

---

**eDeveloper™ V10**



---

# ***Mastering eDeveloper***

*The authoritative book on composing service-oriented solutions*

---

The information in this manual/document is subject to change without prior notice and does not represent a commitment on the part of Magic Software Enterprises Ltd.

Magic Software Enterprises Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms and conditions of the license agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or information recording and retrieval systems, for any purpose other than the purchaser's personal use, without the prior express written permission of Magic Software Enterprises Ltd.

All references made to third-party trademarks are for informational purposes only regarding compatibility with the products of Magic Software Enterprises Ltd.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of eDeveloper.

Magic® is a registered trademark of Magic Software Enterprises Ltd.

Btrieve® and Pervasive.SQL® are registered trademarks of Pervasive Software, Inc.

IBM®, Topview™, iSeries™, pSeries®, xSeries®, RISC System/6000®, DB2®, and WebSphere® are trademarks or registered trademarks of IBM Corporation.

This product also includes software provided by the ICU project. ICU 1.8.1 and later (c) 1995-2003 IBM Corporation and others All rights reserved.

Microsoft®, FrontPage®, Windows™, WindowsNT™, and ActiveX™ are trademarks or registered trademarks of Microsoft Corporation.

Oracle® and OC4J® are registered trademarks of the Oracle Corporation and/or its affiliates.

Linux® is a registered trademark of Linus Torvalds.

UNIX® is a registered trademark of UNIX System Laboratories.

GLOBEtrrotter® and FLEXIm® are registered trademarks of Macrovision Corporation.

Solaris™ and Sun ONE™ are trademarks of Sun Microsystems, Inc.

HP-UX® is a registered trademark of the Hewlett-Packard Company.

Red Hat® is a registered trademark of Red Hat, Inc.

WebLogic® is a registered trademark of BEA Systems.

Interstage® is a registered trademark of the Fujitsu Software Corporation.

JBoss™ is a trademark of JBoss Inc.

Systinet™ is a trademark of Systinet Corporation.

Clip art images copyright by Presentation Task Force®, a registered trademark of New Vision Technologies Inc.

This product uses the FreeImage open source image library. See <http://freeimage.sourceforge.net> for details.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by Computing Services at Carnegie Mellon University (<http://www.cmu.edu/computing/>).

Copyright © 1989, 1991, 1992, 2001 Carnegie Mellon University. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software that is Copyright © 1998, 1999, 2000 of the Thai Open Source Software Center Ltd. and Clark Cooper.

This product includes software that is Copyright © 2001-2002 of Networks Associates Technology, Inc All rights reserved.

This product includes software that is Copyright © 2001-2002 of Cambridge Broadband Ltd. All rights reserved.

This product includes software that is Copyright © 1999-2001 of The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.

All other product names are trademarks or registered trademarks of their respective holders.

Mastering eDeveloper

June 2006

Copyright © 2006 by Magic Software Enterprises Ltd. All rights reserved.





---

## ***Chapter 1: Navigation and Workspace***

<b>How do I Organize the Project Objects? .....</b>	<b>1</b>
To create a folder	
To delete a folder	
To move a folder	
Moving objects into a folder using the Folder column	
Moving objects into a folder using the Move command	
<b>How do I Separate Palettes? .....</b>	<b>4</b>
To separate combined palettes	
<b>How do I Locate Any Line in the Studio? .....</b>	<b>5</b>
To locate a line	
<b>How do I Quickly Jump to a Line Using Its Number? .....</b>	<b>6</b>
Jumping to a line	
<b>How do I Check If an Object Is Being Used and What Other Objects Use It? ...</b>	<b>7</b>
Using the cross-reference	
<b>How do I Bookmark My Current Location for a Quick Return? .....</b>	<b>9</b>
To bookmark your current location	
<b>How do I Quickly Reopen a Recently Opened Project? .....</b>	<b>10</b>
Opening a recent project	
<b>How do I Change the Number of Recently Opened Projects? .....</b>	<b>11</b>
Setting the number of recently opened projects	
<b>How do I Repeat an Entry in the Studio? .....</b>	<b>12</b>
Using Repeat	
<b>How do I Move an Entry in the Studio? .....</b>	<b>13</b>
Moving an entry in the Studio	
<b>How do I Replace an Entry in the Studio with Another ..... Entry?</b>	<b>14</b>
Replacing an entry	
<b>How do I Move Between the Studio Palettes? .....</b>	<b>15</b>
<b>How do I Keep the Property Sheet Showing a Single Section at a Time? .....</b>	<b>16</b>
Changing the property sheet display	
<b>How do I Easily Switch From One Project to Another? .....</b>	<b>17</b>
Switching between projects	
Adding a Module	
Deleting a Module	

---

## ***Chapter 2: Projects and Applications***

<b>How do I Create a New Project?</b> .....	19
Creating a new project	
<b>How do I Set the Icon for My Application?</b> .....	21
Setting the icon for your application	
<b>How do I Set the Caption of My Application?</b> .....	23
Setting the caption for your application	
<b>How do I Set a Default Context Menu for the Entire Application?</b> .....	24
Setting the context menu for your application	
<b>How do I Set or Change the Pulldown Menu for the Application?</b> .....	25
Setting the pulldown menu for your application	
<b>How do I Set the Application to Use Its Own Files for Colors, Fonts, and Keyboard Mapping?</b> .....	26
Setting color, font, and keyboard files for the application	
<b>How do I Read and Write Files from/to the Directory of the Project?</b> .....	27
<b>How do I Read and Write Files from/to the eDeveloper Directory?</b> .....	28
<b>How do I Read and Write Files from/to the System's Temporary Directory?</b> ....	29
<b>How do I Open an Existing Project?</b> .....	30
Using the Open Project dialog box	
Using Recent Projects	
Directly activating the .edp	
<b>How do I Transfer Objects From One Project to Another?</b> .....	33
Exporting Objects	
Importing Objects	

## ***Chapter 3: Models***

<b>How do I Define Reusable Interface Objects?</b> .....	35
Creating a control model	
<b>How do I Define Reusable Data Objects?</b> .....	38
Creating a field model	
<b>How do I Define a Data Source Column Based on a Model?</b> .....	41
Defining a data source column based on a model	
<b>How do I Unify and Standardize the Project's Data Fields and Visual Controls?</b>	42
Specifying a control for a data model	

---

**How do I Prevent an Object from Being Affected by Any Change of its Model's Properties?**  
44

- Manually breaking inheritance
- Automatically breaking inheritance

**How do I Set a Broken Property to Inherit its Value? . . . . . 46**  
Manually breaking inheritance

**How do I Change the Class of a Model? . . . . . 47**

**How do I Automatically Drop Form Controls Using a Specific Control Model? . 48**  
Selecting the control and model from the palette

**How do I Export a Program or Table While Keeping Their Models? . . . . . 49**  
Exporting with models

**How do I Share a Collection of Models with Several Projects? . . . . . 51**  
Sharing models as components

## ***Chapter 4: The eDeveloper Engine***

**How do I Define Application Level Events? . . . . . 53**  
Creating a global event

**How do I Work with the eDeveloper Engine as an Event-Driven Engine? . . . . . 55**  
The Concept of Events  
Creating a logic unit

**How do I Utilize the Event Hierarchy? . . . . . 57**  
Blocking a system event  
Adding functionality to a system event

**How do I Prevent an Internal Event (action) like Delete Line, from Occurring? 59**  
Blocking an internal event

**How do I Set a Dynamic Name for an Input/Output File? . . . . . 62**  
Setting the I/O file name

**How do I Set a Dynamic Name for a Data Source? . . . . . 64**  
Renaming a data source at the task level

**How do I delete a File or a Data Source in a Task that Handles the Same File or Data Source?**  
65

**How do I Prevent The Engine from Creating an Output File or Printing an Empty Page to the Printer when the Task Eventually Does not Output Anything? . . . . . 66**  
Setting IO device timing

**How Can I Set a New Task to Automatically Create Logic Units for the Basic Task Levels (Task and Record)? . . . . . 67**

---

---

Automatically creating Task and Record Logic Units	
<b>How Can I Set a New Task to Create No Default Logic Units?</b> .....	68
Turning off automatic creation of logic units	
<b>How do I Retrieve the New value of an Edit Control While Handling Its Events?</b>	69
<b>How do I Prevent the End-User from Modifying Any Record in a Task?</b> .....	70
Setting the data source access mode	
Task Properties Initial Mode	
Task Properties Options	
Field Level Edit	
<b>How do I set the Engine to Execute the Record Suffix Logic Unit Even if the Record has Not Been Updated?</b> .....	74
Forcing record suffix to execute	
<b>How do I Prevent the End User from Changing the Task Modes?</b> .....	75
Preventing users from changing task modes	
<b>How do I Run Two Tasks Concurrently?</b> .....	76
Setting a program to run concurrently	

## ***Chapter 5: Tasks***

<b>How do I Define a Program Dataview?</b> .....	79
Entering a Main Source	
Entering a Linked Source	
Entering a Virtual or Parameter	
<b>How do I Create a Simple Program?</b> .....	83
Creating “Hello world!” in eDeveloper	
Creating the Data View	
Creating your logic	
Creating your display	
Running your program	
<b>How do I Set a Program to Return a Value to the Calling Program?</b> .....	88
Creating a return value	
Using a return value	
<b>How do I Open a Data Source Using a Physical Name that is Different than the one Specified on the Data Repository?</b> .....	90
Renaming a data source at the task level	
<b>How do I Dynamically Change the Display Order of Records in a Program?</b> . . .	91
Using an expression for an index	
<b>How do I Create a Simple Batch Program?</b> .....	93

---

---

Creating a simple batch program	
<b>How do I Stop a Batch Task from Running Forever?</b> .....	94
<b>How do I Delete a Chunk of Records from a Data Table?</b> .....	95
Creating a batch delete task	
<b>How do I Create Tasks that Dump Data Records Into Flat Text Files and Vice Versa?</b>	97
Dumping records to a text file	
Reading records from a text file	
<b>How do I Define Global Values that Can be Dynamically Defined and Accessible by Various Programs?</b> .....	99
Defining global variables	
<b>How do I Dynamically Instruct the Task to Open a Different Form Than the Main Display Entry?</b> .....	100
Choosing forms at runtime	
<b>How do I Synchronize Parameters Between Called Program and Calling Program?</b>	101
<b>How do I Prevent the End-user from Adding New Records?</b> .....	102
Preventing the user from entering create mode	
<b>How do I Prevent the End-user from Deleting Existing Records?</b> .....	103
Preventing the user from entering delete mode	
<b>How do I Prevent the End-user from Modifying Existing Records?</b> .....	104
Preventing the user from entering modify mode	
<b>How do I Prevent the End-user from Modifying the Data in Specific Fields?</b> ...	105
Making a field non-modifiable	
<b>How do I Create a Selection List Program?</b> .....	106
Creating a selection list	
Using a selection list	
<b>How do I Properly Validate the Data Entered by the End-user?</b> .....	109
Using a control verification event	
<b>How do I Set the Tabbing Sequence of the Controls on the Form?</b> .....	111
Setting the tab order	
Setting the tab order for several controls at once	
<b>How do I Exit a Program from a Subtask Level?</b> .....	113
Allowing an Exit event to propagate	
<b>How do I Save My Changes in the Task Editor While Remaining in it?</b> .....	114
Saving changes while editing a program	
<b>How do I Create Logic in a Task?</b> .....	115
Entering a Header line	
Entering an Operation	

---

<b>How do I Create Operations in a Task?</b> .....	116
The Cnd: field	
<b>How do I Copy a Task as a Subtask?</b> .....	120
Copying a program to become a subtask	
<b>How do I Make a Subtask Become the Top Level Task?</b> .....	121
Moving a subtask to the top level	
<b>How do I Select a Column of a Table Control?</b> .....	122
<b>How do I Drop a Control or Variable on a Table to Create a New Column to the Left of the Highlighted Column?</b> .....	123
<b>How do I Place Several Controls on the Same Column in a Table Control?</b> .....	124
<b>How do I Add New Controls to a Form?</b> .....	125
Using the Variable Palette	
<b>How do I Drop a Control Multiple Times Consecutively?</b> .....	127
<b>How do I Select a Container Control (Like Tab or Table) Without Selecting the Controls that are Attached to it?</b> .....	128
<b>How do I Keep the Design of a Form for Future Re-use?</b> .....	129
Saving a form as a template	
Using a template form	
<b>How do I Select Several Controls at the Same Time?</b> .....	130
Selecting controls using a “rubber band”	
Selecting controls using Ctrl+Click	
<b>How do I Show a Table Control with Alternating Colors?</b> .....	131
Using alternating colors on a table	
<b>How do I Hide or Show the Table Control Dividers?</b> .....	132
<b>How do I Make Form Controls Fit the Form when it is Resized?</b> .....	133
<b>How do I Change the Tabbing Order of a Control?</b> .....	135
Setting the tab order	
<b>How do I Display a Control on Top of Another Control?</b> .....	136
Viewing the Z-order	
Manually Using Z-order	
Attaching the control	
<b>How do I Jump to the Main Form of the Task?</b> .....	138
<b>How do I View the Tab Order of All Controls?</b> .....	139
<b>How do I Undo the Last Modification in the Form Editor?</b> .....	140
<b>How do I Make Some Controls Have the Same Height or Width?</b> .....	141

---

<b>How do I Set a Property on Multiple Controls at the Same Time? .....</b>	<b>142</b>
Changing properties on a group of controls	
<b>How do I Change the Width of a Table Control Column?.....</b>	<b>143</b>
<b>How do I Move a Table Control Column? .....</b>	<b>144</b>
Moving a Table Control Column	
<b>How do I Move Between Tabs While Editing a Tab Control? .....</b>	<b>145</b>
<b>How do I Make a Control Transparent? .....</b>	<b>146</b>
Setting a transparent color	
<b>How do I Set a Default Push Button for the Form? .....</b>	<b>148</b>
Setting a Default Push Button	
<b>How do I Automatically Generate a Default Form Layout?.....</b>	<b>149</b>
Using the Form Generator	
<b>How do I Show a Dynamic Form Title?.....</b>	<b>150</b>
Using an expression in a form title	
<b>How do I Set an Icon for a Form? .....</b>	<b>151</b>
Choosing a form icon	
<b>How do I Set a Default Context Menu For All Controls of a Form?.....</b>	<b>152</b>
Setting the default context menu for a form	
<b>How do I Set a Context Menu for an Individual Control?.....</b>	<b>153</b>
Creating a field level context menu	
<b>How do I Improve Performance When a Task is Being Called Repeatedly by a Batch Task?</b>	
154	
Improving performance of batch subtasks	

## ***Chapter 6: Extended Logic***

<b>How do I Send an eMail?.....</b>	<b>157</b>
Connecting to the server	
Sending the eMail	
<b>How do I Add an Attachment to an eMail I Send?.....</b>	<b>159</b>
<b>How do I Receive An Email? .....</b>	<b>160</b>
A note on server types	
<b>How do I Handle eMail Attachments? .....</b>	<b>162</b>
<b>How do I Retrieve a Web Page or other URL Content?.....</b>	<b>163</b>
HTTPGet()	
<b>How do I Simulate a Keystroke?.....</b>	<b>164</b>

---

---

KbPut()	
<b>How do I Let The End-user Mark Several Records In a Table And To Handle The Marked Records Collection?</b> .....	166
Preparing the table for multi marking	
Handling marked records	
Other processing using MM functions	
<b>How do I Manipulate the Menu Entries to Become Invisible, Disabled or Checked?</b>	168
The menu entry name	
Menu Paths	
The Menu number	
MnuCheck()	
MnuEnabl()	
MnuShow()	
MnuAdd()	
MnuRemove	
MnuReset()	
MnuName()	
<b>How do I Call a DLL Function?</b> .....	173
Using CallDLL	
Using Invoke UDP	
<b>How do I Call a Program By Its Name?</b> .....	176
Calling a program by name	
<b>How do I Call a Program Dynamically By Its Index?</b> .....	177
Using Call by Expression	
Using CallProg()	
Selecting the program	
<b>How do I Retrieve the Name of the Control the User Parked On?</b> .....	179
LastPark()	
<b>How do I Retrieve the Name of the Control the User Clicked On?</b> .....	180
LastClicked()	
<b>How do I Retrieve a Value from the System Environment Settings?</b> .....	181
OSEnvGet	
<b>How do I Delete a Disk File?</b> .....	182
FileDelete()	
<b>How do I Copy a Disk File?</b> .....	183
FileCopy()	
<b>How do I Check if a File Exists on Disk?</b> .....	184
FileExist()	



---

<b>How do I Retrieve the Content of a File Directory? .....</b>	<b>185</b>
FileListGet()	
<b>How do I Rename a File on Disk? .....</b>	<b>186</b>
FileRename()	
<b>How do I Get the Size of a File on Disk? .....</b>	<b>187</b>
FileSize()	

## ***Chapter 7: Deployment***

<b>How do I deploy my project? .....</b>	<b>189</b>
<b>How do I Create a Cabinet File?.....</b>	<b>191</b>
Creating a cabinet file	
<b>How do I Create a Shortcut for my Application? .....</b>	<b>192</b>
Creating a shortcut to the project file	
Creating a shortcut to the eDeveloper engine	
Using your own icon	
<b>How do I Make the Runtime Engine Automatically Load a Cabinet File? .....</b>	<b>194</b>
Setting the Default Project in the Magic.Ini	
Specifying the cabinet file in the Shortcut	
<b>How do I Display a Splash Image on Loading an Application?.....</b>	<b>196</b>
Setting the Logo File	

## ***Chapter 8: Subforms***

<b>How do I Make the Subform Control Fit the Dimensions of the Called Program Form?.....</b>	<b>197</b>
Setting Autofit	
<b>How do I Manually Refresh the View of the Subform?.....</b>	<b>198</b>
Manually refreshing a subform	
<b>How do I Refresh the Subform View only on Modifying the Last Argument When Passing Several Arguments to the Subform? .....</b>	<b>200</b>
Using an expression on a subform automatic refresh	
<b>How do I Control the Visibility of a Subform When it is Placed on a Tab Control?.....</b>	<b>202</b>
Adding a subform to a tab	
<b>How do I Set the Subform to Be Tabbed Into from a Specific Control of the Parent Form? .....</b>	<b>203</b>
Setting the Tab Order of a Subform Control	

---

---

<b>How do I Automatically Return Back to the Parent Form by Tabbing Out of the Last Control of the Subform Display?</b> .....	204
Setting an event to exit a subform	
<b>How do I Execute Task Prefix/Suffix Logic of a Subform only on Opening the Subform Task for the First Time by its Parent?</b> .....	205
Coding a block that only executes when the subform executes the first time	
<b>How do I Execute the Task Prefix/Suffix Logic of a Subform Whenever the User Enters the Subform?</b> .....	206
Coding a block that only executes when the user enters the subform	
<b>How do I Execute Task Prefix/suffix Logic of a Subform Whenever the Subform is Refreshed?</b> .....	207
Coding a block that only executes when the subform is refreshed	
<b>How do I know which Autofit Option to use?</b> .....	208

## ***Chapter 9: Splitter***

<b>How do I Display Two Forms Using a Splitter?</b> .....	211
Setting up a split form	
<b>How do I set the Initial Proportions of the Split Screens?</b> .....	214
<b>How do I set the Parent Task Display on the Opposite Side of the Split Screen?</b>	215
Setting the parent display position	
<b>How do I Dynamically Set the Offset of the Splitter?</b> .....	216
Setting a dynamic offset	
<b>How do I Obtain the Current Offset of the Splitter?</b> .....	217
<b>How do I Keep the Splitter Location as Set at Runtime by the End-User?</b> .....	218
How to set a User State Identifier	
<b>How do I Set a Minimal Size of the Split Areas?</b> .....	220
Setting a minimal size	

## ***Chapter 10: Tree Control***

<b>How do I Display Data In A Tree-format?</b> .....	221
Implementing a tree control	
<b>How do I Properly Define an Hierarchical Data Source that Will Fit a Tree Control Display?</b>	223

---

<b>How do I Set Icons for the Tree Nodes?</b> .....	224
Using icons	
<b>How do I Show\hide the Expand\collapse Buttons?</b> .....	226
Turning on/off the expand/collapse button	
<b>How do I Add a New Child Node at Runtime?</b> .....	227
Creating a Child Node	
<b>How do I Add a New Sibling Node at Runtime?</b> .....	228
Creating a Sibling Node	
<b>How do I Explicitly Expand\Collapse Tree Nodes at Runtime?</b> .....	229
Raising an Expand or Collapse Node event	
<b>How do I Automatically Open the Tree with All Its Nodes or only Several Nodes Expanded?</b>	
230	
Setting Auto expand to a specific node	
Setting Auto expand to a tree level	
<b>How do I Set the Tree Control to Display the Expand Button only in the Relevant Nodes that</b>	
<b>Actually Have Child Nodes?</b> .....	233
Turning on preload	
<b>How do I Show\Hide the Connecting Lines of the Tree Control?</b> .....	234
Turning connecting lines on and off	
<b>How do I Show\Hide the Root Node as Part of the Tree Content?</b> .....	235
Turning show root on and off	
<b>How do I Skip to a Specific Tree Node?</b> .....	236
Using TreeNodeGoto()	
<b>How do I Respond to any Expand\Collapse Activity Performed on the Tree Control By the</b>	
<b>End-user?</b> .....	238
Capturing the collapse and expand events	
<b>How do I Respond to the End-user Movement from one Node to another?</b> .....	239
Capturing movement entering a node	
Capturing movement upon leaving a node	
<b>How do I Highlight the Entire Line of the Current Tree Node?</b> .....	240
Turning row highlight on and off	

## ***Chapter 11: More on Logic***

<b>How do I Create a Function?</b> .....	241
Function Scope	
Creating a Function	

---

---

<b>How do I Set the Function's Parameters? .....</b>	<b>244</b>
Creating parameters for a function	
<b>How do I Set the Return Value of a Function? .....</b>	<b>246</b>
Setting a return value for a function	
<b>How do I Create a Function That is Available for the Current Task Only? .....</b>	<b>247</b>
Creating a local function	
<b>How do I Create a Function That is Available For the Entire Project? .....</b>	<b>248</b>
Creating a global function	
<b>How do I Share a Function Between Several Projects? .....</b>	<b>249</b>
Creating a global function	
<b>How do I Iterate on a Series of Operations?.....</b>	<b>250</b>
LoopCounter()	
Creating a Block While operation	
<b>How do I Update a Variable? .....</b>	<b>252</b>
Using the Update operation	
<b>How do I Condition the Execution of an Operation Based on a Variable's Value?254</b>	
Entering an expression for an operation	
<b>How do I Retrieve the Sequential Number of a Record That is Handled by a Batch Program?</b>	
255	
Using Counter()	
<b>How do I Condition the Logic to Be Executed Only for the First Record in an online Task?</b>	
256	
Using IsFirstRecordCycle()	
<b>How do I Make the cursor jump to a Specific Control? .....</b>	<b>257</b>
Using CtrlGoto()	
<b>How do I Set Logic to Be Executed When the End-user Enters\Leaves a Control?259</b>	
Using Control Prefix	
Using Control Suffix	
<b>How do I Condition an Operation to be Executed only When the End-user Tabs from one Field to Another in a Certain Direction? .....</b>	<b>260</b>
<b>How do I Condition an Operation to be Executed only if the End-user Sequentially Tabs from one Control to another, or When the End-user Skips to a Specific Control?261</b>	
Using Control Prefix	
<b>How do I Retrieve the Newly Entered Data of an Edit Control, Rich Edit Control, and Multi Choice List Box While Remaining on the Control?.....</b>	<b>262</b>
Using EditGet()	
<b>How do I Identify from Which Control an Event Was Triggered? .....</b>	<b>263</b>

