OPP_AVL is AVL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
AVL_UID	None	The unique identifier for a resource availability record.	GUID	VARCHAR2(22)	NOT NULL
CLH_UID	Calendar ID	The unique identifier of the resource calendar for a resource usage record.	GUID	VARCHAR2(22)	NULL
DIR_UID	Project File	The unique identifier that links a resource usage record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	Last Updated	The date and time that the resource availability record was last updated.	DATE	DATE	NOT NULL
PALLOC_UID	Project Allocation	The unique identifier of the project directory object for which this resource was reserved.	GUID	VARCHAR2(22)	NULL
RES_LEVEL	Res. Level	Quantity available of this resource during the time period defined by the start and finish dates.	DBLE	NUMBER(15, 2)	NOT NULL
RES_UID	Resource ID	The unique identifier for a resource.	GUID	VARCHAR2(22)	NOT NULL
RFDATE	Finish Date	A valid date upon which the resource availability ends.	FDAT	DATE	NULL
RSDATE	Start Date	A valid date upon which the resource become available.	DATE	DATE	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
USR_ID	Updated By	The user ID of the last user to update the resource availability record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_PSU - Project Summary Usage

The Resource Project Summary Usage table stores information that is used to produce a complete summary usage of project resources. The data dictionary table identifier for table is OPP_PSU is PSU.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ACT_ID	Activity ID	An activity ID for a project summary usage record.	TEXT	VARCHAR2(59)	NULL
CLD_UID		The unique identifier that links a project summary usage record to a unique calendar object directory record for the calendar file assigned to the project.	GUID	VARCHAR2(22)	NULL
CLH_UID	Calendar ID	The unique identifier that links the project summary usage record to a calendar detail record.	GUID	VARCHAR2(22)	NULL
DIR_UID	Project File	This is the unique identifier for the Resource File (not a project object).	GUID	VARCHAR2(22)	NOT NULL
PALLOC_UID	Project Allocation	The unique identifier of the project directory object for which this resource was reserved.	GUID	VARCHAR2(22)	NULL
PRJ_UID	None	The unique identifier of a project object directory record.	GUID	VARCHAR2(22)	NOT NULL
SU_UID	None	The unique identifier for a project summary usage record.	GUID	VARCHAR2(22)	NULL
RES_CST	Resource Cost	The cost of the reserved resource.	DBLE	NUMBER(15, 2)	NULL
RES_ESC	Resource Escalated Cost	The escalated cost of the reserved resourced.	DBLE	NUMBER(15, 2)	NULL
RES_LEVEL	Res. Level	The amount of reserved resource that was assigned.	DBLE	NUMBER(15, 2)	NULL
RES_UID		The unique identifier of the reserved resource.	GUID	VARCHAR2(22)	NOT NULL
RES_SKL_UID		The unique identifier that links the assignment record to a resource skill record.	GUID	VARCHAR(22)	NULL
RES_USED	Res. Used	The total amount of resource that is reserved.	DBLE	NUMBER(15, 2)	NULL
RFDATE	Finish Date	The date the resource reservation ends.	FDATE	DATE	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RS_PRIORTY	Priority	Used to determine the order of processing during resource scheduling.	INTE	NUMBER(10, 0)	NOT NULL
RSDATE	Start Date	The date the resource reservation begins.	DATE	DATE	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL

Table OPP_RES - Resource Definitions

The Resource Definition table contains the resource elements that make up a resource breakdown structure. The data dictionary table identifier for table is OPP_RES is RES.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CLC_COST	Cost based on Progress Qty	If True, the cost of the resource is based on the progressed quantity of the resource.	BOOL	VARCHAR2(4)	NULL
CLC_PROG	Progress based on Act Progress	If True, the resource's progress is calculated from the progress of the activity to which it is assigned.	BOOL	VARCHAR2(4)	NULL
DESCRIPTION	Res. Desc.	The description of a resource record.	TEXT	VARCHAR2(125)	NULL
DIR_UID	None	The unique identifier that links a resource record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
EFF_FACTOR	Effort Factor	The effort factor for a resource.	DBLE	NUMBER(15, 2)	NULL
EMAIL	Email Address	The email address of a resource.	TEXT	VARCHAR2(60)	NULL
EMP_ID	Employee ID	The employee ID of a resource.	TEXT	VARCHAR2(20)	NULL
LASTUPDATE	Last Updated	The date and time that the resource file was last updated.	DATE	DATE	NOT NULL
MSPUNIQUEID	MSP Unique ID	The MSP unique ID field for activities imported from Microsoft Project.	INTE	NUMBER(10, 0)	NULL
NO_LIST	Suppress in Lists	If selected, this option suppresses the inclusion of this resource in all lists. While the resource is not deleted from the file (and, thus, does not affect cost calculations), the resource cannot be used in resource assignments, nor is it included in any list of resources.	BOOL	VARCHAR2(4)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PALLOC_UID	Project Allocation	The unique identifier of the project directory object for which this resource was reserved.	GUID	VARCHAR2(22)	NULL
POSITION_NUM	Child Position		INTE	NUMBER(10, 0)	NULL
PWAGUID	Project Server Resource Guid	The Globally Unique Identifier (GUID) for resources imported from Microsoft Project Server.	TEXT	VARCHAR2(36)	NULL
RES_CLASS	Res. Category	The resource category. Valid values are: Labor, Material, Other Direct Costs, and Subcontract. These values are stored in the Open Plan database as L, N, C, and S, respectively.	RESC	VARCHAR2(20)	NULL
RES_ID	Resource ID	The identifier of a resource, unique within a resource file.	HIER	VARCHAR2(59)	NOT NULL
RES_TYPE	Res. Type	The resource type. Valid values are: Normal, Consumable, Perishable, Resource Pool, and Skill. These values are stored in the Open Plan database as null, C, D, P and S, respectively.	REST	VARCHAR2(20)	NULL
RES_UID		The unique identifier for a resource record.	GUID	VARCHAR2(22)	NOT NULL
ROLLCOST	Res. Type	If set to False, Open Plan does not consider this resource during cost calculations and it is not rolled up to the activity level.	BOOL	VARCHAR2(4)	NULL
ROLLUP	Roll-up for Scheduling	If selected, this option specifies that all availabilities and assignments for this resource should be treated as availabilities and assignments for the first parent of the resource in the hierarchy that does not have this option selected.	BOOL	VARCHAR2(4)	NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SUPPRESS	Suppress	If selected, this option suppresses resource scheduling for this resource. During resource scheduling, Open Plan treats suppressed resources as though they were unconstrained.	BOOL	VARCHAR2(4)	NULL
THRESHOLD	Res. Threshold	Specifies an additional number of units of availability for a resource	DBLE	NUMBER(15, 2)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		that can be used to avoid project delays if necessary.			
UNIT	Res. Units	The unit in which a resource's cost is measured.	TEXT	VARCHAR2(20)	NULL
UNIT_COST	Res. Unit Cost	The unit cost of a resource.	DBLE	NUMBER(15, 2)	NULL
USR_ID	Updated By	The last user to update the record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_RSL - Resource Cost Escalation

The Resource Cost Escalation table stores information that allows resource cost to change over time. The data dictionary table identifier for table OPP_RSL is RSL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
COST	Res. Unit Cost	The resource escalation rate of a resource.	DBLE	NUMBER(15, 2)	NOT NULL
DIR_UID		The unique identifier for a resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
LASTUPDATE	Last Updated	The date and time that the resource escalation record was last updated.	DATE	DATE	NOT NULL
RES_UID		The unique identifier that links a resource escalation record to a resource record.	GUID	VARCHAR2(22)	NOT NULL
RSL_DATE	Res. Effective Date	The date upon which the resource escalation rate takes effect.	DATE	DATE	NOT NULL
RSL_UID		The unique identifier for a resource escalation record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	Update Count	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
USR_ID	Updated By	The user ID of the last user to update the resource escalation record.	TEXT	VARCHAR2(20)	NOT NULL