---

Using HandledCtrl()

**How do I Define an Event Handler to Be Executed Only When the User is Parked on a Specific Control? .....** 264

Using the Control name property of an event

## ***Chapter 12: Date & Time***

**How do I Retrieve the Current Date?.....** 265

Date Storage

Using Date()

**How do I Retrieve the Current Time? .....** 267

Using Time()

**How do I Retrieve the Current Time Using Milliseconds Precision? .....** 268

Creating a timestamp using mTime()

**How do I Increment a Date Value? .....** 269

**How do I Increment a Time Value?.....** 270

**How do I Increment the Value of Date-Time Combined Variables? .....** 271

Using AddDateTime()

About Variable References

**How do I Calculate the Difference Between Two Datetime Values? .....** 273

Using DifDateTime()

About Variable References

**How do I Calculate the Beginning of the Month of a Given Date?.....** 275

Using the BOM() function

**How do I Calculate the Beginning of the Year of a Given Date?.....** 276

Using the BOY() function

**How do I Calculate the End of the Month of a Given Date?.....** 277

Using the EOM() function

**How do I Calculate the End of the Year of a Given Date?.....** 278

Using the EOY() function

**How do I Retrieve the Name of the Day of the Week of a Given Day?.....** 279

Using the CDOW() function

**How do I Retrieve the Name of the Month of a Given Day?.....** 280

Using the CMonth() function

**How do I Calculate the Day\Month\Year Portion of a Given Date Value? .....** 281

**How do I Calculate the Hour\Minute\Seconds Portion of a Given Time Value? .....** 282

---

The Time Functions	
<b>How do I Calculate the Day of the Week of a Date, as a Number?</b> .....	283
Using DOW()	
<b>How do I Convert a Date Value to a String?</b> .....	284
Date Pictures	
Using DStr()	
<b>How do I Calculate a Date Value That Is Stored In a String?</b> .....	286

## ***Chapter 13: Handling GUI***

<b>How do I Enable the End-User to Park on a Control Only by Mouse?</b> .....	287
Setting Tab Into	
<b>How do I Change the Mouse Cursor Icon?</b> .....	289
Using the SetCrsr() function	
<b>How do I Copy Data Within an Application by Dragging It?</b> .....	291
<b>How do I Drag Data From eDeveloper to External Applications?</b> .....	292
<b>How do I Retrieve a Full Path Name From a Dragged File?</b> .....	293
<b>How do I Drag Data From External Applications to eDeveloper?</b> .....	294
<b>How do I Drag Data in Table Format From eDeveloper to Excel?</b> .....	296
<b>How do I Allow Drag and Drop Only Within eDeveloper so That External Applications Cannot Retrieve the Data?</b> .....	297
<b>How do I Display Different Text Than the Stored Values in a Choice Control?</b> .....	298
<b>How do I Display Special Characters in a Choice Control?</b> .....	300
<b>How do I Switch Between Choice Control Options in the Form Editor?</b> .....	301
<b>How do I Set Accelerators to Choice Control Options?</b> .....	302
<b>How do I Limit the Length of the Displayed Drop Down List of a Combo Box?</b> .....	303
<b>How do I Associate Controls to Different Tabs in a Tab Control?</b> .....	304
Showing controls on one tab	
Showing controls on all tabs	
<b>How do I Associate a Task to a Specific Tab in a Tab Control?</b> .....	306
<b>How do I Design Vertical Tab Controls?</b> .....	307
<b>How do I Associate Images to Different Tabs in a Tab Control?</b> .....	309
<b>How do I Implement a Radio Button?</b> .....	311
Containing a radio button in one control	

---

Using several controls for a radio button	
<b>How do I Set the Default Option for a Choice Control?</b> .....	314
<b>How do I Dynamically Set the Option List of a Choice Control?</b> .....	315
Creating a dynamic choice control	
<b>How do I Implement a Choice Control Whose Data Comes From a Database Table?</b>	317
Creating a choice control tied to a database table	
<b>How do I Combine Additional Options With a Data Bound Choice Control?</b> ..	320
<b>How do I Implement Logic With Push Buttons?</b> .....	321
1. Create your user event	
2. Put your push buttons on the form	
Create a logic unit to perform the desired logic when the event is raised	
<b>How do I Allow Keyboard Navigation to a Push Button?</b> .....	324
Creating a parkable push button	
<b>How do I Skip Verification Logic From Being Executed When a Push Button is Pressed?</b>	326
<b>How do I Create Image Buttons?</b> .....	327
Creating an image button	
<b>How do I Combine Image and Text on a Button?</b> .....	329
Specifying a Text on image button	
<b>How do I Specify the Text on a Parkable Push Button?</b> .....	330
Using an Init to specify push button text	
Using a default value to specify push button text	
Using the picture to specify push button text	
<b>How do I Set Up One Push Button to Affect Either the Subform or its Parent Task?</b>	333
Using the Task in focus property	
<b>How do I Set Accelerators to Push Buttons?</b> .....	335
Setting an accelerator to a push button	
<b>How do I Dynamically Create Rich Formatted Text?</b> .....	336
<b>How do I Retrieve Data From a Multiple Selection List Box?</b> .....	338
Using a Multiple Selection List box	

## ***Chapter 14: XML***

<b>How do I Create an XML Doc from Scratch?</b> .....	341
Initializing an XML Document	
<b>How do I Find an XML Schema?</b> .....	343
The XML Schema	

---

---

<b>How do I Create an XML View? .....</b>	<b>345</b>
Creating an XML View	
<b>How do I Update NodeId and ParentId? .....</b>	<b>347</b>
<b>How do I Access a Certain Compound in an XML File? .....</b>	<b>348</b>
Repeating elements	
Selecting one compound	
<b>How do I Modify an Existing XML Document? .....</b>	<b>350</b>
Accessing a parent record	
Accessing a child record	
<b>How do I Determine the eDeveloper Datatypes Corresponding to XML Datatypes?353</b>	
Viewing the schema setting	
<b>How do I Retrieve Data from an XML Doc in a Preferred Order? .....</b>	<b>355</b>
Creating an alternate index for an XML view	
<b>How do I Handle Multi-Occurrence Elements in an XML Doc?.....</b>	<b>356</b>
Displaying repeatable elements	
<b>How do I Allow Different Users Access to the Same XML Document? .....</b>	<b>358</b>
Setting the access mode for an XML document	
<b>How do I Create Different XML Docs Based on the Same Schema? .....</b>	<b>359</b>
<b>How do I Access an XML Document Stored in an eDeveloper Data Variable? ..</b>	<b>360</b>
Using a BLOB as a data source	
Using a BLOB as in IO file	
<b>How do I Validate an XML Document? .....</b>	<b>362</b>
Using XMLValidate()	
XMLValidate() syntax	
<b>How do I Retrieve Validation Errors of an XML Document?.....</b>	<b>364</b>
<b>How do I Handle Errors Encountered During XML Access?.....</b>	<b>365</b>
Creating a global error handler	
<b>How do I Determine the Encoding for an XML Document? .....</b>	<b>366</b>
Using XMLGetEncoding()	
<b>How do I Handle XML Data Which is Base64 Encoded?.....</b>	<b>367</b>
<b>How do I Pass an XML Document as a Parameter?.....</b>	<b>368</b>
<b>How do I Handle an XML Document with No Schema?.....</b>	<b>369</b>
<b>How do I Retrieve / Update /Insert Data According to a Certain Path in an XML Document?</b>	
370	
Retrieving XML Data	
Updating XML Data	



---

Inserting XML Data  
Deleting XML Data

**How do I Uniquely Identify a Data Element and Its Hierarchy Within an XML Document?**  
372

**How do I Handle Mixed-Content in an XML Document?** ..... 374

## ***Chapter 15: COM***

**How do I Define the COM Object That I Want to Use?** ..... 375  
Defining a COM object

**How do I Set a Single Event Handler for Several COM Objects of the Same Type?**378  
Setting a handler by class

**How do I Enable Event Handling for a COM Object While in a Descendant Task?**379

**How do I Call a COM Object Method?** ..... 380  
Calling a COM Method

**How do I Set or Get a Property of a COM Object?** ..... 382  
Using Get and Set Property with a COM Object

**How do I Change a Reference to a Certain COM Object?** ..... 384  
Changing a COM Library Reference

**How do I Set a Value of an Enumerated Type Parameter of a COM Object?** .. 386

**How do I Extract/Set Data From/To a Variant Type of a COM Object?** ..... 387  
Creating a Variant with VariantCreate()  
Extracting data from a Variant using VariantGet()  
Variant Data Attributes  
Variant Data Types

**How do I Determine the Type and Corresponding eDeveloper Attribute of a Variant Value Belonging to a COM Object?** ..... 392  
Extracting the attribute type using VariantAttr()  
Extracting the data type using VariantType()

**How do I Send and Retrieve Array Values from COM Objects?** ..... 394  
Passing an array to a COM object  
Fetching a Collection

**How do I Pass/Receive a COM Object as a Parameter To/From a COM Object?**400  
Receiving a COM Object

**How do I Re-use COM Object Definitions?** ..... 401

**How do I Keep an Instance of a COM Object Available Across Programs?** .... 402  
Defining a COM object in the Main Program

---

---

<b>How do I Trap Events Triggered by a COM Object?</b> .....	403
Creating an Event Handler for a COM event	
<b>How do I Handle an Error Triggered by a COM Object?</b> .....	404
<b>How do I Handle an ActiveX Control Without Displaying it on the Form?</b> .....	405
Using an invisible ActiveX Object	
<b>How do I Expose eDeveloper Logic as COM Methods?</b> .....	406
Creating a COM interface	
<b>How do I set a Class ID When Exposing eDeveloper Logic as a COM Server?</b> ..	408
<b>How do I Configure a COM Client Locally Accessing eDeveloper as a COM Server?</b>	409
<b>How do I Determine the COM Datatypes When Exposing eDeveloper Logic as COM Methods?</b> .....	410
Viewing and Changing Method Argument Details	

## ***Chapter 16: Components***

<b>How do I Reuse eDeveloper Objects Across Projects?</b> .....	411
Creating an eDeveloper Component	
<b>How do I Determine Which Data is Used by Your Component Builder?</b> .....	417
<b>How do I Load a Component Into My Project?</b> .....	418
Using an eDeveloper Component	
<b>How do I Implement Changes Done in Existing Component, Into a Host Application Using This Component?</b> .....	420
Changing a runtime component	
Changing a component in the Studio	
Renaming objects in a component	
<b>How Can I Provide My Own Help File for a Component?</b> .....	422
Implementing a Help file	
<b>How do I See Detailed Information About a Component's Objects?</b> .....	424
<b>How do I Access the Directory in which the Component Resides?</b> .....	425
<b>How do I Determine if a Currently Running Application is a Component?</b> .....	426
<b>How do I Dynamically Call a Program Within Another Application not Defined as a Component?</b> .....	427
Using Call Remote	
Using Call by Name	
<b>How do I Handle Recursive Calls Between Applications?</b> .....	430
<b>How do I Implement Environmental Requirements for a Component?</b> .....	431

---

<b>How do I Optimize Access to a Component? .....</b>	<b>434</b>
---	------------

## ***Chapter 17: Environment***

<b>How do I Use My Windows Login to Log on to eDeveloper? .....</b>	<b>435</b>
Using the Windows userid to log in to eDeveloper	
Considerations	
<b>How do I Automatically Have a Project Opened When Invoking the Studio? ..</b>	<b>437</b>
Setting the Default Project	
Testing server applications	
<b>How do I Force the Runtime Engine to Run its Application as an SDI Application?</b>	<b>439</b>
Creating an SDI Application	
<b>How do I run a program as an SDI program? .....</b>	<b>440</b>
Defining an SDI context	
Modifying the SDI context	
<b>How do I Specify the Screen Refresh Rate in a Batch Task? .....</b>	<b>443</b>
Specifying the batch event interval	
<b>How do I Determine the Interval Being Used by the Engine to Poll Async Events?</b>	<b>444</b>
<b>How do I Implement Transactions with ISAM Files? .....</b>	<b>445</b>
ISAM Transactions	
ISAM-Force Locking Within Transaction	
<b>How do I Determine the Location of the eDeveloper Locking File? .....</b>	<b>447</b>
<b>How do I Prevent the Creation of an I/O File Until it is Actually Used? .....</b>	<b>448</b>
<b>How do I Optimize Image Access During Runtime? .....</b>	<b>449</b>
Image Cache Size	
Check image change time	
<b>How do I Control the Displayed Checker Messages? .....</b>	<b>450</b>
Studio Checker Minimal Level	
Group Checker Messages by	
Jump Automatically to the first item in checker list	
Allow Access to Checker Messages	
<b>How do I Develop an Application Using Components Without the Need to Create a Cabinet File for Each Component? .....</b>	<b>453</b>
Running a Component from a Project File	
<b>How do I Determine the Codepage for Translation of ANSI Strings to Unicode and Vice Versa? .....</b>	<b>455</b>

---

## ***Chapter 18: Defining Data Sources***

<b>How do I Create a Database Table Using eDeveloper?</b> .....	457
Make sure the gateways and DBMS are loaded	
Set up the Database definition	
Create the table	
Create the columns	
Create the indexes	
Syntax check the table	
Test the Table by creating a few records	
<b>How do I Access an Existing Database Table?</b> .....	464
Accessing an existing database table	
<b>How do I Retrieve Data from a Database View?</b> .....	466
<b>How do I Alter a Database Table Definition?</b> .....	467
Altering database tables defined only in eDeveloper	
Altering the eDeveloper data source to match an external table	
<b>How do I Define the Mapping Between an eDeveloper Field and a Database Column?</b>	469
Def/Null	
Storage	
SQL	
<b>How do I Set Data Source Mappings to Support Working with Multiple DBMS's?</b>	472
Defining fields to be portable	
Creating portable WHERE clauses	
<b>How do I Define a Table Entry for Non- Persistent data?</b> .....	474
Creating a Memory Table	
<b>How do I Retrieve Records from a Database Table in a Predefined Order?</b> ....	475
Specifying the Index	
Specifying direction for a Main Source	
Specifying direction for a linked source	
Creating an index on the fly	
<b>How do I Browse a Database Table?</b> .....	478
Creating a Browse Program	
<b>How do I Dump Data from a Database Table to a Text File?</b> .....	479
Creating a text export program	
<b>How do I Load Data to a Database Table from a Text File?</b> .....	480
Creating a text import program	
<b>How do I Fetch Data from a Database Table a Single Time for the Whole Application?</b>	481
Resident Settings	

---

## **How do I Determine the Bulk Size When Fetching Records from a Database Table?**483

- DbSize()
- DbRecs()
- DbViewSize()

## **How do I Dynamically Set a Data Source Name?** ..... 485

- Overriding the Data source name in a declaration
- Overriding the Data source name in a function
- Using Logical Names for Data sources

## ***Chapter 19: Main Program***

### **How do I Initialize My Application?**..... 487

- What to set up in the Data View section of the Main Program
- What to set up in the Logic section of the Main program

### **How do I Skip Initialization Code?**..... 489

### **How do I Implement a Background /Wallpaper for My Application?** ..... 490

- Implementing a background in the Main Program

### **How do I Set and Use Global Variables (per Context Only)?** ..... 492

- Using Global Variables

### **How do I Run a Certain Procedure Upon Application Invocation?** ..... 494

### **How do I Run a Certain Procedure Upon Application Termination?**..... 495

## ***Chapter 20: Data View***

### **How do I Retrieve the Number of Records in a Task's Data View?**..... 497

- DbViewSize()

### **How do I Determine a Task's Main Data Source?**..... 499

- Entering the Main Source

### **How Can I Dynamically Set the Order of the Retrieved Records in a Task?** ... 501

- Using a Hard-coded Index
- Using an Index Expression
- Using a Task Sort

### **How do I Define the Incoming and Outgoing Arguments in a Program?** ..... 505

- Parameters
- Returned Values

### **How do I Define Task Variables?** ..... 507

- Creating a Column

---

---

<b>How do I Define a Range for a Task's Data View?</b> .....	510
Using a From/To Range	
Multiple Views of the From/To Range	
Using a Range Expression	
Using an SQL Where clause	
<b>How do I Make the Task Position on a Certain Record When it Starts?</b> .....	516
Using the Locate property	
The effect of Locate Order	
Using both Locate From and Locate To	
Using a Locate Expression	
Centering the record	
<b>How do I Access a Data Source for Read-only in a Task?</b> .....	519
How to open a Data Source as read-only	
<b>How do I Determine Access Between Multiple Users for a Data Source, Within a Task?</b>	520
<b>How do I Set the Value of a Task Variable?</b> .....	521
The Default Value Property	
The Init Property	
Update Operation	
VARSet()	
<b>How do I Retrieve Data From Multiple Tables in a Single Task?</b> .....	525
Setting up the Link Locate columns	
Kinds of links	
The Link Success indication property	
The Link Condition	
Using Range on Linked columns	
<b>How do I Fetch the First or Last Record of a Table in a Task?</b> .....	531
Range Order vs. Locate Order	

## ***Chapter 21: Expressions***

<b>How do I Format an Expression in the Expression Window?</b> .....	537
Formatting an Expression	
<b>How do I Automatically Complete the Name of a Typed-in Function?</b> .....	539
Using the Auto-complete feature	
<b>How do I Set the Colors of the Colored Elements in the Expanded Expression View?</b>	540
Setting the Expression Colors	
<b>How do I get to the Help Page of a Function?</b> .....	542
Finding a Help Page for a function from anywhere	

---

Finding Help from the Expressions List	
Finding Help from the Functions list	
<b>How do I Manually Expand the Expression Line to Make it Easier to Edit Large Expressions?</b>	544
<b>How do I Syntax Check an Expression while in Wide Mode?</b>	545
<b>How do I Enter Line Breaks into String Values?</b>	546
<b>How do I Set an Expression Directly in the Task Editor?</b>	547
Entering a Quick Expression	
<b>How do I Open the Variable List from the Expression Editor?</b>	548
<b>How do I Open the Functions List from the Expression Editor?</b>	549
Selecting a Function from the Functions List	
<b>How do I Find Where an Expression is Being Used?</b>	551
Using Find Reference on an Expression	
<b>How do I Re-use Expressions Within Another Expression?</b>	552
Using a User Expression	
Using ExpCalc()	
<b>How do I Construct and Evaluate an Expression at Runtime?</b>	555
Using EvalStr	
Using EvalStrInfo	
<b>How do I Repeat Existing Expression in a Task?</b>	557
Using Repeat Line (@)	
<b>How do I Find All Unused Expressions?</b>	558
Finding unused expressions	
<b>How do I Clear All Unused Expressions?</b>	559
<b>How do I Define a String Value in an Expression?</b>	560
<b>How do I Avoid Creating Duplicate Simple Expressions?</b>	561

## ***Chapter 22: Reports***

<b>How do I Dynamically Export the Dataview of a Task into an XML, HTML, Text, or CSV file?</b>	563
<b>How do I Allow the End-user to Dynamically Export Data?</b>	564
Creating the data view program	
Using the Print Data Utility	
<b>How do I Create a Quick Browse Program?</b>	567
Generating a simple browse program	

---

---

<b>How do I Dump the Current View to a Text File in HTML, XML or Simple Delimited Format?</b> .....	568
<b>How do I Export Data into a Text File?</b> .....	569
Using DataViewToText()	
<b>How do I export Data Into a CSV File?</b> .....	571
<b>How do I Export Data Into an HTML File?</b> .....	572
Using DataViewToText()	
Using the HTML Template File	
<b>How do I Export Data Into an XML File?</b> .....	576
Using DataViewToXML()	
Creating a schema file	
Using an XML Template file	
<b>How do I Create a Report?</b> .....	578
1. Create a “Launch screen”	
2. Create a simple text export program	
3. Check your sort order and range	
4. Set up the batch task	
5. Set up the I/O device	
Create your forms	
Editing your forms	
Making the detail form print	
Printing the header	
Printing the footer	
<b>How do I Define Page Header and Footer Information?</b> .....	588
<b>How do I Define a Global Page Header or Footer?</b> .....	590
Creating and using a global form	
Handling variable data on the global page header or footer	
<b>How do I Enumerate Report Pages?</b> .....	592
1. Set up two I/O Devices	
2. Set up the calling task	
3. In Task Prefix, zero out report variables	
4. Create a Page Header event to count the pages	
5. Store the total number of pages on the first go-around	
6. Output to two I/O devices	
<b>How do I Print a Report from Several Programs to the Same I/O Device?</b> .....	597
Set up the calling program	
Setting up the called program	
<b>How do I Create Report Break Levels?</b> .....	600
1. Make sure your record order matches the break level	



- 
2. Set up a variable that will change at the right time
  3. Set up your Group levels
  4. Sum the totals

<b>How do I Define Aggregates per Break Level? .....</b>	<b>603</b>
1. Define the aggregate variables	
2. Update the aggregates	
3. Zero out the aggregates	
<b>How do I Include All Data From a Multi-Line Control in My Report? .....</b>	<b>605</b>
1. Change the field Property	
2. Change the Form Property	
<b>How do I Set Repeating Captions for a Table in a Report? .....</b>	<b>607</b>
<b>How do I Produce PDF Documents? .....</b>	<b>608</b>
Setting up for PDF when the user chooses the output	
Setting up PDF as a default print preview	
Setting up a PDF for batch jobs	
<b>How do I Implement Page-Based Calculations? .....</b>	<b>612</b>

## ***Chapter 23: Merge***

<b>How do I Merge Data Into a Text File? .....</b>	<b>613</b>
1. Create your text template	
2. Specify your output file	
3. Create your Merge form	
4. Set up the Merge form Properties	
5. Select the Tags	
6. Associate each Tag with data	
7. Output the merge form	
<b>How do I Create a Dynamic Word Document? .....</b>	<b>619</b>
<b>How do I Create a Dynamic Excel Document? .....</b>	<b>621</b>
<b>How do I Insert Repeatabe Data into a Predefined Template? .....</b>	<b>623</b>
<b>How do I Insert Repeatabe Data into a Table Format in a Predefined HTML Template? .....</b>	<b>624</b>
<b>How do I Condition Inserting Data into a Predefined Template? .....</b>	<b>625</b>
Specifying a condition	
<b>How do I Set Up Replaceable Tokens in a Predefined Template? .....</b>	<b>626</b>
<b>How do I Differentiate HTML Tags and Merge Tokens? .....</b>	<b>627</b>
<b>How do I Set the Preferred HTML Editor of My Choice? .....</b>	<b>628</b>

---

---

<b>How do I Merge Data into One Document From Several Tasks? .....</b>	<b>629</b>
Merging data from two tasks in the same program	
Merging data from two different programs	
<b>How do I Present Data as Grouped, Within a Predefined HTML Template? ...</b>	<b>632</b>
1. Setting up the template	
2. Writing the detail line	
3. Writing the header	
<b>How do I Embed a File into a Predefined Template? .....</b>	<b>638</b>
Syntax of <!\$MGINCLUDE>	

## ***Chapter 24: Messaging***

<b>How do I send a Message to MSMQ? .....</b>	<b>639</b>
1. Open Queue	
2. Send Message	
3. Close Queue	
<b>How do I Receive an MSMQ message?.....</b>	<b>644</b>
1. Open the Message Queue	
2. Read messages	
3. Close queue	
<b>How do I Set a Label for a Message Sent to MSMQ?.....</b>	<b>647</b>
Setting a the MSMQ Label property	
Send the table to the component	
<b>How do I Set the Priority for a Message Sent to MSMQ? .....</b>	<b>648</b>
Setting a the MSMQ Priority property	
Send the table to the component	
<b>How do I Access a Public MSMQ Queue? .....</b>	<b>649</b>
<b>How do I Verify That a Message Sent to MSMQ, Was Read? .....</b>	<b>650</b>
Setting a the MSMQ Ack property	
Send the table to the component	
<b>How do I send the MSMQ External Message Table to a the Message Setup Program?</b>	<b>651</b>
<b>How do I Trap Messaging Errors? .....</b>	<b>652</b>
<b>How do I Debug MSMQ Applications? .....</b>	<b>654</b>
Viewing Messages in Windows	
<b>How do I Set Up My Computer for MSMQ?.....</b>	<b>656</b>
Installing MSMQ	
Adding Queues	
Installing the Messaging Component	

---

## ***Chapter 25: Multi-Tasking***

<b>How do I Run More Than One Interactive Task Simultaneously? .....</b>	<b>659</b>
Making a task run in parallel	
<b>How do I run a Report or Batch Process in Parallel to Other Tasks?.....</b>	<b>662</b>
Making a batch task run in parallel	
<b>How do I Manage Communication Between Parallel Tasks? .....</b>	<b>663</b>
<b>How do I Retrieve a List of All Running Contexts? .....</b>	<b>666</b>
<b>How do I Fetch the Name of the Current Context? .....</b>	<b>667</b>
<b>How do I Set a Different Name for the Current Context? .....</b>	<b>668</b>
<b>How do I Switch Control to a Different Context? .....</b>	<b>669</b>
<b>How do I Retrieve the Context ID of a Called Parallel Program? .....</b>	<b>670</b>
<b>How do I Share Variables Amongst Contexts? .....</b>	<b>671</b>
Using SharedValSet()	
Using SharedValGet()	
<b>How do I Share Memory Tables Amongst Contexts?.....</b>	<b>673</b>
<b>How do I Control the Initialization of a Parallel Task?.....</b>	<b>674</b>
<b>How do I Force a Single Instance of a Running Program? .....</b>	<b>676</b>
<b>How do I Identify a Repetitive Call to a Single Instance Program? .....</b>	<b>677</b>

## ***Chapter 26: Window Interface***

<b>How do I Keep the End User's Form Customization?.....</b>	<b>679</b>
Using Form State Persistency	
<b>How do I Set the Icon for a Window?.....</b>	<b>681</b>
<b>How do I Dynamically Set the Caption for a Window? .....</b>	<b>682</b>
Setting the Form name property to an expression	
<b>How do I Prevent the User from Resizing a Window? .....</b>	<b>683</b>
Preventing resizing	
<b>How do I Set a Minimal Size for a Window?.....</b>	<b>684</b>
Setting the minimal size	
<b>How do I Remove the Window Title? .....</b>	<b>685</b>

---

## ***Chapter 27: Menuing***

<b>How do I Set an Accelerator for a Menu Entry? .....</b>	<b>687</b>
Setting Accelerators for non-Internal events	
Setting Accelerators for Internal Events	
<b>How do I Change the Menu of a Running program? .....</b>	<b>689</b>
MnuAdd()	
MnuRemove()	
MnuReset()	
<b>How do I Enable Keyboard Window Switching? .....</b>	<b>693</b>
Using the Windows Menu	
<b>How do I Hide/Reveal a Menu Entry? .....</b>	<b>695</b>
Using MnuShow()	
<b>How do I Remove a Menu Bar? .....</b>	<b>697</b>
<b>How do I Add an Icon to a Menu Entry? .....</b>	<b>698</b>
Specifying a menu icon	
Choosing your own icon	
<b>How do I Add an Icon to the Toolbar? .....</b>	<b>700</b>

## ***Chapter 28: Unicode***

<b>How do I Enable Support for Multi-Lingual Data? .....</b>	<b>701</b>
The Unicode Font	
The Unicode Attribute	
<b>How do I Create or Read a Unicode text file? .....</b>	<b>705</b>
<b>How do I Convert Between ANSI and Unicode? .....</b>	<b>706</b>
Converting from Unicode back to ANSI	
<b>How do I Find the Unicode Value of a Character? .....</b>	<b>708</b>
<b>How do I Convert the Unicode Value to a Unicode Character? .....</b>	<b>709</b>

## ***Chapter 29: Application Debugging***

<b>How do I Debug my Application Using the Debugger? .....</b>	<b>711</b>
<b>How do I Use the Debugger When Running Parallel Programs? .....</b>	<b>714</b>
<b>How do I Set Breakpoints in the Application? .....</b>	<b>715</b>
Turning breakpoints on and off	

Finer control of breakpoints (breakpoint properties)	
Go to source	
Setting a watch on variables	
<b>How do I Control the Information Logged by the Debugger? .....</b>	<b>720</b>
The Logging() function	
Performance issues	
<b>How do I Manipulate Data While Debugging? .....</b>	<b>722</b>
<b>How do I Debug a Component? .....</b>	<b>723</b>
<b>How do I Log the Database Activity? .....</b>	<b>724</b>
Log Level	
DBMS Logging	
Performance issues	

## ***Chapter 30: Events and Handlers***

<b>How do I Refresh the Variable's Value After Returning From a Selection Program Opened by an Event? .....</b>	<b>727</b>
Solution 2: Attach the selection list to the control	
<b>How do I Force a Handler to Use the Newly Updated Values That Have Not Yet Been Committed? .....</b>	<b>731</b>
<b>How do I Handle the Same Event Using Several Handlers? .....</b>	<b>732</b>
Allowing an event to propagate	
<b>How do I preserve eDev Functionality for Internal Events Handled by User Handlers?733</b>	
<b>How do I Ensure That the Handling of an Event Will Only be Active in the Task Where It Was Declared? .....</b>	<b>734</b>
<b>How do I Disable eDeveloper Handling for Internal Events? .....</b>	<b>735</b>
Using Conditional Propagate	
<b>How do I Restrict the Handling of an Event to a Specific Control?.....</b>	<b>737</b>
<b>How do I Provide a Default Handler for my Entire Application? .....</b>	<b>738</b>
<b>How do I Handle Events Raised in a Hosting Application Using a Handler Defined in a Component? .....</b>	<b>739</b>
Define the event in the Component	
Raising the event in the Host	
Handling the event in the Component	
<b>How do I Raise Events Defined in the Host Application From a Component?..</b>	<b>742</b>
Raising the host event	
Handling the event in the host	

---

---

<b>How do I Force Immediate Handling for Events? .....</b>	<b>744</b>
<b>How do I Postpone the Handling of an Event? .....</b>	<b>745</b>
<b>How do I Raise A Dynamically Named Event? .....</b>	<b>746</b>
Set up the event in the Main Program	
Call the event with a Call public	
<b>How do I Execute a Set of Operations After a Time Interval? .....</b>	<b>748</b>
Using a Timer event	
<b>How do I Execute a Set of Operations When a Condition is Met? .....</b>	<b>749</b>
Creating an Expression event	
<b>How do I Send Information With a Raised Event? .....</b>	<b>750</b>
Create the event with parameters	
Accept the parameters in your Event handler	
Call the event	

## ***Chapter 31: Security***

<b>How do I Encrypt And Decrypt Data? .....</b>	<b>753</b>
Using Cipher()	
Using Decipher()	
Supported encryption methods	
<b>How do I Encrypt Data Inside a Table? .....</b>	<b>757</b>
Encrypting a database table	
<b>How do I Hide Database Login Information? .....</b>	<b>758</b>
Setting a Secret Name	
Using a Secret Name	
<b>How do I Declare Administrator Rights in an Application? .....</b>	<b>760</b>
Deleting the Security file	
<b>How do I Limit Execution of Specific Programs? .....</b>	<b>762</b>
<b>How do I Customize the Menu According to the User Logged On? .....</b>	<b>763</b>
Securing menu items	
<b>How do I Limit Functionality According to the User Logged On? .....</b>	<b>765</b>
Creating a Rights() Expression	
Making a control disappear for unauthorized users	
Preventing logic from executing for unauthorized users	
<b>How do I Restrict Access to the Entire Application? .....</b>	<b>768</b>
Using Security keys	
<b>How do I Implement Roles? .....</b>	<b>770</b>

## **Chapter 32: Utilities**

<b>How do I Browse a Data Source? .....</b>	<b>775</b>
<b>How do I Create Simple Browse Program for a Data Source? .....</b>	<b>777</b>
Creating a browse program	
<b>How do I Syntax Check a Program? .....</b>	<b>780</b>
Syntax checking one program	
Syntax checking many programs at once	
<b>How do I Validate a Data Source Structure? .....</b>	<b>782</b>
Syntax checking one Data source	
Syntax checking many Data sources at once	
<b>How do I Filter the Messages Shown by the Checker? .....</b>	<b>784</b>
Set the Checker Minimal Level	
Setting the Level of a message	
<b>How do I Use the Checker Results? .....</b>	<b>786</b>
Using the Checker Results List	
Using Ctrl+F8	
<b>How do I Back Up the Project? .....</b>	<b>787</b>
Backing up the operating system files	
Creating export files	
Making copies of programs as you work	
<b>How do I Search and Replace Text in the Project's Objects? .....</b>	<b>789</b>
Finding Text	
Replacing Text	
<b>How do I Save or Print the Search Results? .....</b>	<b>792</b>
Saving the search results	
Printing the search results	
<b>How do I Override eDeveloper Functions? .....</b>	<b>793</b>
<b>How do I Account for Incompatibilities in Table Structure Between eDeveloper and the Database? .....</b>	<b>794</b>
Allowing eDeveloper to automatically convert the table	
Checking for compatibility	
<b>How do I Export a Pervasive ISAM Table Structure to be Used with External Tools? .....</b>	<b>798</b>
<b>How do I Add External Tools to the Studio? .....</b>	<b>799</b>
Adding an external tool	

---

---

<b>How do I Run External Processes Automatically? .....</b>	<b>802</b>
Creating the command file	
Setting up the command file to automatically execute	
Setting up the automatic processing in batch	
<b>How do I Limit the Use of my Application Using the eDeveloper License Mechanism?804</b>	
Running the MakeKey utility	
Using LMChkOut()	
Using LMChkIn()	
<b>How do I Find Which Computers Use a Specific eDeveloper License?.....</b>	<b>806</b>

## ***Chapter 33: Studio Configuration***

<b>How do I Prevent the Property Sheet and the Navigator From Appearing Automatically?807</b>	
<b>How do I Separate the Palettes Once They are Merged? .....</b>	<b>809</b>
To separate combined palettes	
<b>How do I Change the Studio Color and Fonts?.....</b>	<b>810</b>
Changing the Studio Colors	
Changing the Studio Fonts	
<b>How do I Automatically Load a Project Upon Running the Studio? .....</b>	<b>815</b>
<b>How do I Control the Location of the XML Source Files of a Project? .....</b>	<b>816</b>
Changing the value of the Source directory in an existing .edp	
Changing the default source directory	
<b>How do I Define the Location of the .ini?.....</b>	<b>818</b>
Using a different .ini file	
<b>How do I Add Additional .ini Settings? .....</b>	<b>820</b>
Overriding .ini settings	
Using the overrides	

## ***Chapter 34: Consuming Web Services***

<b>How do I Access a Web Service? .....</b>	<b>821</b>
<b>How do I Reload Web Service Definitions?.....</b>	<b>824</b>
<b>How do I Send or Receive Complex Arguments?.....</b>	<b>825</b>
<b>How do I Securely Access a Web Service?.....</b>	<b>829</b>
<b>How do I Work With Web Service Attachments? .....</b>	<b>831</b>



---

## ***Chapter 35: Providing Web Services***

.....	833
<b>How do I Provide Web Services With eDeveloper?</b> .....	833
<b>How do I Deploy a Web Service Module?</b> .....	842
<b>How do I Test a Web Service?</b> .....	843
<b>How do I Provide Web Services With Attachments?</b> .....	846
<b>How do I Trace SOAP Requests?</b> .....	849
<b>How do I set up Authorization for a Deployed Service?</b> .....	850

## ***Chapter 36: Database***

<b>How do I Define a Connection to a Database?</b> .....	855
<b>How do I Create a Database Table From eDeveloper?</b> .....	864
<b>How do I Access an Existing Database Table or View?</b> .....	865
<b>How do I View SQL Statements Sent by eDeveloper to the Database?</b> .....	866
<b>How do I Send My Own SQL Statements to the Database?</b> .....	867
<b>How do I Handle a Database Error or Exception?</b> .....	870
<b>How do I Limit the End User's Access to and Manipulation of the Data?</b> .....	872
<b>How do I Determine the Order of the Records Retrieved from the Database?</b> ..	873
<b>How do I Access a Specific SQL Type?</b> .....	875
<b>How do I Minimize Database Access for Read-only Data?</b> .....	877
<b>How do I Reduce Database Access?</b> .....	879
<b>How do I Affect the Select Statement Sent to the Database?</b> .....	882
<b>How Can I Determine eDeveloper Behavior When Several Users are Modifying the Same Row?</b> .....	884
<b>How do I Implement a One-to-Many Relationship in eDeveloper when there is a Database Constraint?</b> .....	888
<b>How do I Implement a Nested Transaction?</b> .....	889
<b>How do I Force Writing the Current Record to the Database?</b> .....	891
<b>How do I Refrain From Opening a Transaction?</b> .....	892
<b>How do I Explicitly Roll Back a Transaction?</b> .....	893
<b>How do I Affect the Database Optimizer Behavior?</b> .....	894

---

How do I Initiate a Database Transaction? .....	895
---	-----

## ***Chapter 37: Browser***

How do I Set My Preferred HTML Editor? .....	899
How do I Implement JavaScript Functions Within the Application? .....	901
How do I Implement a One-to-Many Relationship? .....	902
How do I Prevent a New Window From Being Opened When Calling a Task or Program?	906
How do I Display the Interface of different Programs in the Same Window? ...	907
How do I Direct the Browser to a Given URL When Closing the Top Level Program?	909
How do I Map Keyboard Strokes to eDeveloper Internal Events? .....	911
How do I Distinguish Between Server Side and Client Side Operations and Functions?	913
How do I Set the Number of Repeatable Records in a Table? .....	914
How do I Set the Number of Records in a Record Set that are Passed to the Browser?	916
How do I Add Controls to an Existing Form? .....	917
How do I Implement Styles and Classes? .....	919
How do I Implement ActiveX Controls on a Page? .....	924
How do I Invoke eDeveloper Logic from an External Script in an HTML Page?	927
How do I Allow a Program to be Called Externally? .....	928

## ***Chapter 38: The Broker***

How do I Configure an IIS Web Server so eDeveloper Will Receive Requests? .	929
How do I Configure an Apache Web Server so eDeveloper Will Receive Requests?	935
How do I Monitor Broker Activity? .....	939
How do I Limit the Number of Requests That eDeveloper Will Handle Simultaneously?	941
How do I Configure the Broker to Automatically Load an Application? .....	942
How do I Define the Broker Password? .....	943
How do I Define an Alternate Broker? .....	944
How do I Disable the Runtime Engine From Serving Broker Requests? .....	945
How do I Remove a Request Waiting in the Queue? .....	946
How do I Implement Load Balancing .....	947

---

## ***Chapter 39: Source Management***

<b>How do I Create a Project to be Managed by Version Control? .....</b>	<b>949</b>
<b>How do I Add an Existing Project to a Version Control Database?.....</b>	<b>951</b>
<b>How do I Remove a Project from Version Control? .....</b>	<b>955</b>
<b>How do I Add a New Developer to a Project Managed by Version Control?....</b>	<b>956</b>
<b>How do I Develop a Project Managed by Version Control, When the Version Control Server is not Available? .....</b>	<b>958</b>
<b>How do I Track Changes? .....</b>	<b>961</b>
<b>How do I Determine the Version Control Provider? .....</b>	<b>965</b>
<b>How do I Rollback to a Prior Copy of an Object?.....</b>	<b>966</b>
<b>How do I Work with CVS? .....</b>	<b>967</b>

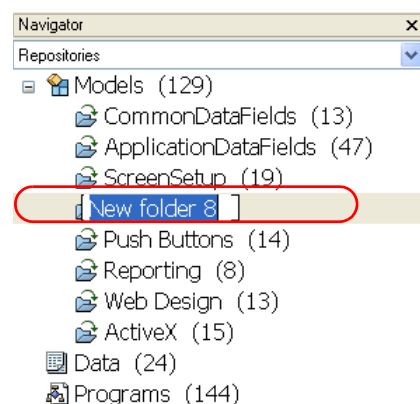


# Chapter 1: Navigation and Workspace

## How do I Organize the Project Objects?

eDeveloper projects are often very large, mission-critical applications, so it's important to keep your work organized. The Studio itself helps you in this, dividing your work into repositories -- the **Model**, **Data**, **Program**, **Help**, **Rights**, **Menu**, and **CRR** repositories. Some of these repositories are shown in the figure on the right.

Within each of these repositories, you can further categorize your entries into folders. In this example we have created seven folders for the **Model** repository, each containing a different type of model.



### To create a folder

1. Position the cursor on the line above the desired folder position ("ScreenSetup" in this example).
2. Press **F4** (or right-click and select **Create Line**, or from the overhead menu **Edit->Create Line**).
3. Type in the name for your folder.

**Hint:** The number to the right of each entry indicates the number of items in each section. For example, in the **Models** section we have 129 total lines. 13 of those lines are in the **CommonDataFields** folder.

When you add a new folder, however, a number will appear to the right of the **New folder** placeholder. This just indicates the sequence of this folder. In this example, our **New folder** is the 9th folder to be added.

### To delete a folder

**Prerequisite:** Before you delete a folder, it needs to be empty. Move the items in the folder to another folder. Also, when you are using version control, be sure you have the repository checked out first.

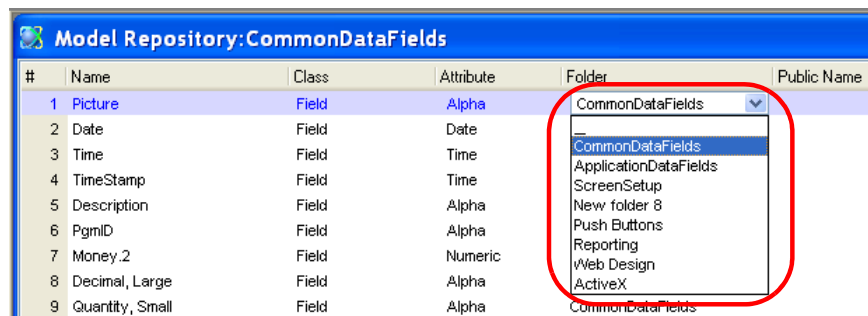
1. Position the cursor on the folder you wish to delete.
2. Press **F3** (or right-click and select **Delete Line**, or from the overhead menu **Edit->Delete Line**).
3. Answer **Yes** to the **Are you sure?** prompt.

### To move a folder

1. Click on the folder you wish to move, so its items are open in the Workspace.
2. Drag the folder to the desired position.
3. Answer **Yes** to the **Are you sure?** prompt.

## Moving Objects into a Folder

There are several ways to move existing objects into a folder. These are listed below.



#	Name	Class	Attribute	Folder	Public Name
1	Picture	Field	Alpha	CommonDataFields	
2	Date	Field	Date		
3	Time	Field	Time		
4	TimeStamp	Field	Time		
5	Description	Field	Alpha		
6	PgmID	Field	Alpha		
7	Money_2	Field	Numeric		
8	Decimal, Large	Field	Alpha		
9	Quantity, Small	Field	Alpha		

**Prerequisite:** When you are using version control, be sure you have the repository checked out first.

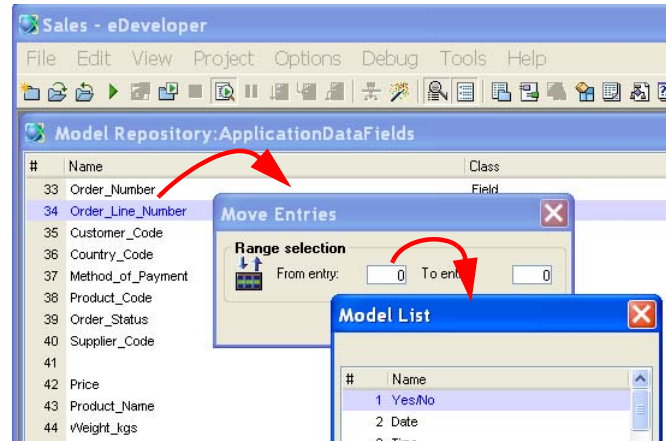
### Moving objects into a folder using the Folder column

1. Select the object you wish to use.
2. In the *Folder* column, click the combo box.
3. Select the desired *folder*.

The object will disappear from the current folder and reappear in the chosen folder.

### Moving objects into a folder using the Move command

1. Go to the folder you want to move the objects into.
2. Press **Ctrl+Shift+M**. The *Move Entries* dialog box will appear.
3. Select the first item in the group of entries you would like to move (you can **zoom** to select from a list).
4. Press **Enter** to just move that one item.
5. Or, enter the last item in the group of entries you would like to move, and press **Enter**.



The objects will appear below the current cursor position, and disappear from their old location.