Table OPP_SKL - Skill Assignments

The Resource Skill Assignment table stores skill assignments to be used against resource definitions. The data dictionary table identifier for table OPP_SKL is SKL.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DIR_UID	None	The unique identifier that links a skill record to a unique resource object directory record.	GUID	VARCHAR2(22)	NOT NULL
PROFICIENCY	Proficiency	User-specified proficiency rating for a skill assigned to a resource.	DBLE	NUMBER(15,2)	NULL
RES_SKL_UID	None	The unique identifier for a skill assigned to a resource.	GUID	VARCHAR2(22)	NOT NULL
RES_UID	None	The unique identifier for the resource record that represents a skill record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
SKL_UID	None	The unique identifier for a skill record.	GUID	VARCHAR2(22)	NOT NULL

User Defined Fields

Table WST_NTX - Note Text

The Note Text table stores the note text associated with note categories assigned to a data object. The data dictionary table identifier for table WST_NTX is NTX.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier that links a baseline record to a note record.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier for the directory object to which the note text belongs.	GUID	VARCHAR2(22)	NOT NULL
FIELD_NAME	None	The category of a note text field, which is a field name in OPP_UDF.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_VALUE	None	The text of a note.	MEMO	LONG	NULL
FK_UID	None	The unique ID that links a note text record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID	None	A unique identifier for the note text record.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The data type for which the note text is stored. Valid values are: ACT (Activity), CDR (Code Directory), COD (Code Data), PRJ (Project Directory), RDS (Resource	TEXT	VARCHAR2(30)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
		Directory), and RES (Resource Data).			

Table WST_UDF - User Field Definitions

The User Fields table stores the definitions of user-defined fields for each table with which they are associated. The data dictionary table identifier for table WST_UDF is UDF.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
DESCRIPTION	None	The description of a user-defined field.	TEXT	VARCHAR2(60)	NULL
FIELD_NAME	None	The name of a user defined field.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_TYPE	None	The name of a user defined field, note category, or code assignment field.	TEXT	VARCHAR2(4)	NOT NULL
LOOKUP_FIELD	None		TEXT	VARCHAR(30)	NULL
LOOKUP_TABLE_NAME	None		TEXT	VARCHAR(30)	NULL
LOOKUP_TABLE_TYPE	None		TEXT	VARCHAR(30)	NULL
SEQUENCE	None	Used to manage multi-user concurrency in shared mode.	INTE	NUMBER(10, 0)	NOT NULL
TABLE_TYPE	None	The data type for which the user-defined field is specified. Valid values are:	TEXT	VARCHAR2(30)	NOT NULL
UDF_UID	None	The unique identifier for a user-defined field record.	GUID	VARCHAR2(22)	NOT NULL

Table WST_UDT - User Date And Finish Date Data

The User Dates table stores user-defined date field data. This table is a sparse storage mechanism that contains only the data for fields that have data defined. The data dictionary table identifier for table WST_UDT is UDT.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier for the directory object to which the user-defined field belongs.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
FIELD_NAME	None	The name of a user defined date or finish date field.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_VALUE	None	The value of a user defined date or finish date field.	DATE	DATE	NOT NULL
FK_UID	None	The unique ID that links a user defined date or finish date record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID	None	The unique identifier for a user-defined date or finish date value.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The data type for which the user-defined date field data is stored. Valid values are: ACT (Activity), BSA (Baseline Activity), CDR (Code Directory), COD (Code Data), PRJ (Project Directory), ASG (Resource Assignment), REL (Relationship), AVL (Resource Availability), RES (Resource Data), and RDS (Resource Directory.	TEXT	VARCHAR2(30)	NOT NULL