**Hint:** *This is the easiest way to move large blocks of items into folders.*

**See also:** Chapter 1, “How do I Move an Entry in the Studio?” on page 13.

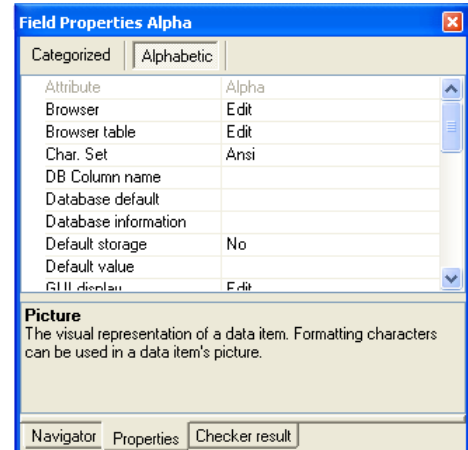
## How do I Separate Palettes?

The Studio palettes are very flexible and can be resized, moved, or docked according to your work style. To save space, the palettes can also be combined as a single window. Sometimes they become combined accidentally; if you move one too closely to another.

### To separate combined palettes

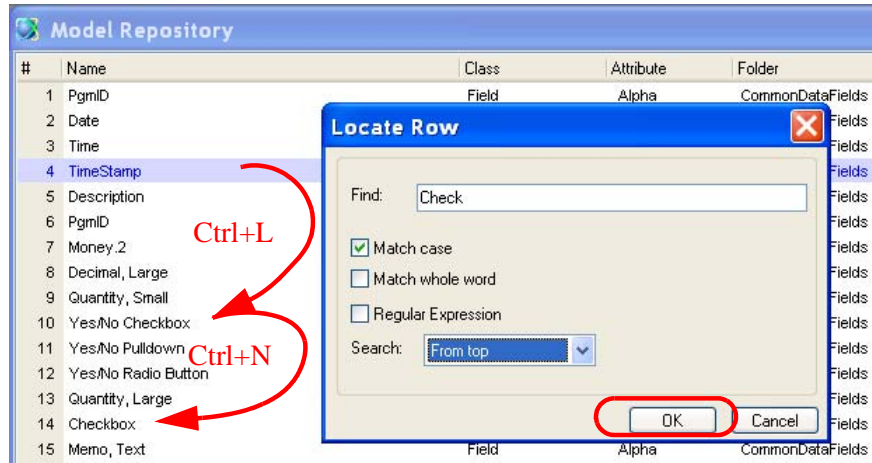
1. Move the focus to the palette.
2. Holding down the **Ctrl** key, drag the title bar off in any direction.
3. Let go of the **Ctrl** key.
4. Repeat the process if you have three combined palettes.

The palettes will now be separated.





## How do I Locate Any Line in the Studio?



Often you will want to find a particular item in a repository, or one particular line in a program. This is easily done using the Locate functionality. Locate has several options, so it is rather flexible. By default, it will find text anywhere within the list, from the current location on down.

For instance, in the example above, **Check** will match **Yes/No Checkbox** and also **Checkbox**. Entering **Alpha** would match any line with the Attribute of **Alpha**. Since we had the *Match case* box checked, it will only match items that have the same capitalization as the text to find.

If you check the *Regular Expression* box, you can enter rather intricate expression masks too. The eDeveloper Help has the syntax details for regular expressions.

### To locate a line

1. Press **Ctrl+L** (or **Edit->Quick Access->Locate Row**).
2. Type in the text you want to find in the **Locate** dialog box.
3. Press **Enter** (or click **OK**).

The cursor will move to the first line that matches your search criteria. Use **Ctrl+N** (or **Edit->Quick Access->Locate Next Row**) to find the next line(s) that match.

**Hint:** The **Locate Line** option works within the current open list. So, if you are working in one open folder, it will only find objects in that folder. If you want to search the entire repository, open the entire repository.

**See also:** Chapter 1, "How do I Quickly Jump to a Line Using Its Number?" on page 6.

## How do I Quickly Jump to a Line Using Its Number?

All the items in eDeveloper have a sequence number. You never need to memorize this number -- you can use the Zoom and Locate options to easily find what you want. However, there are times when you will know the line number and want to go directly to that line.

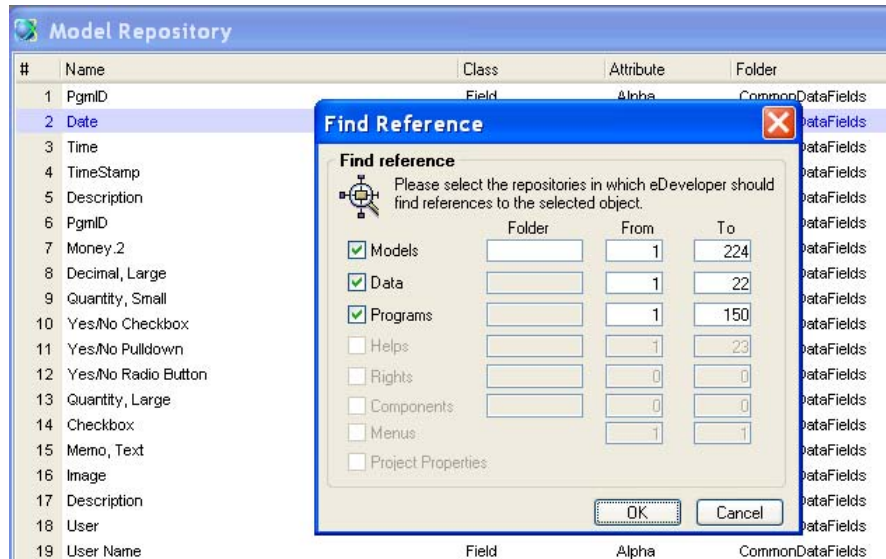
### Jumping to a line

1. Move the focus to the repository list you want.
2. Press **Ctrl+J** (**Edit->Quick Access->Jump to Row**).
3. Enter the line number you want.
4. Press **Enter**.

You will now be positioned on that line.

## How do I Check If an Object Is Being Used and What Other Objects Use It?

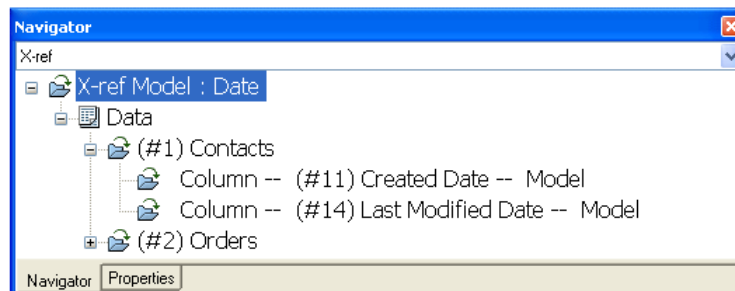
One of the major issues in all programming is “If I change this item, what other items will be affected?”. One simple change can sometimes cause unintended effects. Fortunately, eDeveloper has an excellent cross-referencing system that makes it easy to find (and if needed, change) all objects that refer to any given object.



### Using the cross-reference

1. Move to the item you want to cross-reference.
2. Press **Ctrl+F** (**Edit->Find and Replace->Find Reference**).
3. You will get the **Find Reference** dialog box. By default, eDeveloper checks all references, but if you want, you can narrow the search here.
4. Press **Enter** (or click **OK**).

You will then be presented with a list of all the places that use that object.



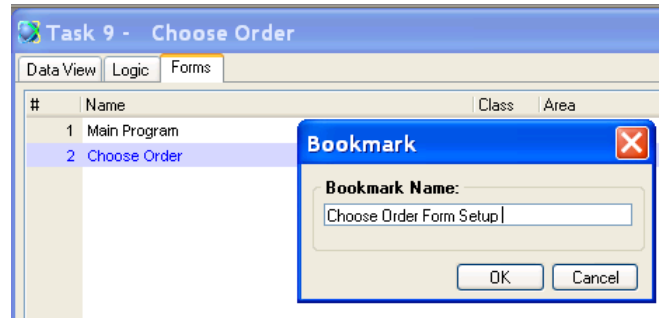
This list is extremely useful. For one thing, you can click on the list entry and go directly to the place that uses the object. This is very nice when you are “fixing” a lot of references to an object.

You can also delete the entries from the cross-reference. So as you fix each item, just delete the entry (**F3**, or **Edit->Delete Line**, just like any other item in eDeveloper).

You can also save the cross-reference to a text file, or print it, using the **Edit->Find and Replace->Save Find Result** and **Print Find Result** options.

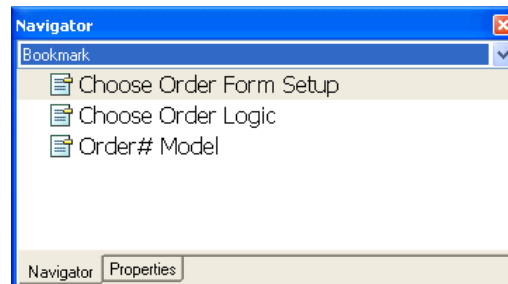
## How do I Bookmark My Current Location for a Quick Return?

You will probably find that while you are working, there are one or more places you need to return to repeatedly while you are programming and testing. Some of these places might be several layers down a program tree. You can mark these places in a series of bookmarks, which will then be in the **Navigator->Bookmark** pane for quick reference.



### To bookmark your current location

1. Press **Ctrl+Shift+B** (or **Options->Bookmark**).
2. The **Bookmark** box will appear. Type in whatever name you want for this bookmark.
3. Press **Enter** (or click **OK**).



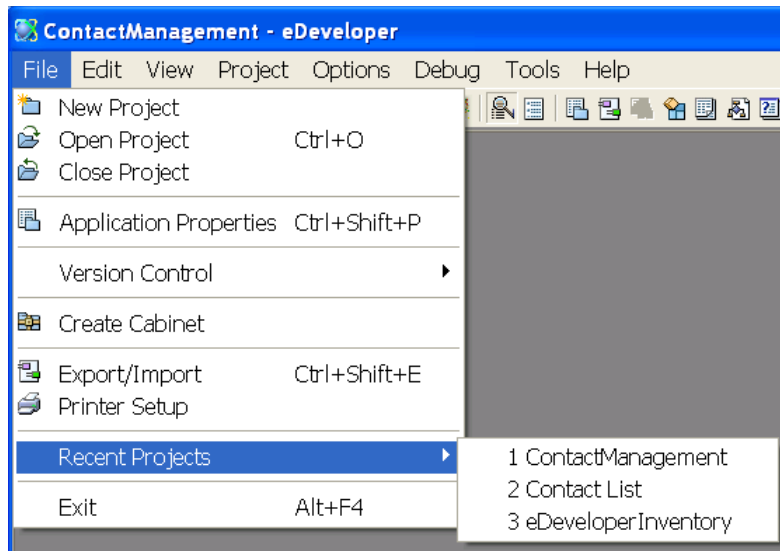
Now, your bookmark will appear in the **Navigator->Bookmark** section. Clicking on this bookmark will cause you to jump immediately to that spot.

You can change the name of the bookmark by using the popup menu **Edit Node** option, and delete the bookmark by using **F3 (Edit->Delete Line)**.

**Hint:** There is a maximum number of bookmarks that can be opened. You can change the maximum number of bookmarks in **Options->Settings->Environment->Preferences->Maximum number of bookmarks**.

## How do I Quickly Reopen a Recently Opened Project?

You may find yourself working with a set of projects. You can open any project in your network using **Ctrl+O** (**File->Open Project**) but sometimes it is simpler to just go to the last few things you were working on.



### Opening a recent project

1. Select **File->Recent Projects**.
2. Click on the project you want to open.

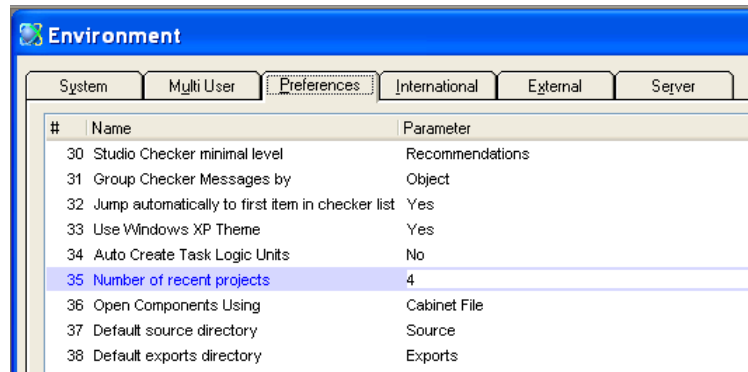
Whatever work you were doing will be saved, and you will jump to the selected project.

**Note:** While you are scrolling through the recent projects, the directory location of the project appears on the status bar. This is useful if you have separate projects with the same or similar names.

**See also:** Chapter 1, “How do I Easily Switch From One Project to Another?” on page 17.  
Chapter 1, “How do I Change the Number of Recently Opened Projects?” on page 11.

## How do I Change the Number of Recently Opened Projects?

You have control over how many projects show up on the “recently opened projects” menu item. The default is 4, but you can enter any number up to 99.



### Setting the number of recently opened projects

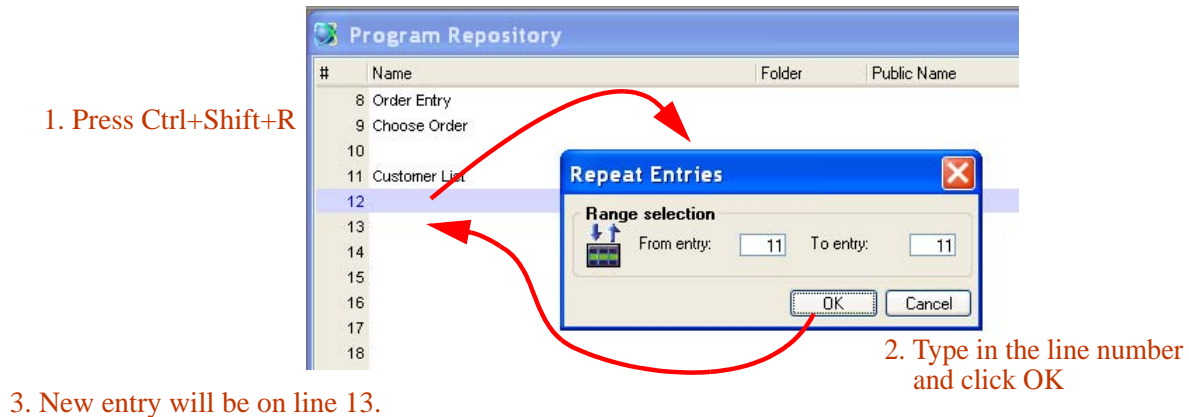
1. Choose **Options->Settings->Environment->Preferences**.
2. Set the line **Number of Recent Projects** to the number you would like.

Setting this to a higher number, results in more items to choose from.

You can also change this by going directly to the Magic.ini file and editing the **NumberOfRecent-Projects** entry.

## How do I Repeat an Entry in the Studio?

When you need to create a new object in eDeveloper, it often saves time to start with a copy of an existing object that is similar to what you need. To obtain a copy of one entry, you need to repeat it.



**Prerequisite:** When you are using version control, be sure you have the repository checked out first.

### Using Repeat

1. Move to the line *above* where you want your new entry.
2. Press **Ctrl+Shift+R** (**Edit->Entries->Repeat entry**)
3. When the dialog box appears, enter the item to repeat. If you know the sequence#, just type it in. Otherwise, zooming on the item# will bring up a list to choose from. This is the **From** item#.
4. The **To** item# defaults to the same number, so if you are only copying one item, just press **Enter** now. Otherwise, zoom from the **To** item# to select a block of entries to repeat.
5. Press **Enter** (or click **OK**).

The selected items will now be copied just below your current position.

**Hint:** Rename the copied entries as soon as you copy them. Otherwise it's easy to get mixed up as to which are the originals and which are the copies. Also, when you are using version control, be sure you have the repository checked out first, or this operation won't work.

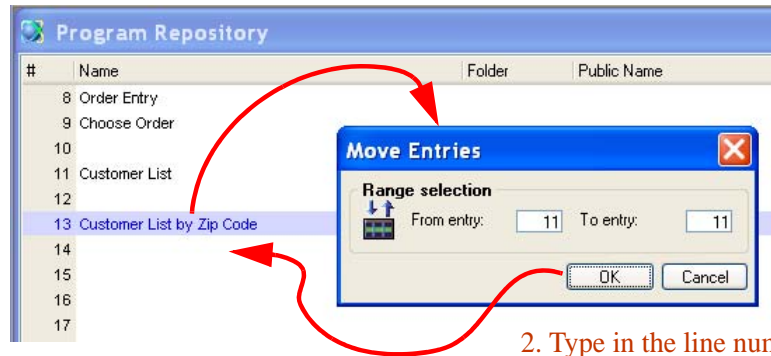
**Note:** In the repositories, the Cut and Paste options only cut and paste text. That is, the name of the item in the repository. They do not copy the object itself.



## How do I Move an Entry in the Studio?

To keep your work organized, you will often need to shuffle entries in the repositories around. For instance, you may want to group items alphabetically, or according to some function. You can move any item in the repositories (except the Main Program).

1. Position cursor on line 13 and press Ctrl+Shift+M



2. Type in the line number (11) and click OK

3. Line 11 will be moved to line 14.

**Prerequisite:** When you are using version control, be sure you have the repository checked out first.

### Moving an entry in the Studio

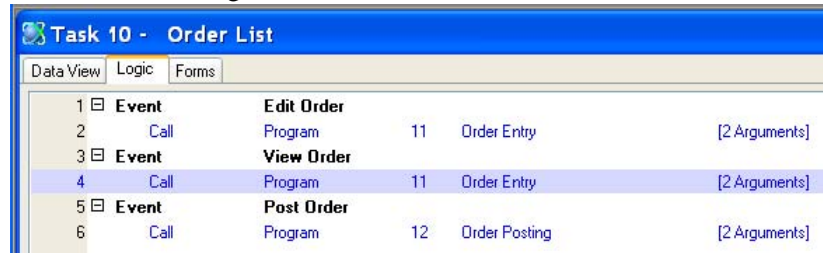
1. Position your cursor on the line *above* where you want the item to go.
2. Press **Ctrl+Shift+M** (Edit->Entries->Move entry).
3. When the dialog box appears, enter the item to repeat. If you know the sequence#, just type it in. Otherwise, zooming on the item# will bring up a list to choose from. This is the **From** item#.
4. The **To** item# defaults to the same number, so if you are only copying one item, just press **Enter** now. Otherwise, zoom from the **To** item# to select a block of entries to move.
5. Press **Enter** (or click **OK**).
6. The selected items will now be moved to just below your current position.

**Hint:** Although you have probably noticed by now that eDeveloper refers to objects by numbers rather than name, it is perfectly safe to move the entries using the **Move Entry** option. eDeveloper will change the references to point to the new position. eDeveloper uses an internal reference number to keep track of the various objects behind the scenes.

There are a few exceptions to this rule, found mainly in older programs, where the programmer did not use the **DSOURCE** or **PROG** literals. If you are working with inherited programs, it can be worth checking for functions such as **DbDel()** with the **Find Text** option and making sure they were programmed correctly.

## How do I Replace an Entry in the Studio with Another Entry?

Items in eDeveloper are often referenced by their sequence number. For instance, you will probably make calls to programs that look something like this:

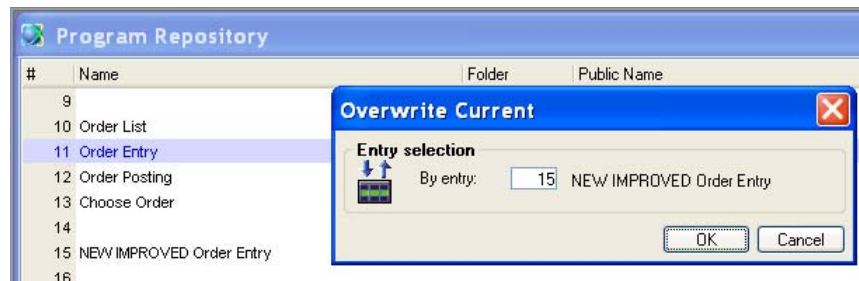


Suppose you *moved* the **Order Entry** program at Line 11 to Line 56. That would be no problem; eDeveloper would automatically change the code above to **Call Program 56 Order Entry**.

But suppose you want to call a *new* version of the **Order Entry** program that you just developed? How do you move it into production?

The answer is, that you need to *replace* Program 11 with your new program.

### Replacing an entry



1. Position the cursor on the line you wish to replace (Line 11, in this case).
2. Press **Ctrl+Shift+O** (**Edit->Entries->Overwrite Entry**).
3. Type in the number of the replacement entry (15 in this case), or **zoom** to select from a list.
4. Press **Enter** (or click **OK**).






Line 11 will now contain the new program (**NEW IMPROVED Order Entry**, in this case). The program at Line 15 will remain unchanged.

**Hint:** This is a good method to use to keep a “quick backup” of your current work. Make a copy of whatever you are working on using **Repeat**, and mark the copies so you don’t get confused. You can keep several working copies this way, and compare them easily, and use **Overwrite** to go back to any version if needed. Of course, for ongoing version control you also have **Rollback**.

## How do I Move Between the Studio Palettes?

While you are working you will usually have one or more palettes open. You can move between them in several ways:

- Click on the palette you want.
- Use **Ctrl+Tab** to move the focus from palette to palette.
- Use the same keys you use to make the palettes appear:
  - **Alt+F1** (**View->Navigator**)
  - **Alt+F2** (**View->Property Sheet**) **Alt+Enter** works also.
  - **Alt+F3** (**View->Checker Result**)

View	Project	Options	Debug
	Navigator	Alt+F1	
	Property Sheet	Alt+F2	
	Checker Result	Alt+F3	
	Comments	Alt+F12	
	Switch Panes	Ctrl+Tab	

**Ctrl+Tab** works a bit differently than the Alt keys. For one thing, Ctrl+Tab treats the combined palettes as one window, while the Alt keys will open a palette even if it is hidden by others. Also, if you happen to have a screen with tabs, such as the *Task Properties* dialog box or the *Task Editor*, the **Ctrl+Tab** will only switch between the tabs, rather than moving to the next palette.

**See also:** Chapter 1, “How do I Separate Palettes?” on page 4.

## How do I Keep the Property Sheet Showing a Single Section at a Time?

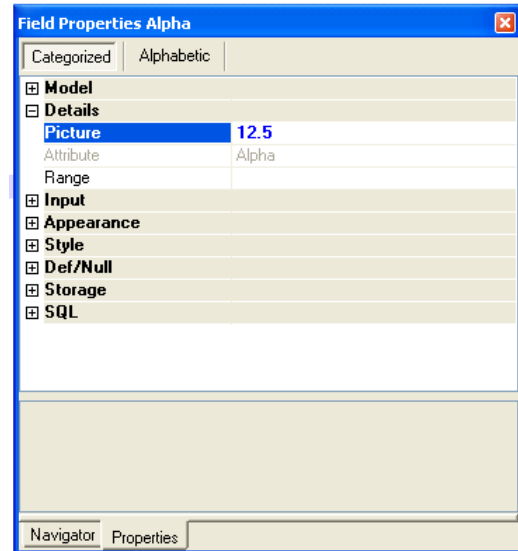
By default, when the property sheet opens, all the sections are open, so you can see all the available properties. However, sometimes you may want to have a more compact property sheet display.

### Changing the property sheet display

1. Close the current project
2. Select **Options->Settings->Environment->Preferences->Single Expand Palettes**
3. Set **Single Expand Palettes** to **Yes**.

Now, whenever you open one section, the previously open section will close. When you click on a new item, all the sections will be closed.

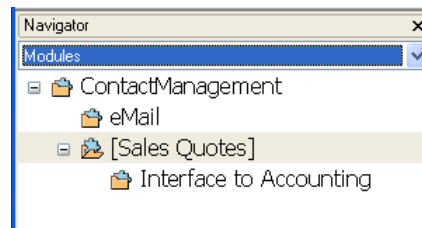
This setting does not change the properties as listed in the **Alphabetic** tab.



## How do I Easily Switch From One Project to Another?

You may have groups of projects that you often work on together, even though they are not necessarily components of each other. You can group these for easy access using the **Modules** entry of the Navigator pane.

In the example below, **Contact Management** has 3 modules. You can tell that **Sales Quotes** is what is currently open, because the folder icon is open and it has brackets around it. Each module represents an eDeveloper project, and they can be located anywhere.



### Switching between projects

1. Move to the Navigator pane.
2. Select **Module**.
3. Move to the module you want to open.
4. Double-click or press **F5 (zoom)**.

The project represented by that module will automatically be opened, and the current project will be closed. Any unsaved changes will be saved.

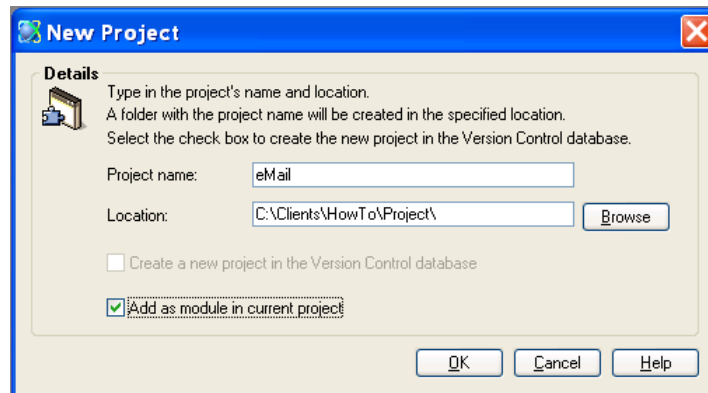
### Adding a Module

**Prerequisite:** If you are using version control, you must have the repository checked out first.

1. Select **Project->Add Module** from the overhead menu.
2. A **File Browser** dialog box will appear. Select the project you want to add.

Your new entry will now appear in the **Modules** section of the Navigator pane.

Also, you can check the **Add as module in current project** box when you create a new project, and it will automatically be placed in the **Modules** section of the current opened project.



**Prerequisite:** When you are using version control, be sure you have the repository checked out first.

## Deleting a Module

**Prerequisite:** Before you delete a module, you have to be located on the module ancestor. You cannot delete the module that is currently open, or the module's ancestor. Also, if you are using version control, you must have the repository checked out first.

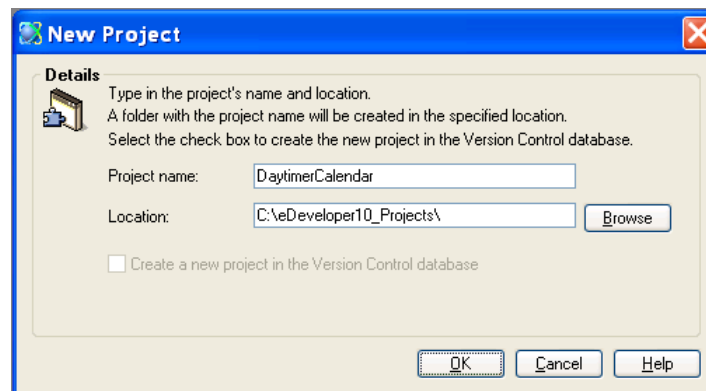
1. Position the cursor on the Module node you want to delete.
2. Press **F3** (**Edit->Delete Line**).
3. Answer **Yes** to the **Confirm Delete** dialog box.

The node will now be deleted from the **Modules** tree. This does not delete the project though; it's still there and you can re-add it any time.

**See also:** Chapter 1, "How do I Quickly Reopen a Recently Opened Project?" on page 10.

## Chapter 2: Projects and Applications

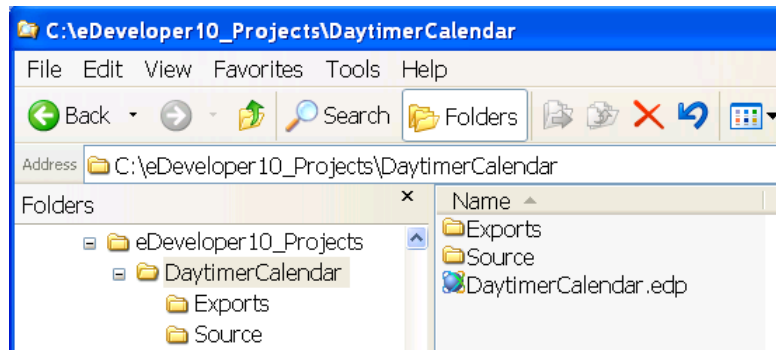
### How do I Create a New Project?



#### Creating a new project

1. Select **File->New Project**
2. Type in a *Project name*.
3. Type in the directory location.
4. Press **OK**.

If you have a project currently open, it will be closed, and a new project created.



When the project is created, it is created in a new subdirectory of the location path you entered. The **Project name** will be the directory name, and it will also be the name of the **eDeveloper project file**, which ends in **.edp**.

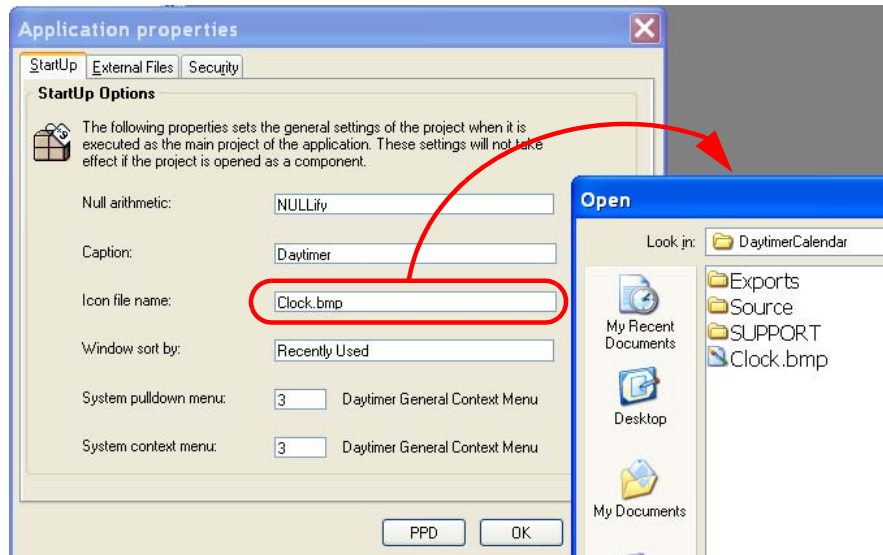
If had a project already open, and you selected **Add as module in current project**, then the project will also be on the module list.

**See also:** Chapter 2, “How do I Open an Existing Project?” on page 30.



## How do I Set the Icon for My Application?

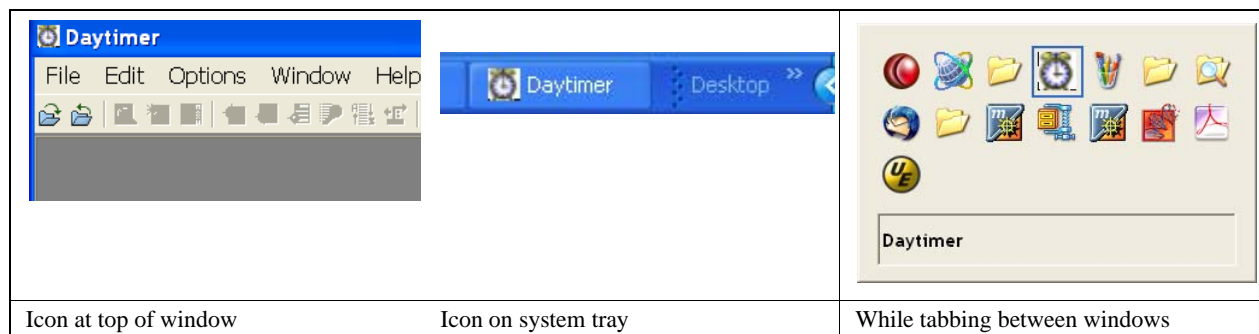
When you are creating your own application, you will probably want your own unique icon. You set this internally to the project, as shown below.



### Setting the icon for your application

1. Select **File->Application Properties (Ctrl+Shift+P)**.
2. Type in the icon file name for the icon you want, or use **zoom** to select the file.
3. Click **OK**.

Now, when you run the application, you will see the icon at the upper left hand side of the window. You will also see it on the taskbar, and when you press **Alt+Tab** to switch between windows.

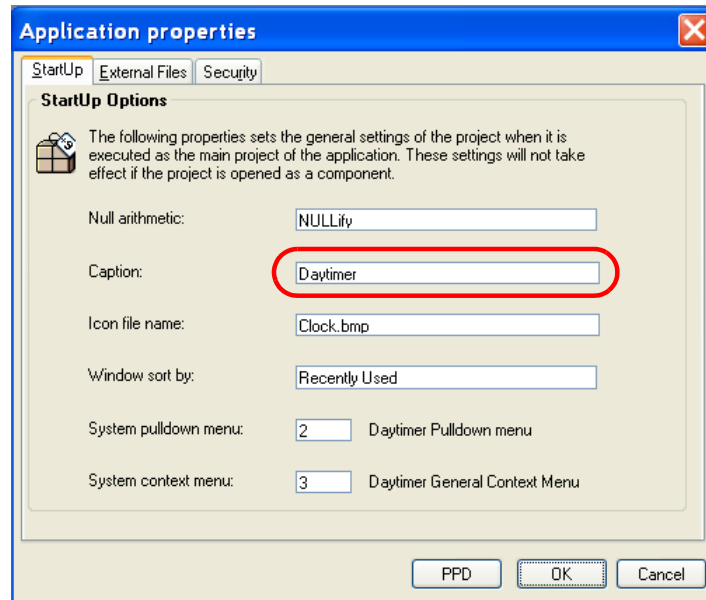


**Hint:** It is best not to use a hard-coded path name for this sort of internal file, since your user will probably have a different setup than you do. The default path will be your working directory (where the project EDP file is), so you can put your image file there, as in the example, or use a relative sub-directory.

**See also:** Chapter 2, “How do I Read and Write Files from/to the Directory of the Project?” on page 27.

## How do I Set the Caption of My Application?

Next to the icon, you would usually want some text describing your application. You can set this in the same area where you set the icon.



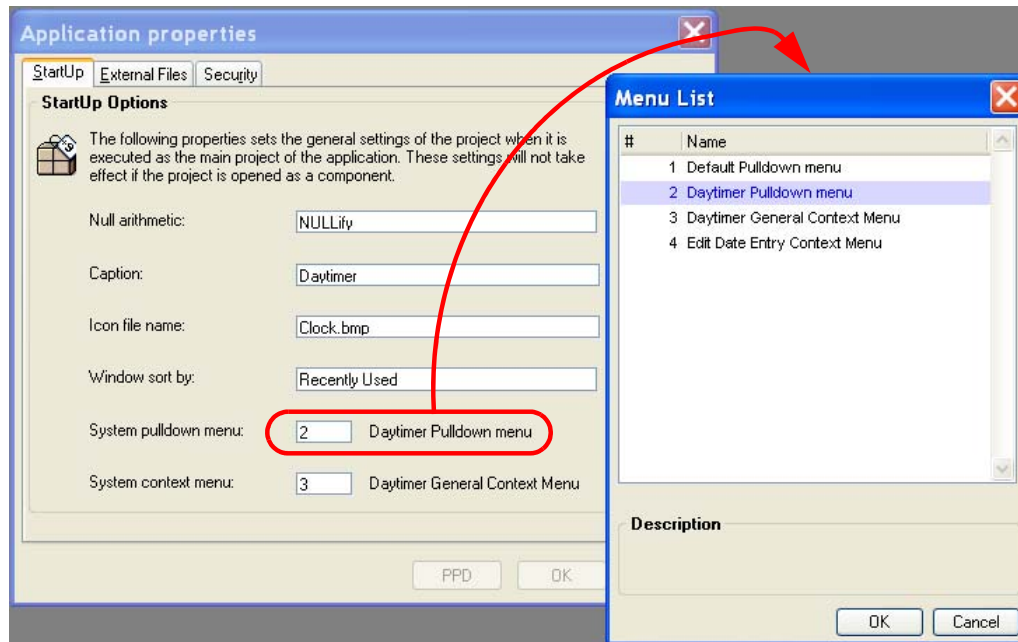
### Setting the caption for your application

1. Select **File->Application Properties (Ctrl+Shift+P)**.
2. Type in the caption name.
3. Click **OK**.

Now, when you run the application, you will see the icon at the upper left hand side of the window. You will also see it on the system tray and while using **Ctrl+Tab** to move between windows.

## How do I Set a Default Context Menu for the Entire Application?

For most online tasks you will want a customized context menu. eDeveloper allows you to set context menus for specific programs, but you will also probably want one default for the entire application.



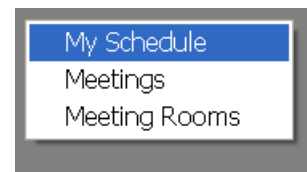
**Prerequisite:** The menu you want to specify must already exist.

### Setting the context menu for your application

1. Select **File->Application Properties (Ctrl+Shift+P)**.
2. Zoom (**F5** or **double-click**) from the System context menu field.
3. Position the cursor on the context menu you want
4. Click **OK**
5. Click **OK**.

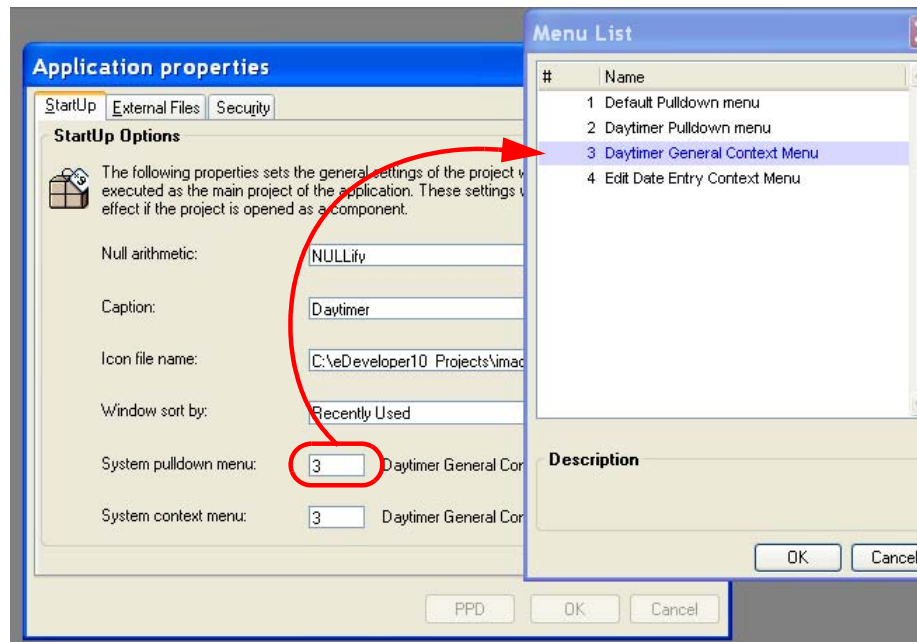
Now, when you run the application, the menu you selected will appear when the user presses the right mouse button.

**See also:** Chapter 5, “How do I Set a Default Context Menu For All Controls of a Form?” on page 152  
Chapter 5, “How do I Set a Context Menu for an Individual Control?” on page 153



## How do I Set or Change the Pulldown Menu for the Application?

eDeveloper comes with a default pulldown menu, but it only has the basic edit commands on it. For a user to run your programs, they will need specific entries that are tied to your application.



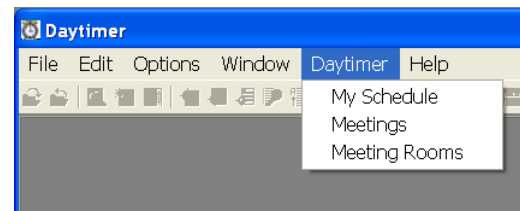
**Prerequisite:** You have already created a menu.

### Setting the pulldown menu for your application

1. Select **File->Application Properties (Ctrl+Shift+P)**.
2. Zoom (**F5** or **double-click**) from the **System pulldown menu** field.
3. Position the cursor on the context menu you want
4. Click **OK**
5. Click **OK**.

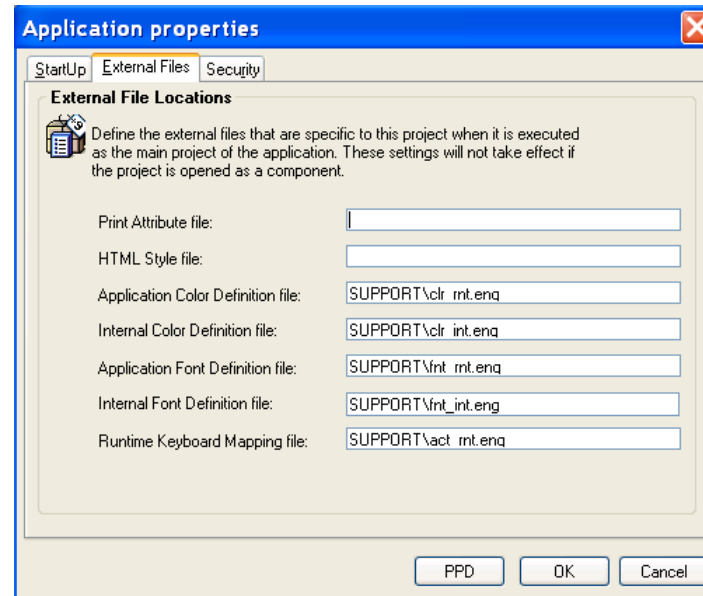
Now, when you run the application, the menu you selected will appear in the overhead menus section.

**Note:** When you set up a menu, there is no distinction between “context” and “pulldown” menus. If you wanted, you could use the same menu for both entries. Typically though, pulldown menus are more complete and structured differently.



## How do I Set the Application to Use Its Own Files for Colors, Fonts, and Keyboard Mapping?

By default, eDeveloper uses the Colors, Fonts, and Keyboard mapping that are installed with eDeveloper. However, you will probably want to customize these features for your particular application. Each application can have its own setup for colors, fonts, and keyboard. Since these are held in text files outside of eDeveloper, they can be changed at runtime and even customized by the user.



**Prerequisite:** You need to copy the font, color, or keyboard file into the desired directory first.

### Setting color, font, and keyboard files for the application

1. Select **File->Application Properties (Ctrl+Shift+P)**.
2. Move to the entry you want to change.
3. Type in the file name.

Now, when you zoom on the file name, you will be able to change the color, font, or keyboard choices for this application.

If you do not specify a full path or logical name, then eDeveloper uses the working directory (**%WorkingDir%**).

**See also:** Chapter 2, “How do I Read and Write Files from/to the Directory of the Project?” on page 27.

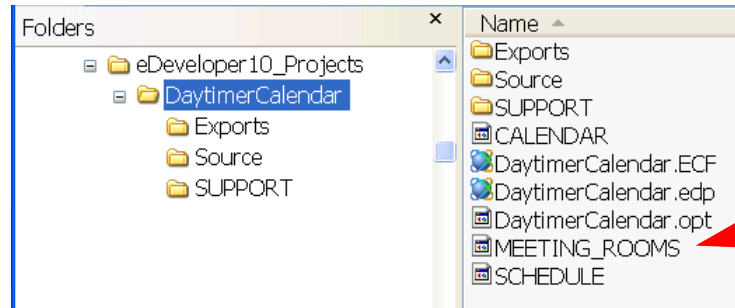
## How do I Read and Write Files from/to the Directory of the Project?

When a project is running, the default directory is the project directory, that is, the location of the **.edp** or **.ecf** file. So, when you create a file in your application and do not give it an explicit path, it will automatically be created in the project directory.

For instance, if we have three db tables in our DayTimer application as shown below:

Data Repository			
#	Name	Data source name	Database
1	Calendar	Calendar	Default Database
2	Schedule	Schedule	Default Database
3	Meeting rooms	Meeting_rooms	Default Database

Then the files will be created in the project directory as shown here:



When you reference other files in your programs, such as i/o files, they work the same way. You can use relative paths to refer to locations beneath the project directory.

If there are multiple projects in the application, then the project directory is determined by the location of the top level project. So, if the DayTimer project called a component which wrote to the project directory, those files would also be in our “DayTimeCalendar” directory.

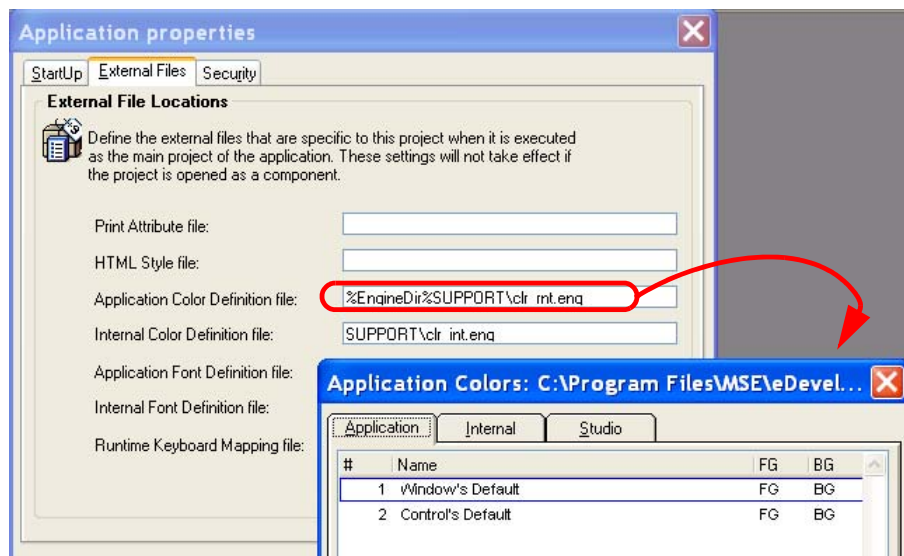
If you want to refer to the project directory specifically, you can use the built-in Logical Name **%Working Dir%**.