Table WST_UDU - User Duration Data

The User Durations table stores user defined duration data. This table is a sparse storage mechanism that contains only the data for fields that have data defined. The data dictionary table identifier for table WST_UDU is UDU.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier for the directory object to which the user-defined field belongs.	GUID	VARCHAR2(22)	NOT NULL
FIELD_NAME	None	The name of a user defined duration field.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_VALUE	None	The value of a user defined duration field.	DURA	VARCHAR2(10)	NOT NULL
FK_UID	None	The unique ID that links a user defined duration record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID	None	The unique identifier for a user-defined duration value.	GUID	VARCHAR2(22)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
TABLE_TYPE	None	The data type for which the user-defined duration field data is stored. Valid values are: ACT (Activity), BSA (Baseline Activity), CDR (Code Directory), COD (Code Data), PRJ (Project Directory), ASG (Resource Assignment), REL (Relationship), AVL (Resource Availability), RES (Resource Data), and RDS (Resource Directory.	TEXT	VARCHAR2(30)	NOT NULL

Table WST_UNM - User Numeric Data

The User Numbers table stores user defined number field data. This table is a sparse storage mechanism that contains only the data for fields that have data defined. The data dictionary table identifier for table WST_UNM is UNM.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier for the directory object to which the user-defined field belongs.	GUID	VARCHAR2(22)	NOT NULL
FIELD_NAME	None	The name of a user defined numeric field.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_VALUE	None	The value of a user defined numeric field.	DBLE	NUMBER(15, 2)	NOT NULL
FK_UID	None	The unique ID that links a user defined numeric record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID	None	The unique identifier for a user-defined numeric value.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The data type for which the user-defined numeric field data is stored. Valid values are: ACT (Activity), BSA (Baseline Activity), CDR (Code Directory), COD (Code Data), PRJ (Project Directory), ASG (Resource Assignment), REL (Relationship), AVL (Resource Availability), RES (Resource Data), and RDS (Resource Directory.	TEXT	VARCHAR2(30)	NOT NULL

Table WST_UTX - User Text Data

The User Text table stores user defined text field data. This table is a sparse storage mechanism that contains only the data for fields that have data defined. The data dictionary table identifier for table WST_UTX is UTX.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
BAS_UID	None	The unique identifier for a baseline record.	GUID	VARCHAR2(22)	NULL
DIR_UID	None	The unique identifier for the directory object to which the user-defined field belongs.	GUID	VARCHAR2(22)	NOT NULL
FIELD_NAME	None	The name of a user defined text field.	TEXT	VARCHAR2(60)	NOT NULL
FIELD_VALUE	None	The value of a user defined text field.	TEXT	VARCHAR2(255)	NOT NULL
FK_UID	None	The unique ID that links a user defined text record to a record in another table. This value is not displayed in Open Plan.	GUID	VARCHAR2(22)	NOT NULL
ROW_UID	None	The unique identifier for a user-defined text value.	GUID	VARCHAR2(22)	NOT NULL
TABLE_TYPE	None	The data type for which the user-defined text field data is stored. Valid values are: ACT (Activity), BSA (Baseline Activity), CDR (Code Directory), COD (Code Data), PRJ (Project Directory), ASG (Resource Assignment), REL (Relationship), AVL (Resource Availability), RES (Resource Data), and RDS (Resource Directory.	TEXT	VARCHAR(30)	NOT NULL

User Validations

Table OPP_UVD - User Validation Definitions

The User Validations Definitions table contains the definitions of user validations. The data dictionary identifier for the OPP_UVD table is UVD.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
UVD_UID	None	The unique identifier of a user validation record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency.	INTE	NUMBER(10,0)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
UVD_ID	None		TEXT	VARCHAR2(60)	NOT NULL
USR_ID	None		TEXT	VARCHAR2(20)	NOT NULL
LASTUPDATE	None	The date and time that a user validation was last updated.	DATE	DATE	NOT NULL
TABLE_TYPE	None	The table type to which the user validation applies. Valid Values are: ACT (Activity), REL (Relationship) and ASG (Resource Assignment)	TEXT	VARCHAR2(30)	NOT NULL
ERROR_TYPE	None	Specifies if the user validation results in an error or warning. Valid values are: 0 (Warning) and 1 (Error)	INTE	NUMBER(10,0)	NOT NULL
ERROR_MSG	None	The warning or error message that will be displayed when a user validation fails.	TEXT	VARCHAR2(255)	NULL
EXPRESSION	None	A calculated field or filter expression that defines the validation to be performed. A calculated field result of TRUE indicates success and a value of FALSE indicates failure.	TEXT	VARCHAR2(2000)	NULL