## How do I Read and Write Files from/to the eDeveloper Directory?

The *eDeveloper Directory* is the location of the eDeveloper engine, which is wherever eDeveloper was installed. Using the installation defaults, that would be under the Program Files directory on Windows.

eDeveloper differentiates between project-specific files, such as your data or i/o files, and general engine files. For instance, by default, the font, color, and keyboard files shown in the **Options->Settings->Environment->External Files** are all located in the eDeveloper directory.

In your application, you can use the **%EngineDir%** logical name to refer directly to the eDeveloper directory rather than the default working directory. For instance, in our DayTimer project, suppose we wanted to use the color files installed with eDeveloper.

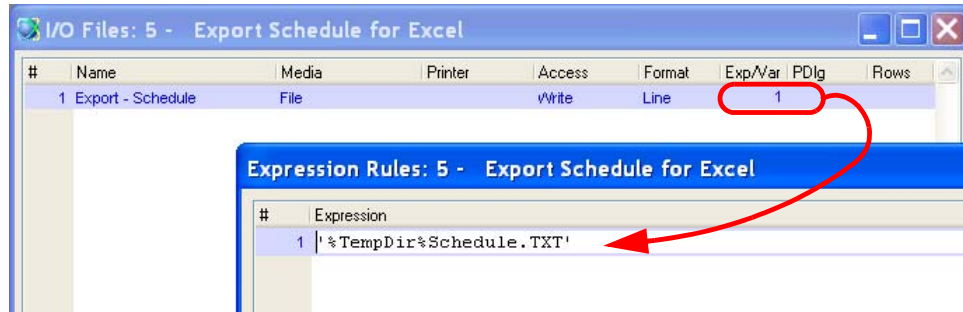


By entering **%EngineDir%** in front of the relative path, the clr\_int.eng file used is the one in C:\Program Files, our default installation directory.



## How do I Read and Write Files from/to the System's Temporary Directory?

The Windows operating system has a temporary directory set up by default, which is used by many applications for writing “scratch” files, generally under **C:\Documents and Settings**. This is a good place to write intermediate files that are intended for, say, importing into another product. You can easily access this directory using the **%TempDir%** logical name.



In this example, we are creating a text file which will be read into Excel. If we did not specify a path name, the temporary file would be created in our working directory. By adding the internally-defined logical name **%TempDir%**, we force eDeveloper to use the system temporary directory instead.

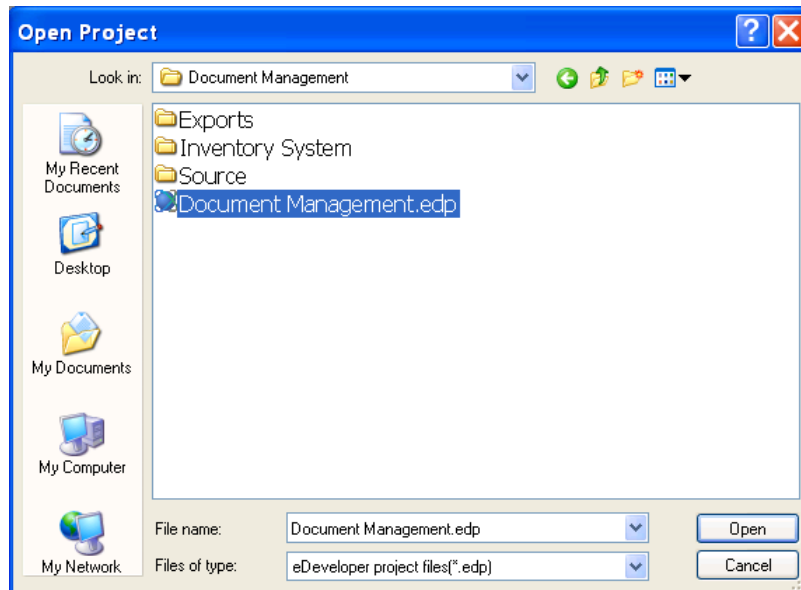
## How do I Open an Existing Project?

Your application may consist of one or many projects, as some logic may be encapsulated in components or even made to run on other servers. You can move between projects easily in the eDeveloper Studio, or open projects from within Windows. Listed below are three basic ways to open an existing project.

### Using the Open Project dialog box

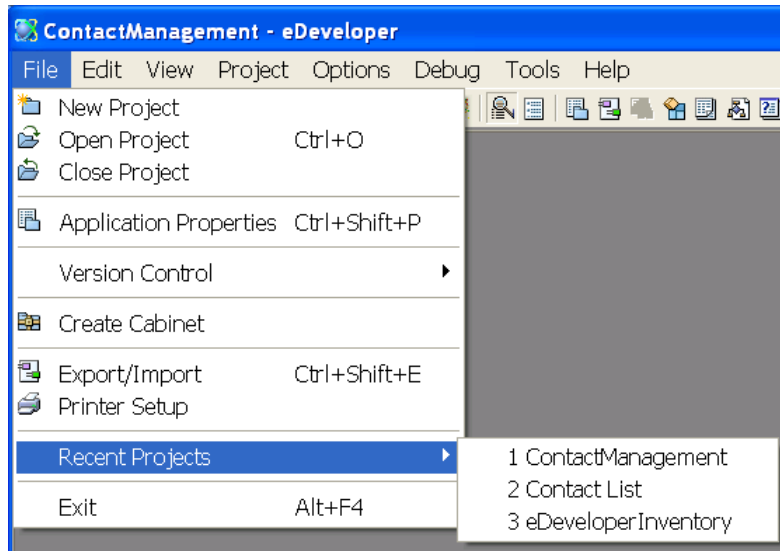
1. Select **File->Open Project** from the overhead menu, or press the **Ctrl+O**.
2. A Windows file selection dialog box will appear. Choose the **.edp** file for the desired project.
3. Press **Open**.

The new project will now be open.



## Using Recent Projects

You do not need to close the current project; it will automatically be closed and saved. You may find yourself working with a set of projects. You can open any project in your network using **Ctrl+O** (**File->Open Project**) but sometimes it is simpler to just go to the last few things you were working on.

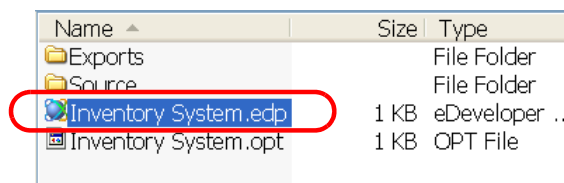


1. Select **File->Recent Projects**
2. Find the project you want. Note that if you hover over an entry with the mouse, you will see the project location, which can be helpful in the case of similarly-named projects.
3. Click on the project you want to open.

Whatever work you were doing will be saved, and you will jump into the selected project.

**See also:** Chapter 1, “How do I Change the Number of Recently Opened Projects?” on page 11.

## Directly activating the .edp



**Just click on it**

1. Go to the EDP file in Windows Explorer
2. Click in the EDP file.

The project will open. This method is a little different in that any other project you happen to have open will remain open, and you will get a new eDeveloper Studio session. You can also use this method by creating a windows shortcut to the EDP file.

This method works because Windows associates the suffix **.edp** with eDeveloper.

**See also:** Chapter 7, “How do I Create a Shortcut for my Application?” on page 192.

## How do I Transfer Objects From One Project to Another?

**Prerequisite:** If the objects involved have no dependencies, then copying them is no problem. However, if one object is dependent on another, the project files must be identical in terms of the dependencies. For instance, suppose you are copying program 21 from **project A** to **project B**. Program 21 uses models 3, 18, and 24, and also calls program 46.

When you import program 21 to **project B** then, models 3, 18, and 24 must be identical in the two projects, and program 46 must be the same also.

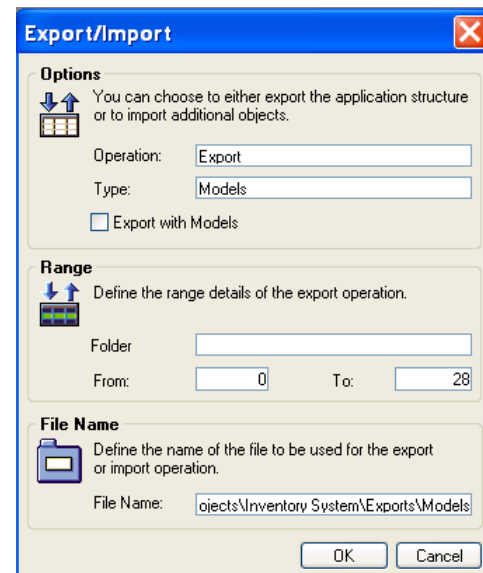
In this example, we will be exporting the models from one project, to use in starting another project.

### Exporting Objects

1. Press **File->Export (Ctrl+Shift+E)**.
2. For Operation, select **Export**.
3. For Type, select the repository you want to export (models, Data sources, programs, help screens, rights, menus, components, Application properties, or the entire project).
4. Select one folder, if you want, or a from/to range, to limit how many objects get exported. You can zoom from the from/to fields to select from a list. By default, all objects of the type you chose will be exported.
5. Enter a file name, if you want. By default, the file will be created in the **Exports** subdirectory of the working directory.
6. Click OK.

A new file will be created. In this example, the file **Models.xml** was created.

Now let's import those models into our new empty project.



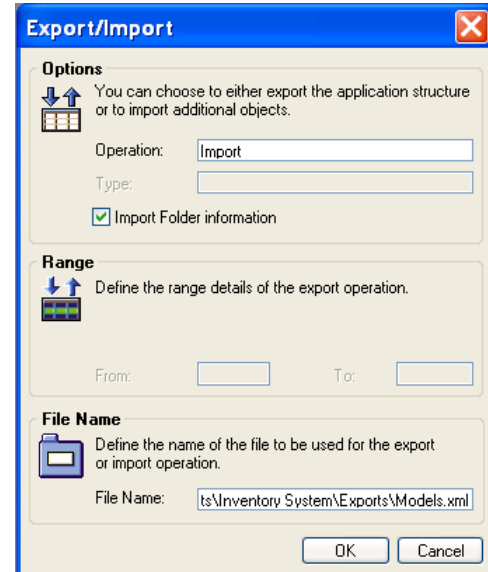
## Importing Objects

1. Press **File->Export** (**Ctrl+Shift+E**).
2. For Operation, select Import.
3. Enter the file name with path of the **XML** file you just exported.
4. Click **OK**.

The objects you exported will now be located in your project. They will import sequentially underneath whatever other items you have in that object repository.

**Hint:** *Import/export is generally used to move programs between two versions of the same project, to import items into an empty project (such as generic models), or to copy simple generic routines. In general, if you need to share objects between several projects, it is better to use a component. Components are truly reusable objects that are easily shared.*

**See also:** Chapter 16, “How do I Reuse eDeveloper Objects Across Projects?” on page 411.



## Chapter 3: Models

---

### How do I Define Reusable Interface Objects?

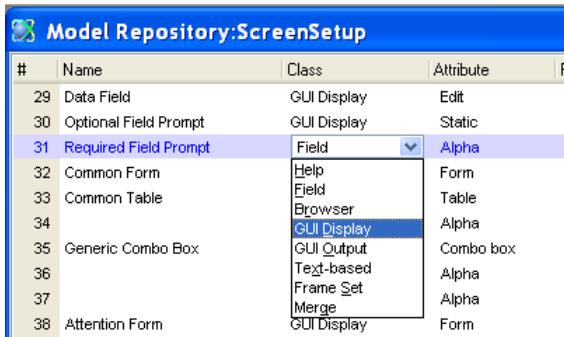
One of the challenges of today's programming is maintaining a consistent look and feel across hundreds of open windows, browsers, and reports. Gone are the days of simple text green screens; today you have a choice of hundreds of fonts, sizes, and colors. Setting these choices for each object is horribly time-consuming, and keeping them consistent or making changes is next to impossible, if they are set on an object-by-object basis.

Fortunately, eDeveloper makes this sort of thing incredibly easy. eDeveloper contains a robust system of models, which can be used to define the look and feel of every interface object, including text fields, radio buttons, tables, table columns, print formats, Web browser screens, and Windows windows.

When these are set up as models, using them is easy: you just choose the model you want while designing your form. Changing them is even easier: you change the model, and all items that use that model automatically change.

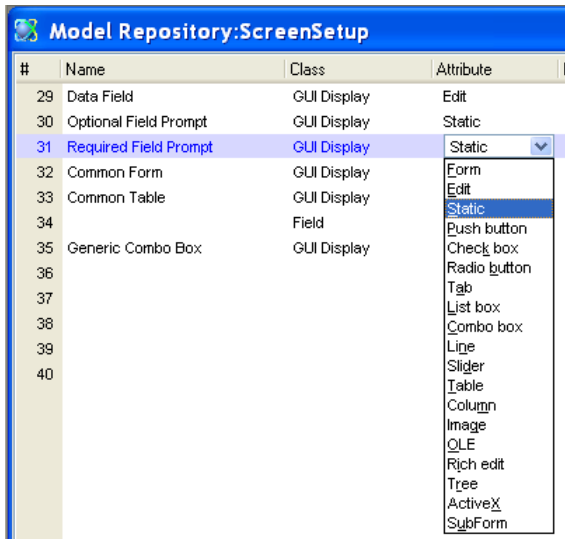
Here we will give an example of setting up a simple model for a field prompt.

Creating a control model

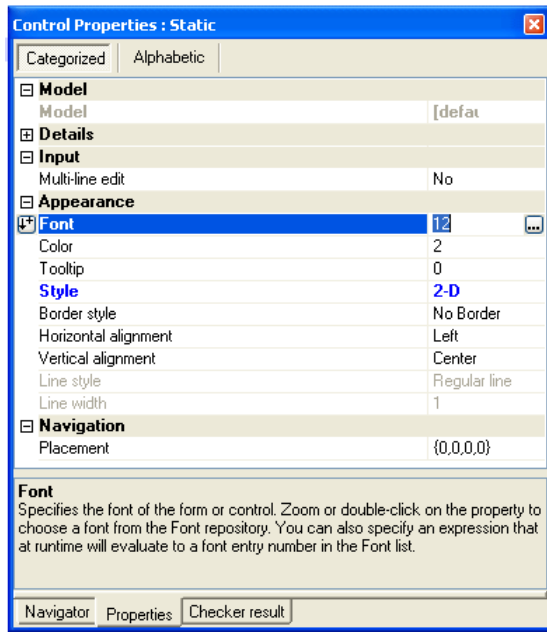


1. Go to the Model repository (**Shift+F1**), and move to the desired location.
2. Press **F4** (**Edit->Create Line**).
3. Type in the *name* of the model (here it is “Required Field Prompt”).
4. Select the *class* you need:
  - **GUI Display:** for any online screen control, which is what we are using in our example.
  - **GUI Output:** for most formatted reports
  - **Text-based:** for text-only output going to another product or older printer.
  - **Browser:** for Browser controls.





5. Select the attribute. The list of attributes changes depending on the class of the model. Here, you can see the standard list of GUI display (online screen) items. For our example, we select *Static*, because we are dealing with static text



6. Finally, you need to change the properties. The list of properties will change depending on the class and attribute.

7. In this case, we are creating a prompt that will be bolder than other field prompts, so we use a different font. Also we want the style to be 2-D, not the default 3-D.

After you are done creating the model, you can use it when creating controls to automatically format them or attach your model to a data type to automatically format it wherever it is used (*Chapter 3, “How do I Unify and Standardize the Project’s Data Fields and Visual Controls?” on page 42*).

## How do I Define Reusable Data Objects?

One of the challenges in creating large applications is keeping data fields consistent. For example, if you have an address field, you would not want it defined as 30 characters in one program, and as 40 characters in another program. If you have a record ID, and it is 10 characters, you do not want to pass it to a program that has the record ID defined as 8 characters.

You also want to have the valid values of fields defined consistently, and to be easily visible to the programmer. For instance, if you have a status code, how does the programmer know what each of the code values stand for?

And perhaps most importantly, how do you ensure that if the length or valid values of a field change, all the tables and programs that use them also change?

eDeveloper makes this easy, using models. If a piece of data is attached to a model, that model can define the field length and valid values of that data. Not only that, the model can determine the actual DBMS storage format, encapsulated selection programs, help screens, prompts, tooltips, and more. The settings are easily accessed by the programmer, so they help document the data type.

The model settings are also easily changed, so if you need to add a new status code, changing the status code model will automatically change all the status codes in tables and programs. What is more, you can also find all those status codes by using the **Find Reference (Ctrl+F)** facility to make sure your change won't cause a problem for the logic.

Models that define data are created with the class of *field*. They are easy to create and even easier to use.

## Creating a field model

#	Name	Class	Attribute
29	Record#	Field	Numeric
30	Status code	Field	Alpha
31	Customer Code	Field	Alpha

1. Go to the Model repository (**Shift+F1**), and move to the desired location.
2. Press **F4** (**Edit->Create Line**).
3. Type in the *name* of the model (here it is “Status Code”).
4. Select the *class* of *Field*.
5. Select the *attribute* you need. For a field model, this would be one of your basic data types: alpha, numeric, date, time, blob, ActiveX, etc.

Field Properties Alpha	
Categorized   Alphabetic	
Model [default]	
<b>Details</b>	
Picture	U
Attribute	Alpha
Range	New, Processing, Shipped, Void
<b>Input</b>	
Select program	0
Select mode	Before
<b>Appearance</b>	
Help screen	0
Tooltip	0
Help prompt	0
<b>Style</b>	
Browser	Edit
Browser table	Edit
GUI display	Radio button
GUI display table	Combo box
GUI output	Edit
GUI output table	Edit
Text Based	Edit
<b>Def/Null</b>	
Null allowed	Yes
Null value	
Null display	
Null default	No
Default value	
Database default	
<b>Storage</b>	
Char. Set	Ansi
Default storage	No
Modifiable	Yes
<b>SQL</b>	
Database information	
DB Column name	OrderStatus
Type	
User type	

6. Now go to the model *properties* pane (**Alt+Enter**) and set the properties you need. There are a lot of choices here, you can read about them more in the *eDeveloper 10 Reference Guide*.

**Details:** In this section, you can specify the length and format of the field in its picture. You can also specify the valid values of the field, either as a range (i.e. A-Z) or as discrete values. Here we have a 1-character uppercase field, with 4 possible values.

**Input:** You can specify a program that will appear when the user presses **F5** or double-clicks (zoom), to aid in selecting a value.

**Appearance:** Here you can attach a help screen, tooltip, or prompt to show on the prompt line.

**Style:** This section specifies how the data will appear depending on where it is displayed. In this case, our status code will appear as a radio button on most GUI display screens, but will show up as a combo box if it is on a GUI table.

**Def/Null:** Determines how nulls are used, and if the field has a default value.

**Storage/SQL:** Here you can specify how the data will be stored in the DBMS.

As you can see, the field models give you a lot of control over how data will be formatted and used in your application. Once the field model is created, you can use it for any instance of this type of data. In this case, we would use it for the status code as it exists in records, as it is passed as a parameter, and in temporary variables in programs.

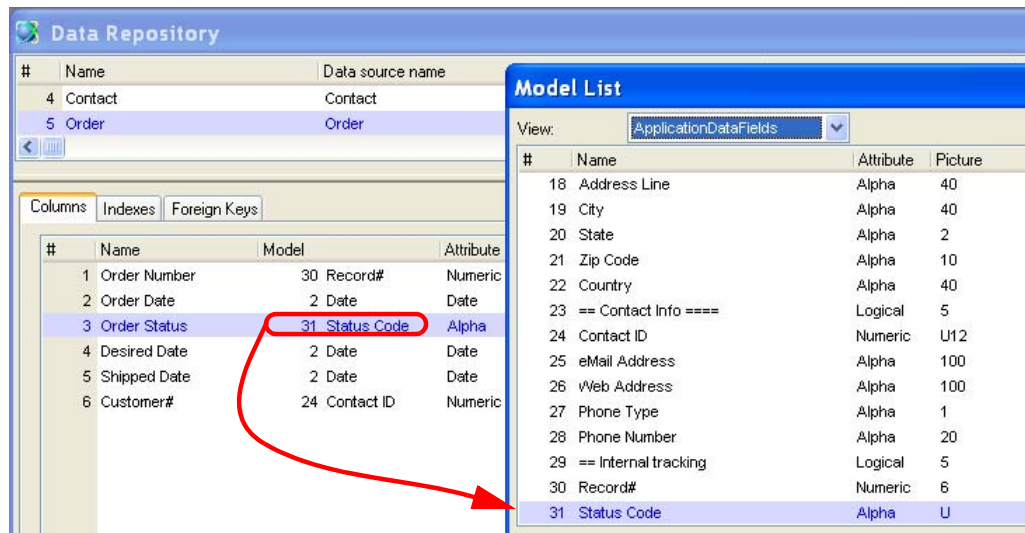
**See also:** Chapter 3, “How do I Unify and Standardize the Project’s Data Fields and Visual Controls?” on page 42.  
Chapter 3, “How do I Define a Data Source Column Based on a Model?” on page 41.

## How do I Define a Data Source Column Based on a Model?

It is very good to have your data defined in a consistent way. Models help enforce consistency. Once you have your data models in place, you can use them to ensure that the same data type is the same in all your data sources.

### Defining a data source column based on a model

**Prerequisite:** The field model must already exist in the Model repository.



1. Go to the *Column* tab of the data source.
2. Use **F4 (Edit->Create Line)** to open up a line for your new column.
3. Because you are using a model, you can leave the name field blank if you want, and the name will be automatically inherited from the model.
4. In the *Model* field, zoom (**F5** or double click) to select the model you want to use. In this instance, we are using a “Status Code” model.

Now, when you look at the new column’s property pane, you will see that it has inherited the properties from the “Status Code” model. This means you don’t have to specify anything further here for your standard status code. If the properties of “Status Code” change, then the properties for your data column will change as well.

You can, however, override the properties if you wish. If you override the properties here, that will break the inheritance and those properties will not change if the model changes.