Table OPP_UVA - User Validation Associations

The User Validation Associations table contains the associations of User Validations to Projects. The data dictionary identifier for the OPP_UVA table is UVA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
UVA_UID	None	The unique identifier of a user validation association record.	GUID	VARCHAR2(22)	NOT NULL
SEQUENCE	None	Used to manage multi-user concurrency.	INTE	NUMBER(10,0)	NOT NULL
DIR_UID	None	The unique identifier for the Project that a user validation has been associated with.	GUID	VARCHAR2(60)	NOT NULL
UVD_UID	None	The unique identifier of the User Validation that has been associated with a project.	TEXT	VARCHAR2(20)	NOT NULL

Common FW Tables not used by Open Plan

Table FW_CFGProcessQueue

The Process Queue table contains the definitions of process queue defined in EPM SA.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID	None	Queue ID		VARCHAR(32)	NOT NULL
QueueName	None	Queue Name		NVARCHAR(40)	NULL
QueueStatus	None	Queue Status		SMALLINT	NOT NULL
Priority	None	Priority		SMALLINT	NOT NULL
ProcessServerMachineName	None	Process Server Machine Name.		NVARCHAR(255)	NULL
MaxRunningThreads	None	Max Running Threads.		INT	NOT NULL
CreateUser	None	Created By		NVARCHAR(20)	NULL
CreateDate	None	Created Date		DATETIME	NULL
ModUser	None	Last Updated By		NVARCHAR(20)	NULL
ModDate	None	Last Updated		DATETIME	

Table FW_CFGProcessQueueSetup

The Process Queue Setup table stores the configuration used by the process queues.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
PKey	None	PKey		VARCHAR(4)	NOT NULL
PollingFrequency	None	Polling Frequency		INT	NOT NULL
Retailistory	None	Retail History		VARCHAR(1)	NOT NULL
DaysToRetainHistory	None	Days to Retain History		SMALLINT	NOT NULL
RetainErrors	None	Retain Errors		VARCHAR(1)	NULL
CreateUser	None	Created By		NVARCHAR(20)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
CreateDate	None	Created Date		DATETIME	NULL
ModUser	None	Last Updated By		NVARCHAR(20)	NULL
ModDate	None	Last Updated		DATETIME	NULL

Table FW_DetailJobs

The Detail Job table contains detailed information about the processes recorded in the Job table. It includes details on the recurrence of these processes.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID(PK)	None	Queue ID		VARCHAR(32)	NOT NULL
ProcessID(PK)	None	Process ID		VARCHAR(32)	NOT NULL
DetailID(PK)	None	Detail ID		VARCHAR(32)	NOT NULL
Seq	None	Seq		INT	NOT NULL
ProgID	None	Prog ID		VARCHAR(200)	NULL
Description	None	Description		NVARCHAR(250)	NULL
TimeStarted	None	Time Started		DATETIME	NULL
TimeEnded	None	Time Ended		DATETIME	NULL
ProcessStatus		Process Status		SMALLINT	NOT NULL
AlertPending		Alert Pending		VARCHAR(1)	NULL
TerminationMessage		Termination Message		NVARCHAR(4000)	NULL
xmlParams		XML Params		NVARCHAR(4000)	NULL
ServerName		Server Name		NVARCHAR(255)	NULL
Company		Company		NVARCHAR(14)	NULL
DATETIMEFormat		Date Time Format		NVARCHAR(4000)	NULL
NumberFormat		Number Format		NVARCHAR(4000)	NULL
UICultureName		UI Culture Name		VARCHAR(10)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ContextState		Context State		NVARCHAR(4000)	NULL

Table FW_DetailJobStatus

The Job Status table stores specific status messages for each process recorded in the Detail Job table.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID(PK)		Queue ID		VARCHAR(32)	NOT NULL
ProcessID(PK)		Process ID		VARCHAR(32)	NOT NULL
DetailID(PK)		Detail ID		VARCHAR(32)	NOT NULL
StatusID(PK)		Status ID		VARCHAR(36)	NOT NULL
StatusDate		Status Date		DATETIME	NULL
StatusMessage		Status Message		NVARCHAR(1000)	NULL

Table FW_DetailProfiles

The Job Profile Detail table contains detailed information of the processes under each group defined in the Job Profile Header table.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID(PK)	None	Queue ID		VARCHAR(32)	NOT NULL
ProcessID(PK)	None	Process ID		VARCHAR(32)	NOT NULL
DetailID(PK)	None	Detail ID		VARCHAR(32)	NOT NULL
Seq	None	Seq		INT	NOT NULL
ProgID	None	Prog ID		VARCHAR(200)	NULL
Description	None	Description		NVARCHAR(80)	NULL
TimeStarted	None	Time Started		DATETIME	NULL
TimeEnded	None	Time Ended		DATETIME	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
ProcessStatus	None	Process Status		SMALLINT	NOT NULL
AlertPending	None	Alert Pending		VARCHAR(1)	NULL
TerminationMessage	None	Termination Message		NVARCHAR(4000)	NULL
xmlParams	None	XML Params		NVARCHAR(4000)	NULL
Company	None	Company		NVARCHAR(14)	NULL
DATETIMEFormat	None	Date Time Format		NVARCHAR(4000)	NULL
NumberFormat	None	Number Format		NVARCHAR(4000)	NULL
UICultureName	None	UI Culture Name		VARCHAR(10)	NULL
ContextState	None	Context State		NVARCHAR(4000)	NULL
CreateUser	None	Created By		NVARCHAR(20)	NULL
CreateDate	None	Created Date		DATETIME	NULL
ModUser	None	Last Updated By		NVARCHAR(20)	NULL
ModDate	None	Last Updated		DATETIME	NULL

Table FW_MasterJobs

The Job Master table is the header table of all processes submitted to the process queue. It stores the start and end time of each process, as well as the process server that executes the process.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID(PK)	None	Queue ID		VARCHAR(32)	NOT NULL
ProcessID(PK)	None	Process ID		VARCHAR(32)	NOT NULL
ProcessStatus	None	Process Status		SMALLINT	NOT NULL
TimeSubmitted	None	Time Submitted		DATETIME	NULL
RunAfter	None	Run After		DATETIME	NULL
TimeStarted	None	Time Started		DATETIME	NULL
TimeEnded	None	Time Ended		DATETIME	NULL
Submitter	None	Submitter		NVARCHAR(20)	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
Description	None	Description		NVARCHAR(250)	NULL
ProcessAlert	None	Process Alert		VARCHAR(1)	NULL
RecurrenceInd	None	Recurrence Ind		VARCHAR(1)	NOT NULL
RecurrType	None	Recurr Type		NVARCHAR(20)	NULL
RecurrFreq	None	Recurr Freq		INT	NOT NULL
RecurrDailyFreq	None	Recurr Daily Freq		SMALLINT	NOT NULL
RecurrDailyWeekDay		Recurr Daily Week Day		VARCHAR(1)	NOT NULL
RecurrWeeklyFreq		Recurr Daily Freq		SMALLINT	NOT NULL
RecurrWeeklySun		Recurr Weekly Sun		VARCHAR(1)	NOT NULL
RecurrWeeklyMon		Recurr Weekly Mon		VARCHAR(1)	NOT NULL
RecurrWeeklyTue		Recurr Weekly Tue		VARCHAR(1)	NOT NULL
RecurrWeeklyWed		Recurr Weekly Wed		VARCHAR(1)	NOT NULL
RecurrWeeklyThu		Recurr Weekly Thu		VARCHAR(1)	NOT NULL
RecurrWeeklyFri		Recurr Weekly Fri		VARCHAR(1)	NOT NULL
RecurrWeeklySat		Recurr Weekly Sat		VARCHAR(1)	NOT NULL
RecurrMonthlyFreq		Recurr Monthly Freq		SMALLINT	NOT NULL
RecurrMonthlyDay		Recurr Monthly Day		SMALLINT	NOT NULL
RecurrYearlyFreq		Recurr Yearly Freq		SMALLINT	NOT NULL
RecurrYearlyDay		Recurr Yearly Day		SMALLINT	NOT NULL
RecurrStartDate		Recurr Start Date		DATETIME	NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RecurrEndType		Recurr End Type		VARCHAR(1)	NOT NULL
RecurrEndDate		Recurr End Date		DATETIME	NULL
RecurrMonthlyOccur		Recurr Monthly Occur		SMALLINT	NOT NULL
RecurrMonthlyOccurFreq		Recurr Monthly Occur Freq		SMALLINT	NOT NULL
RecurrMonthlyOccurDay		Recurr Monthly Occur Day		SMALLINT	NOT NULL
RecurrMonthlyOccurInd		Recurr Monthly Occur Ind		VARCHAR(1)	NOT NULL
AppKey		App Key		VARCHAR(32)	NULL
AppRecordKey		App Record Key		VARCHAR(32)	NULL