**See also:** Chapter 3, “How do I Define Reusable Data Objects?” on page 38.

## How do I Unify and Standardize the Project's Data Fields and Visual Controls?

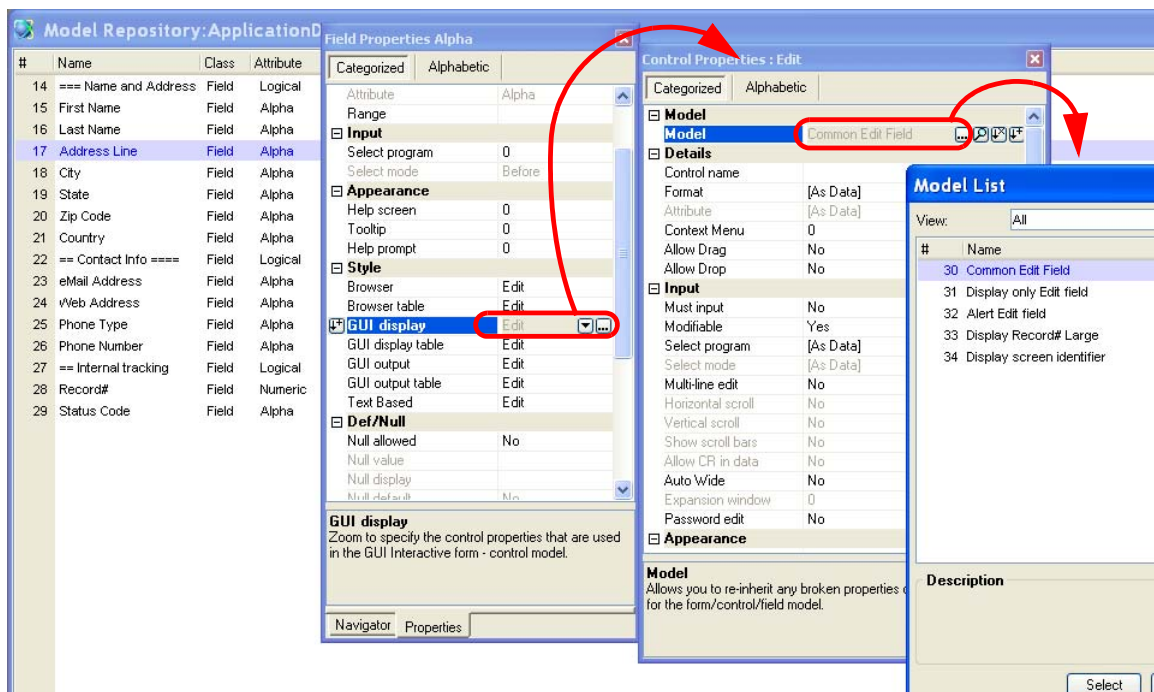
Within any programming project, you have two types of data issues:

1. The data itself: how long each field is, how it is stored, what data type it is, and what are the valid values. In eDeveloper, the field class models encapsulate this information in a *field* model.
2. How the data looks to the user, and how the user interacts with it.
3. eDeveloper, the **GUI Display**, **GUI Output**, **Browser** and *Text-based* models encapsulate the look and feel in a *control* model.

When you put a data field on a form, you can specify all the details of the control, or choose from your set of control models. However, you can also specify the control model as part of the data model, so the two are linked at the most basic level. This will save you a lot of time while programming, and standardize which types of data are represented by which visual control.

### Specifying a control for a data model

**Prerequisite:** The control model must already exist.



1. Position the cursor on the field model.
2. Go to the field properties for the model (**Alt+Enter**).
3. Go to the *Style* you want to change. For instance, if you want to change how fields look when displayed on a Windows screen, go to GUI display.

4. A new properties pane will open up, titled *Control Properties*. This pane allows you to specify details of the control. If you want, you can specify the details here manually. Usually though, it makes more sense to specify a model for the control properties.
5. From the *model* field in *Control Properties*, zoom (F5 or double click) to select the control model you want to use.

Now, your data model is connected to a control model. Whenever that data model is used to define a piece of data, it will automatically have the visual properties specified in the control model.

**See also:** Chapter 3, “How do I Define Reusable Data Objects?” on page 38.

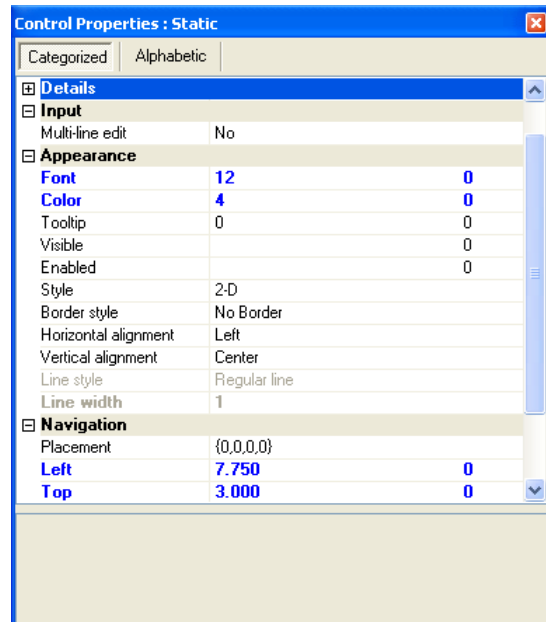
## How do I Prevent an Object from Being Affected by Any Change of its Model's Properties?

Once you have a model defined, any changes to that model will automatically be reflected in all objects that use the model. This is extremely useful. For instance, if we wanted to change our “required fields” standard so that required fields were all in italic, we could change the model and all our screens would be changed instantly.

However, suppose you do not want a particular screen to reflect future changes to model. In that case, you need to *break* the inheritance. In eDeveloper you can tell if inheritance is broken because that property will be shown in bold blue font.

**Note:** You can customize this and the other colors and fonts used in the Studio.

- To change the color, go to **Settings->Options->Colors->Studio**, and change colors 44 and 45.
- To change the font, **Settings->Options->Fonts->Studio**, and change fonts 34 and 35.





In the example on the right, “Font” and “Color” are both in bold blue font. Both of them have inheritance broken.

There are two ways you can break inheritance: manually and automatically.

### Manually breaking inheritance

You can turn inheritance off (and back on) by clicking on the icon to the left of the property. This icon only



appears when you are sitting on the property in question. If the icon looks like , then clicking on it will break the inheritance, and turn the property blue. If the icon looks like , then clicking on it will reset the inheritance and the color will go back to black.

### Automatically breaking inheritance

Whenever you change a property on an object, the inheritance for that property is automatically broken and the property will turn blue. If the inherited color for an object is 2, for instance, and you change it to 4, then you have broken the inheritance for color.



**See also:**

**See also:** Chapter 3, “How do I Set a Broken Property to Inherit its Value?” on page 46

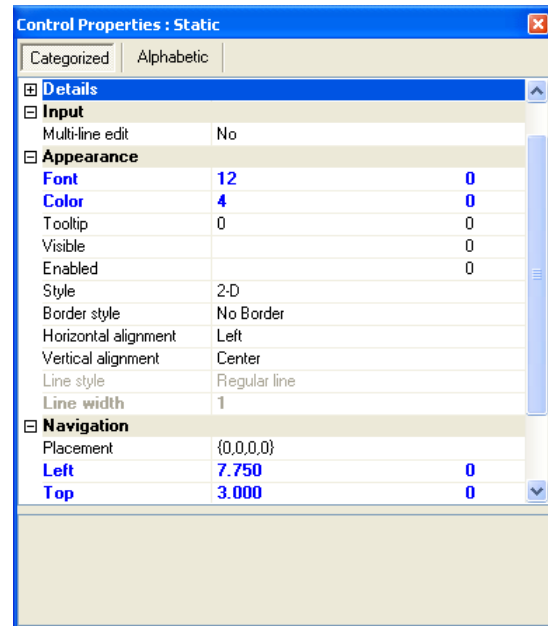
## How do I Set a Broken Property to Inherit its Value?

Once you have a model defined, any changes to that model will automatically be reflected in all objects that use the model. This is extremely useful. For instance, if we wanted to change our “required fields” standard so that required fields were all in italic, we could change the model and all our screens would be changed instantly.

However, suppose you do not want a particular screen to reflect future changes to model. In that case, you need to *break* the inheritance. In eDeveloper you can tell if inheritance is broken because that property will be shown in bold blue font.

**Note:** You can customize this and the other colors and fonts used in the Studio.

- To change the color, go to **Settings->Options->Colors->Studio**, and change colors 44 and 45.
- To change the font, **Settings->Options->Fonts->Studio**, and change fonts 34 and 35.





In the example on the right, “Font” and “Color” are both in bold blue font. Both of them have inheritance broken.

There are two ways you can break inheritance: manually and automatically.

### Manually breaking inheritance

You can turn inheritance off (and back on) by clicking on the icon to the left of the property. This icon only



appears when you are sitting on the property in question. If the icon looks like , then clicking on it will break the inheritance, and turn the property blue. If the icon looks like , then clicking on it will reset the inheritance and the color will go back to black.

## How do I Change the Class of a Model?

Once a model is created, you cannot change the class of that model. After you fill in the initial values for the model, the class becomes locked as soon as you move the cursor off the model line onto another model. If this happens on a newly-created model that has the wrong class for some reason, just use **F3 (Edit->Delete)** to delete the model and start over.

However, if you are not sure if the model is being referenced or not, be sure to check that it isn't used, using **Ctrl+F (Edit->Find and Replace->Find Reference)** before deleting it.

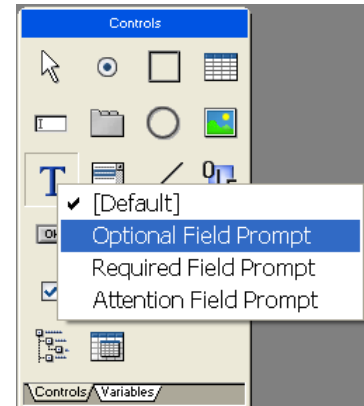
**See also:** Chapter 1, "Using the cross-reference" on page 7

## How do I Automatically Drop Form Controls Using a Specific Control Model?

eDeveloper has a control palette that allows you to put a control on your form by simply clicking on it, which will give you the default properties for that control. You can, however, select the control along with the model you wish to use for that control.

### Selecting the control and model from the palette

1. Move to the control palette.
2. **Right-click** on the control you wish to choose.
3. A list of applicable models will appear. Move to the model you want to use, and **left-click** the mouse.
4. The cursor icon will now change to look like the control you chose. Move it over to your form, and **left-click** to drop the control where you want it.



## How do I Export a Program or Table While Keeping Their Models?

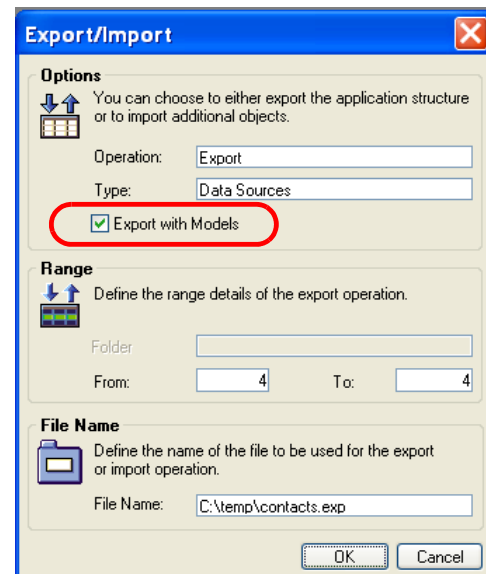
Columns				
Indexes				
Foreign Keys				
#	Name	Model	Attribute	Picture
1	Contact ID	29 Record#	Numeric	5
2	Contact First Name	16 First Name	Alpha	40
3	Contact Last Name	17 Last Name	Alpha	40
4	Address 1	18 Address Line	Alpha	40
5	Address 2	18 Address Line	Alpha	40
6	City	19 City	Alpha	40
7	State	20 State	Alpha	2
8	Zip	21 Zip Code	Alpha	10
9	Phone Type 1	26 Phone Type	Alpha	1
10	Phone Number 1	27 Phone Number	Alpha	20
11	Phone Type 2	26 Phone Type	Alpha	1
12	Phone Number 2	27 Phone Number	Alpha	20
13	Phone Type 3	26 Phone Type	Alpha	1
14	Phone Number 3	27 Phone Number	Alpha	20
15	Notes	0	Blob	

Because models are so useful, your tables and programs are likely to make extensive use of them. However, this makes exporting and importing tables and programs a bit more complicated, because the models must exist in the project you are importing into.

An easy way to handle this is to use the *Export with Models* option on the Export/Import dialog. When you use this option, the models that apply to whatever you are exporting will be exported along with your data sources or programs.

### Exporting with models

1. Export as you usually would, but be sure to check the **Export with Models** check box.
2. When you import the file, the models will be appended at the end of any models currently in the Model repository. The references to those models will also be updated, so the data sources and programs will still be correct.



In our example, we imported the table shown above into a new project which already had 30 model entries. The result, as you can see below, is the same table, even though the model sequence numbers are different. A total of 10 models was added to the new project.

ColumnsIndexesForeign Keys				
#	Name	Model	Attribute	Picture
1	Contact ID	39 Record#	Numeric	5
2	Contact First Name	31 First Name	Alpha	40
3	Contact Last Name	32 Last Name	Alpha	40
4	Address 1	33 Address Line	Alpha	40
5	Address 2	33 Address Line	Alpha	40
6	City	34 City	Alpha	40
7	State	35 State	Alpha	2
8	Zip	36 Zip Code	Alpha	10
9	Phone Type 1	37 Phone Type	Alpha	1
10	Phone Number 1	38 Phone Number	Alpha	20
11	Phone Type 2	37 Phone Type	Alpha	1
12	Phone Number 2	38 Phone Number	Alpha	20
13	Phone Type 3	37 Phone Type	Alpha	1
14	Phone Number 3	38 Phone Number	Alpha	20
15	Notes	0	Blob	

**See also:** Chapter 2, “How do I Transfer Objects From One Project to Another?” on page 33.

## How do I Share a Collection of Models with Several Projects?

You will probably find that you are using the same models over and over again in different projects. You can export them from one project into another, but it is easier to maintain a library of commonly used models that you share between projects. This also allows you to change some of the functionality of a model without reprogramming. For instance, you might want to add a calendar as a popup program for a date field, or you might want to have several components with different money definitions depending on the country your application is running in.

Model Repository: CommonDataFields					
#	Name	Class	Attribute	Folder	Public Name
1	Yes/No	Field	Logical	CommonDataFields	YesNo
2	Date	Field	Date	CommonDataFields	DateDefault
3	Time	Field	Time	CommonDataFields	TimeDefault
4	TimeStamp	Field	Time	CommonDataFields	TimeStamp
5	Description	Field	Alpha	CommonDataFields	Description
6	PgmID	Field	Alpha	CommonDataFields	PgmID
7	Money.2	Field	Numeric	CommonDataFields	Money.2
8	Decimal, Large	Field	Alpha	CommonDataFields	DecimalLarge
9	Quantity, Small	Field	Alpha	CommonDataFields	QtySmall
10	Quantity, Large	Field	Alpha	CommonDataFields	QtyLarge
11	Checkbox	Field	Logical	CommonDataFields	Checkbox
12	Memo, Alpha	Field	Alpha	CommonDataFields	MemoAlpha
13	Memo, RTF	Field	Blob	CommonDataFields	MemoRTF
14	Image	Field	Blob	CommonDataFields	Image

### Sharing models as components

1. Define your models as you usually would.
2. Give each model you want to share a unique public name.
3. Proceed to create and install your component as you would for any component.

When your component is attached to your project, you will see the component models wherever you can select a model from a list. You can use component models in exactly the same ways as you would the models in the project's Model repository.

**See also:** Chapter 16, "How do I Reuse eDeveloper Objects Across Projects?" on page 411.





# Chapter 4: The eDeveloper Engine

---

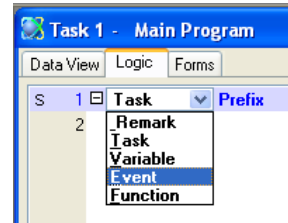
## How do I Define Application Level Events?

In eDeveloper, you can define events to happen in any task. However, sometimes you will want an event that can be triggered from any task, or when no task is running. Examples might be a messaging system that checks for messages every few minutes, or a global error handler that gives a message for any unhandled database errors, or a pop-up program that is set to a hot key, which runs on top of any current task.

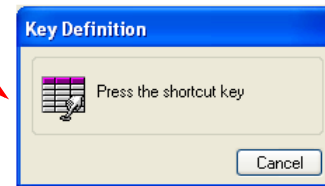
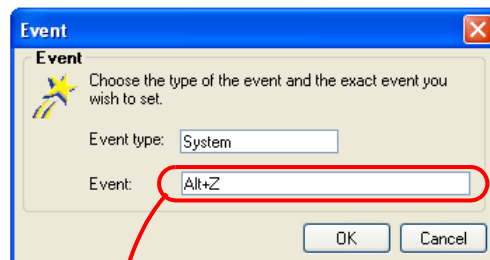
These global events are always coded in the *Main Program* of the project.

## Creating a global event

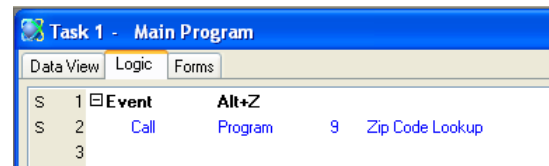
1. Open the *Main Program*
2. Click on the *Logic* tab (*ctrl+1*).
3. Press **F4** (**Edit->Create Line**) to open up a line. You will two new lines appear. Go to the first line and select **Event** from the drop-down list.



4. Tab to the right. An *Event* dialog box will appear.
5. Choose the event type you want to use. In this example we are using a *system* event, which would be a keystroke.
6. Zoom (**F5**, **Edit->Zoom**) from the Event field. from will bring up a *Key Definition* dialog box.
7. Press the keystrokes you want to use to trigger the event. In this case we used **Alt+Z**.



8. Now you have a global even that will be triggered when **Alt+Z** is pressed. All you have to do is add whatever logic you want to execute when the event is triggered. Here, we added a call to “Zip Code Lookup” which would show a list of states and zip codes.



**See also:** Chapter 11, “How do I Define an Event Handler to Be Executed Only When the User is Parked on a Specific Control?” on page 264,

## How do I Work with the eDeveloper Engine as an Event-Driven Engine?

Back in the COBOL days, programs were mainly procedural. That is, the program started at the top, and kept going until it got to the bottom. There was little or no input from the user, other than to start the program.

Today though, most programs are *event-driven*. That is, the program sits there and waits for something to happen. When something happens, it responds in a specific away.

Some simple examples of this are websites and SOAP services. The website is always there, but only responds when you press a key, or hover over a certain field. SOAP services return strings of data, but only if you send them certain strings of data.

You can write both kinds of programs in eDeveloper, but mostly you will be using the event-driven paradigm even in batch programs. This is basically because it is a paradigm that is easier and more efficient to use than the old procedural methods.

### The Concept of Events

The basic concept of events is simple. Something happens (the *trigger*) and the program responds by executing some logic (the *handler*).

The trigger is often something the user does, such as hovering over a certain field or pressing a certain key. Or that the cursor moved into a field, or it moved out of a field. It can also be something that does *not* happen, such as if the keyboard is idle for too long. Or it can be that a certain time was reached.

However, programs also raise events specifically to interact with other programs. A good example of this is an ActiveX object, which contains its own group of specific events which you can respond to, or not.

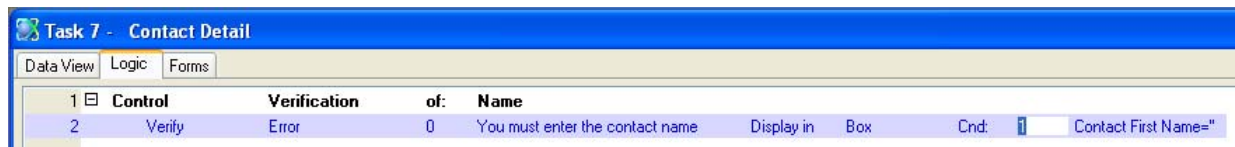
Here are some of the things that can happen:

What happens Events	How it is handled in eDeveloper	
	Logic Header	Subtype
A task just started	Task	Prefix
A task just ended	Task	Suffix
A new record was just read	Record	Prefix
The user just moved off this record	Record	Suffix
The user moved the cursor into a field	Control	Prefix
The user moved the cursor out of a field	Control	Suffix
No one touched the keyboard for x seconds	Event	a timer event
The user pressed a push button	Event	User Event
The value of a certain variable just changed	Variable Change	
It just turned 11:03 AM.	Event	Expression
An ActiveX object raised an Event	Event	ActiveX

What happens  Events	How it is handled in eDeveloper	
	Logic Header	Subtype
The user pressed a certain key or key combination	Event	System
There was a DBMS “Duplicate Record” error.	Event	Error

Although there are hundreds of possible events you can handle, in reality most of the routine work is done automatically for you by eDeveloper. For instance, you do not have to explicitly open or close DB tables, and initialization, default values, and most of your data validation can be handled by your field models. You can also attach selection lists at the model level, or as part of the control on the form, either as called selection list programs or as dynamic combo boxes.

For the events you do want to handle, you will add some lines to the *Logic* section of your task. Let’s take a simple example, forcing entry of a required field.



**Prerequisite:** The control must already be on the form, and have a control name.

## Creating a logic unit

1. In your task, click on the *Logic* tab (*ctrl+1*).
2. Move to the line where you want to enter your logic unit.
3. Press **Ctrl+H** (**Edit->Create Header Line**) to create a new logic header.
4. Select the logic unit type *Control*. Tab to the next field.
5. Select the control type of *Verification*. Tab to the next field.
6. Zoom (**F5** or **double-click**) on the *of:* field to select the control you want to verify. If you don’t see your control on the list, either it is not on the form or it does not have a control name.

You now have a logic unit that will execute whenever the user passes the field in question. Inside that logic unit, we added a *Verify* operation, which will in this case give an error message in a box and prevent the user from doing anything else until they enter some data in the field.

**See also:** The Event Handling concept paper.  
Chapter 4, “How do I Utilize the Event Hierarchy?” on page 57.

## How do I Utilize the Event Hierarchy?

One of the issues to think about with events is, who will handle the event? That is, when an event is raised, there may be several levels of tasks that could handle it. One good example of this would be a DBMS error. If the DBMS raises an error, you may want to give the user an error message for some types of errors. However, you may also want to have a global error tracking program that logs all DBMS errors to a database for reporting.

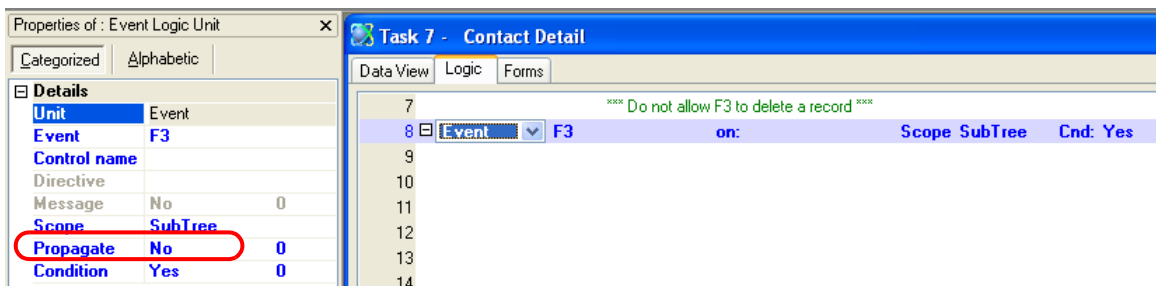
You also might want to override, say, a keystroke. For instance, **F3** is by default mapped to delete line, and it will automatically delete the current record from the DB table. But you might want to trap **F3** and just mark the record deleted instead. Or you might want to trap **F3**, delete some related records in your logic unit, and then have eDeveloper delete the current record.