Table FW_MasterProfiles

The Job Profile Header table contains information of the name of the group of processes that can be submitted as a single process.

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
QueueID(PK)		Queue ID		VARCHAR(32)	NOT NULL
ProcessID(PK)		Process ID		VARCHAR(32)	NOT NULL
ProcessStatus		Process Status		SMALLINT	NOT NULL
TimeSubmitted		Time Submitted		DATETIME	NULL
RunAfter		Run After		DATETIME	NULL
TimeStarted		Time Started		DATETIME	NULL
TimeEnded		Time Ended		DATETIME	NULL
Submitter		Submitter		NVARCHAR(20)	NULL
Description		Description		NVARCHAR(80)	NULL
ProcessAlert		Process Alert		VARCHAR(1)	NULL
RecurrenceInd		Recurrence Ind		VARCHAR(1)	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RecurrType		Recurr Type		NVARCHAR(20)	NULL
RecurrFreq		Recurr Freq		INT	NOT NULL
RecurrDailyFreq		Recurr Daily Freq		SMALLINT	NOT NULL
RecurrDailyWeekDay		Recurr Daily Week Day		VARCHAR(1)	NOT NULL
RecurrWeeklyFreq		Recurr Daily Freq		SMALLINT	NOT NULL
RecurrWeeklySun		Recurr Weekly Sun		VARCHAR(1)	NOT NULL
RecurrWeeklyMon		Recurr Weekly Mon		VARCHAR(1)	NOT NULL
RecurrWeeklyTue		Recurr Weekly Tue		VARCHAR(1)	NOT NULL
RecurrWeeklyWed		Recurr Weekly Wed		VARCHAR(1)	NOT NULL
RecurrWeeklyThu		Recurr Weekly Thu		VARCHAR(1)	NOT NULL
RecurrWeeklyFri		Recurr Weekly Fri		VARCHAR(1)	NOT NULL
RecurrWeeklySat		Recurr Weekly Sat		VARCHAR(1)	NOT NULL
RecurrMonthlyFreq		Recurr Monthly Freq		SMALLINT	NOT NULL
RecurrMonthlyDay		Recurr Monthly Day		SMALLINT	NOT NULL
RecurrYearlyFreq		Recurr Yearly Freq		SMALLINT	NOT NULL
RecurrYearlyDay		Recurr Yearly Day		SMALLINT	NOT NULL
RecurrStartDate		Recurr Start Date		DATETIME	NULL
RecurrEndType		Recurr End Type		VARCHAR(1)	NOT NULL
RecurrEndDate		Recurr End Date		DATETIME	NULL
RecurrMonthlyOccur		Recurr Monthly Occur		SMALLINT	NOT NULL

Field Name	Field Label	Description	Open Plan Data Type	Oracle Data Type	Nulls
RecurrMonthlyOccurFreq		Recurr Monthly Occur Freq		SMALLINT	NOT NULL
RecurrMonthlyOccurDay		Recurr Monthly Occur Day		SMALLINT	NOT NULL
RecurrMonthlyOccurInd		Recurr Monthly Occur Ind		VARCHAR(1)	NOT NULL
AppKey		App Key		VARCHAR(32)	NULL
AppRecordKey		App Record Key		VARCHAR(32)	NULL
CreateUser		Created By		NVARCHAR(20)	NULL
CreateDate		Created Date		DATETIME	NULL
ModUser		Last Updated By		NVARCHAR(20)	NULL
ModDate		Last Updated		DATETIME	NULL

Appendix C: If You Need Assistance

If you need assistance installing, implementing, or using Open Plan, Deltek makes a wealth of information and expertise readily available to you.

Customer Services

For over 30 years, Deltek has maintained close relationships with client firms, helping with their problems, listening to their needs, and getting to know their individual business environments. A full range of customer services has grown out of this close contact, including the following:

- Extensive self-support options through the Deltek Support Center.
- Phone and email support from Customer Care analysts
- Technical services
- Consulting services
- Custom programming
- Classroom, on-site, and Web-based training

Attention: Find out more about these and other services from the Deltek Support Center.

Deltek Support Center

The Deltek Support Center is a support Web site for Deltek customers who purchase an Ongoing Support Plan (OSP).

The following are some of the many options that the Deltek Support Center provides:

- Search for product documentation, such as release notes, install guides, technical information, online help topics, and white papers
- Ask questions, exchange ideas, and share knowledge with other Deltek customers through the Deltek Support Center Community
- Access Cloud-specific documents and forums
- Download the latest versions of your Deltek products
- Search Deltek's knowledge base
- Submit a support case and check on its progress
- Transfer requested files to a Customer Care analyst
- Subscribe to Deltek communications about your products and services
- Receive alerts of new Deltek releases and hot fixes
- Initiate a Chat to submit a question to a Customer Care analyst online

Attention: For more information regarding Deltek Support Center, refer to the online help available from the Web site.

Access Deltek Support Center

To access the Deltek Support Center:

1. Go to https://deltek.custhelp.com.
2. Enter your Deltek Support Center **Username** and **Password**.
3. Click **Login**.

> **Note:** If you forget your username or password, you can click the **Need Help?** button on the login screen for help.

Additional Documentation

The following table lists the Deltek documentation available for this release. Except where noted, all the user guides and quick reference guides listed in this table are available for download from the Deltek Support Center.

Document Name	Description
Deltek Open Plan Batch Processor Guide	This document contains information about running the Open Plan multi-instance batch processor.
Deltek Open Plan Guided Tour	This document introduces you to the basic Open Plan operations using sample project data that comes with your installation.
Deltek Open Plan Installation Guide	This document is meant to serve as a guideline for hardware and software requirements and provide your firm's IT department with information about the technical deployment architecture.
Deltek Open Plan Milestone Professional Interface Add-In Module	This document is meant to serve as a guideline for using the Milestones Link tool.
Deltek Open Plan Release Notes	This document contains a summary of the pre-installation information, new features and enhancements, database changes, and software issues resolved.
Deltek Open Plan Technical Overview and System Requirements	This document is meant to serve as a guideline for hardware and software requirements and provide your firm's IT department with information about the technical deployment architecture.

About Deltek

Better software means better projects. Deltek delivers software and information solutions that enable superior levels of project intelligence, management and collaboration. Our industry-focused expertise makes your projects successful and helps you achieve performance that maximizes productivity and revenue. www.deltek.com