This is all done using the *Propagate* flag. If propagate is set to Yes, then the logic unit executes, and passes the event up to the next level. If propagate is set to No, then the logic unit executes, and the event is blocked.

eDeveloper uses a specific search order to determine the logic hierarchy. It executes events that are tied to a specific control first, then the events that are not tied to any controls. It proceeds up the task tree, up to the Main Program. Last, eDeveloper handles the event, if it was not blocked.

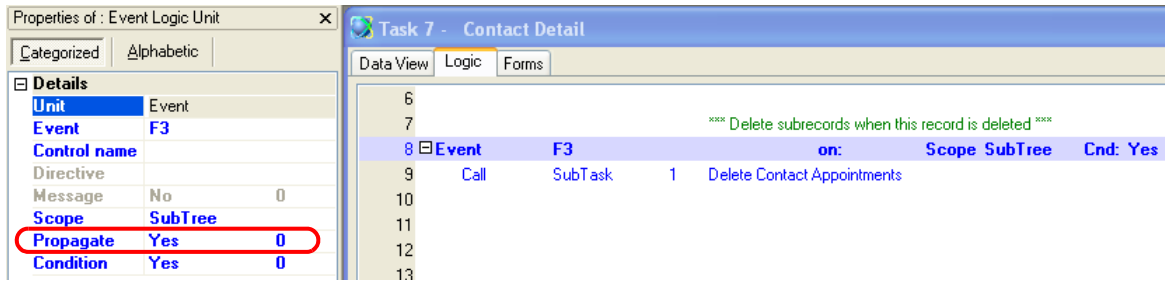
Let's take a specific example.

### Blocking a system event



Here you have an event that totally blocks F3. If the user presses F3, that will trigger this event. But the event does not propagate, so eDeveloper will not handle it, and will not delete the record.

## Adding functionality to a system event



This logic unit executes a call to a subtask to delete some related records. Then it passes the **F3** system event to eDeveloper, which will delete the current record.

**Hint:** Though this example shows how to block a keystroke, it is usually better to intercept an action like **delete line** at the internal event level, because the user can also use the menu options to delete a line, or the keyboard may be re-mapped, or the delete line event might be raised by the program itself, in a push button for example. You can see how to intercept the delete line event in Chapter 4, “Blocking an internal event” on page 59.

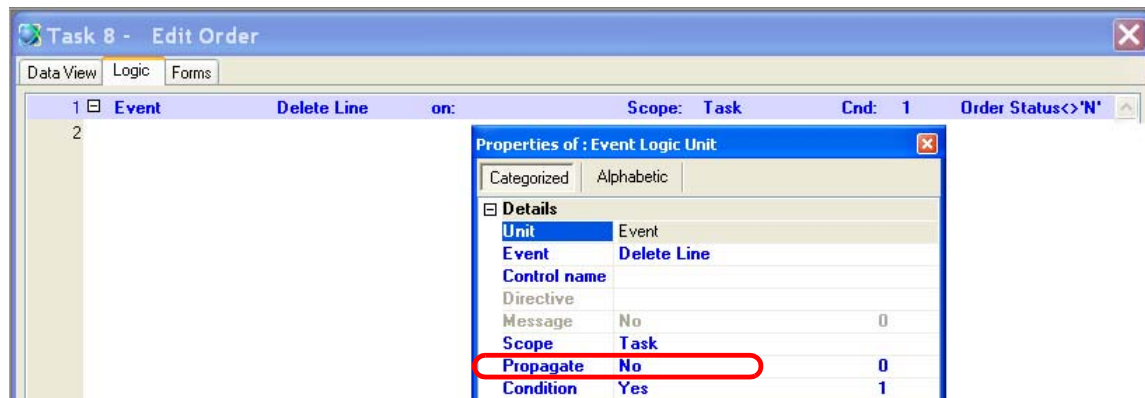
## How do I Prevent an Internal Event (action) like Delete Line, from Occurring?

eDeveloper has a lot of built-in functionality, which saves you a lot of time. Common actions, like **Create Line**, **Delete Line**, and **Exit** are built into the engine and you do not need to specifically code them.

You can block these events at a system level, by changing the default keyboard mapping and menus. You can also redefine some of the events in **Task Properties->Options**. But you have more control and more options, at a task level, by coding a logic unit that intercepts the internal event.

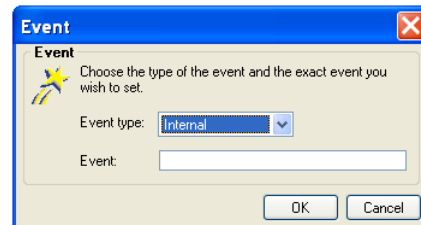
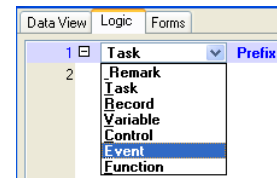
In this example, we will block the delete line action if the status of an order is not 'N' (New).

### Blocking an internal event



Here is a logic unit that will block the Delete Line event, if the Order Status <> 'N'. In this logic unit you might want to do something like give a message to the user, but that is not required to do the blocking. Below we will go through how to enter this event, step by step.

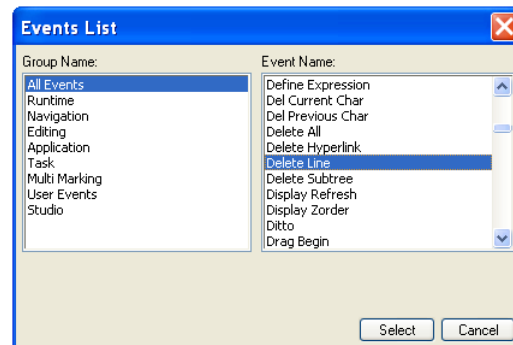
1. In your task, click on the *Logic* tab (*ctrl+1*).
2. Move to the line where you want to enter your logic unit.
3. Press **Ctrl+H** to create a logic unit header.
4. Select *Event* for the header type.
5. A dialog box will appear. For *Event type*, select *Internal*. Then tab to the next field, *Event*.



6. After you tab, an *Events list* dialog box will appear.

You can select the event you want by typing in the first letters of the event. In this case, typing “del” will get us to the delete events, then we can arrow down to “Delete Line”.

After you find the event you want, press the *Select* button, or the *Enter* key.



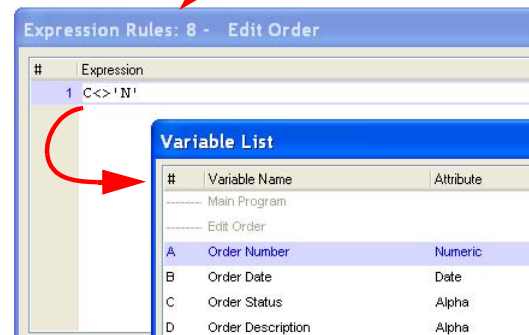
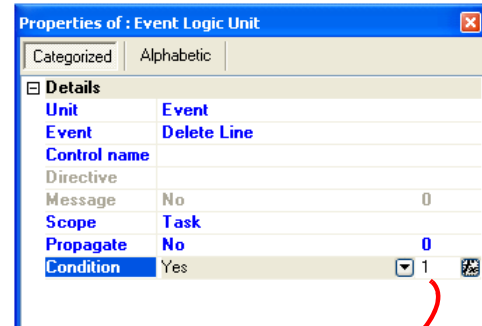
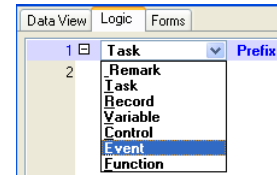


1. In your task, click on the *Logic* tab (*ctrl+1*).
2. Move to the line where you want to enter your logic unit.
3. Press **Ctrl+H** to create a logic unit header.
4. Select *Event* for the header type.
7. Press **Enter** again to close the *Event* dialog box.
8. You now have an *Event* logic unit which will execute when the *Delete Line* event happens. To keep the event from propagating, go to the *Event Properties* pane (**Alt+Enter**), and set *Propagate* to No.
9. You will probably also want to set the *Scope* to “Task”, because usually you would not want to trap the *Delete Line* event at a lower task level.
10. At this point the logic unit will be executed whenever the *Delete Line* event occurs, and trap it.

However, we also wanted the event to only be trapped if the status code was not ‘N’.

To do this we **zoom** (**F5**, double-click) on the *Condition* field.

This brings up the Expression Editor. You can **zoom** from the Expression to find the variables you want, or **Right+Click** to find functions (See Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537).



Data View	Logic	Forms
1	Event	Delete Line
2	Verify	Warning
		0
		Order is being processed, cann Display in Box
		Cnd: Yes

Now you are done! You can add more logic to your logic unit, such as this warning message we added here.

**See also:** Chapter 4, “How do I Work with the eDeveloper Engine as an Event-Driven Engine?” on page 55.

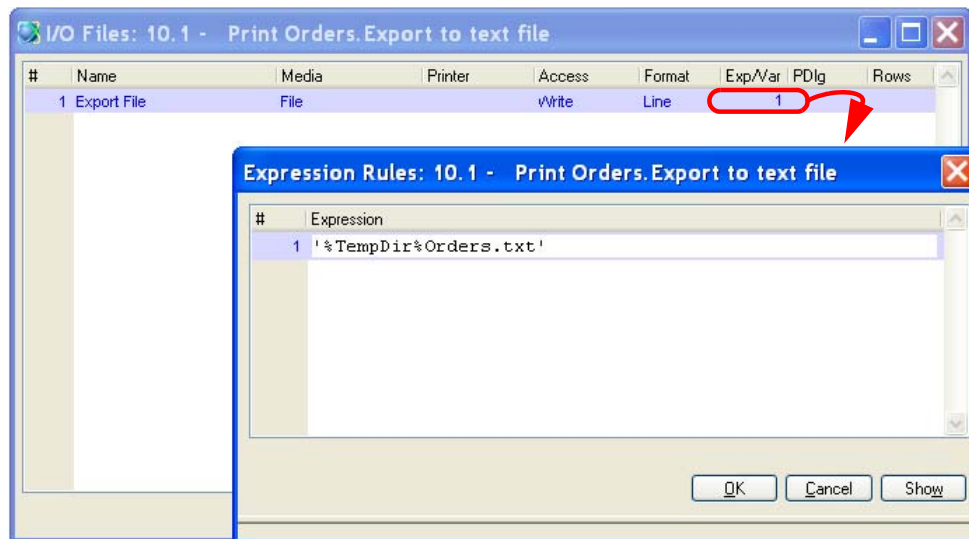
## How do I Set a Dynamic Name for an Input/Output File?

When you create or use an input/output file in eDeveloper, you have several options for coding the file name:

- You can **hard-code** the file name, such as 'C:\Temp\Report.txt'.
- You can use **variables** to hold the file name, such as TRIM(AF). The variable, AF, might have been held in a table, or it might have been a file name selected or typed in by the user. You can also generate the file name randomly and hold it in a variable, which is useful for temporary text files.
- You can use **logical names** to set all or part of the file name. This is a good practice for the path name, as you want to be able to set the path names at runtime. An especially useful logical name is **%TempDir%**, which is automatically set up by eDeveloper.

In all cases, the file name will be entered in the *Expression Editor*. Here we will show you how to do it step-by-step for a text output file.

### Setting the I/O file name



1. First, open up the task which is doing the I/O. You can do that by clicking on the task in the navigator pane (**Alt+F1**).
2. Press **Ctrl+I** (**Task->I/O Devices**). This will take you to a list of your I/O devices. If you don't already have one in the task:
  - Add an I/O device by pressing **F4** (**Edit->Create Line**).
  - In the Name column, give this I/O device a name that is meaningful to you.
  - In the *Media* column, select *File*.
  - In the *Access* column, select *Write*.
  - In the *Format* column, select *Line*.
3. Now, tab to the Expression column and **zoom** (**F5**, double-click) to the *Expression Editor*.

4. Enter the expression you want to use for the file name. In this case we used ‘%Temp-Dir%Orders.txt’. You can use any combination of logical names and variables though, as long as it evaluates to a valid path name.

Now, when the export runs, the file will be created in my Windows temporary directory.

**See also:** Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537.  
Chapter 2, “How do I Read and Write Files from/to the Directory of the Project?” on page 27.

## How do I Set a Dynamic Name for a Data Source?

Usually, the data source names are set in the Data repository. This makes maintenance easier; you can find the data source names easily and change them easily in one central place.

However, there will be times when you want to change the data source name at the task level. For instance, you might want to use the same table description to define separate tables for different users, or to create archival copies of data named by date/month.

### Renaming a data source at the task level

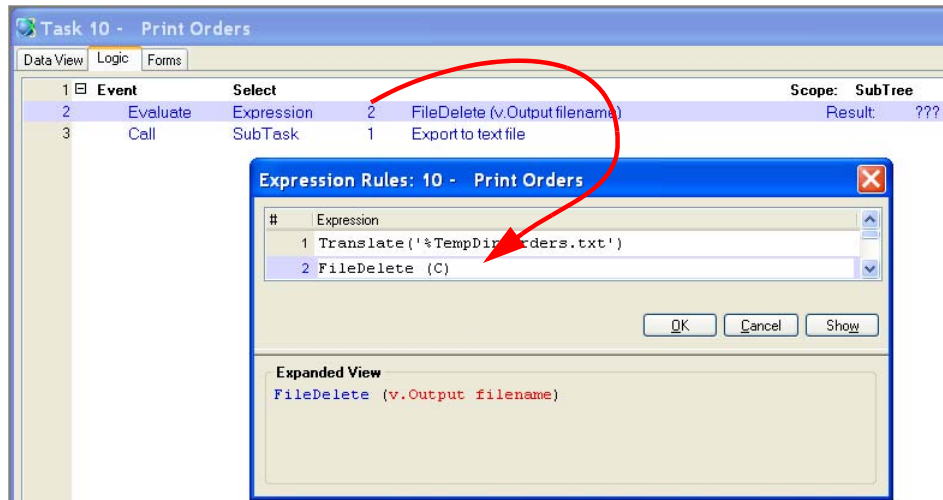
The screenshot displays the eDeveloper Engine interface for a task named "Task 11.1 - Archive Orders.Export to text file". The task is in the "Data View" tab. It shows a "Main Source" (Index 0) and a "Link Write" operation (Index 1). The "Link Write" operation is linked to the "Main Source" and has a "Data source number" of 5. The "Data source description" is "Order Header -- Archive". The "Data source name" is set to "1". The "Data source name" property is highlighted in the "Properties of Link Operation" dialog box. The "Expression Rules" for the "Link Write" operation are shown in a separate window, containing the expression: `Trim(Translate('%Archive%')) & 'OEHDR' & DStr(Date(), 'YYMM')`.

In this example, we have a subtask that creates a copy of the Order Header record. It opens the same item in the Data repository, but uses the Data source name property to override the name. The data will be added or modified in a file named after the year and month ('0712' for Dec 2007, for instance), in the directory represented by the logical name %Archive%.

**Note:** You cannot name the same data source two different names in the same task. In this example, we create the archived record in the *subtask*, because the parent task opens the same data source object under the default name.

**See also:** Chapter 18, "How do I Create a Database Table Using eDeveloper?" on page 457.

## How do I delete a File or a Data Source in a Task that Handles the Same File or Data Source?



When an eDeveloper task starts, all the files and data sources used by that task are opened before you get control of the program. This means that you cannot delete the file or data source from within that task.

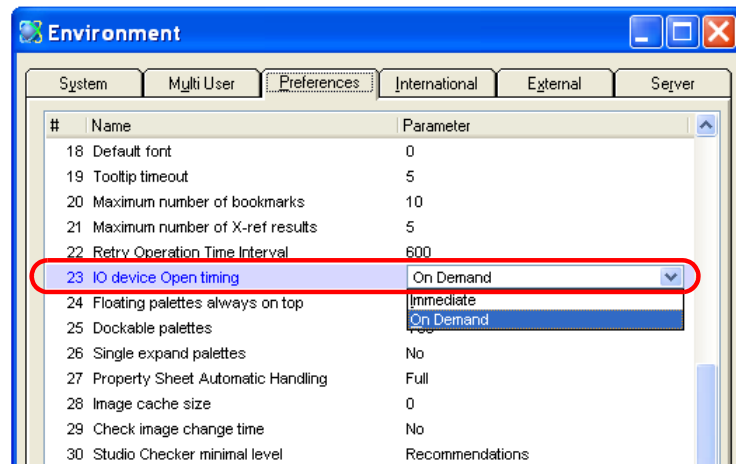
However, you will commonly want to delete a file before a task runs. To do this, you would delete the file in a parent task, usually the one that launches the task.

In this example, we have a logic unit that deletes the old file, then calls the task that will create the file. We are using an function, `FileDelete()`, passing it the name of the file in variable `C`. Because this happens in the parent task, there will be no conflict with the subtask opening the file.

## How do I Prevent The Engine from Creating an Output File or Printing an Empty Page to the Printer when the Task Eventually Does not Output Anything?

By default, eDeveloper opens output files and printers before the task starts. This has been the behavior for all previous versions of eDeveloper. However, if it turns out that the task does not create any output at all, you might be left with an empty file or a blank page on the printer.

As a programmer, you do not always know if there will in fact be output for a given job. So, to prevent empty pages and empty files, you can tell eDeveloper to delay opening the device until it actually has data to print.



### Setting IO device timing

1. Go to **Settings->Environment->Preferences**.
2. Go to line 23, *IO device timing*.
3. Select *On Demand*.

Now, the device will not be opened until eDeveloper is about to execute a *file write* operation.

## How Can I Set a New Task to Automatically Create Logic Units for the Basic Task Levels (Task and Record)?

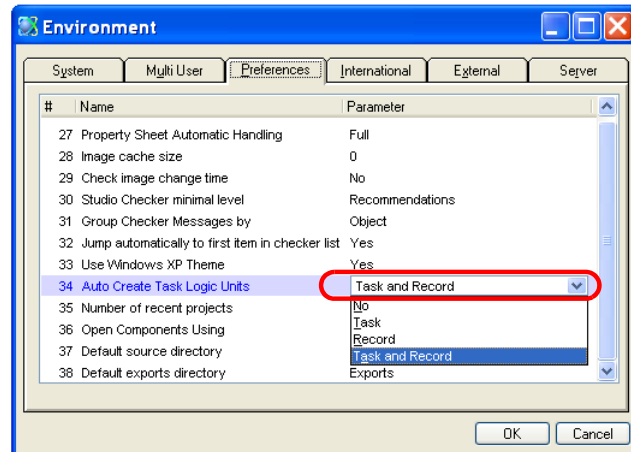
If you have been using previous versions of eDeveloper, you are used to seeing the Task and Record logic units in your programs. They were always shown onscreen, even though often not all of them were used.

By default, eDeveloper 10 does not create any logic units; you create them by pressing **Ctrl+H** as needed. However, if you want to have them automatically created, just change the settings as follows.

### Automatically creating Task and Record Logic Units

1. Go to **Settings->Environment->Preferences**.
2. Go to line 34, *Auto Create Task Logic Units*.
3. Select *Task and Record*.

Now, when you open a new task, the task and record logic units will be automatically created.



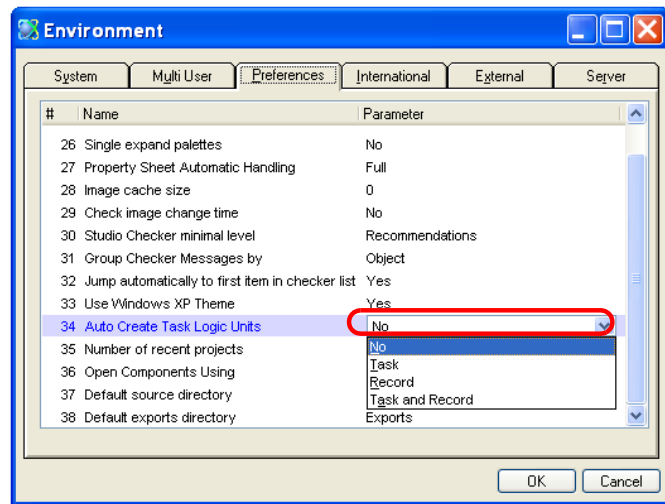
## How Can I Set a New Task to Create No Default Logic Units?

Whether or not eDeveloper automatically creates logic units is set in the Environment settings. If eDeveloper is creating logic units for you every time you create a new task, you can turn off that feature as follow.

### Turning off automatic creation of logic units

1. Go to **Settings->Environment->Preferences**.
2. Go to line 34, *Auto Create Task Logic Units*.
3. Select *No*.

Now, when you open a new task, no logic units will be created.



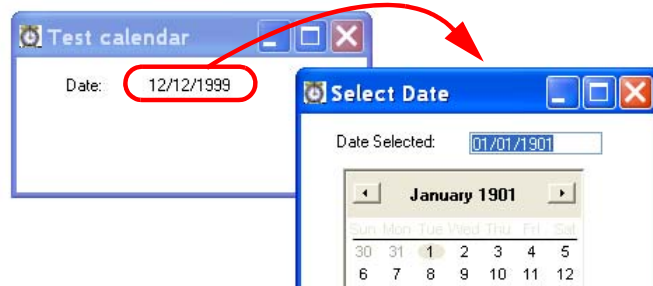


## How do I Retrieve the New value of an Edit Control While Handling Its Events?

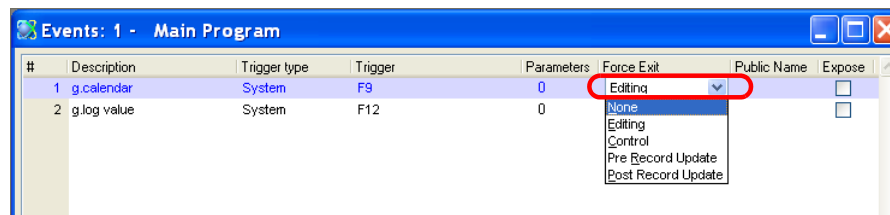
By default, when a handler executes while the cursor is sitting on an edit control, the value of that control has not yet been read. That means that even though the user has in fact typed in a new value, that new value does not exist in the variable until the user leaves the field.

This can be an issue for handlers that are called via a control key. Suppose, for instance, we have a calendar that pops up when the user presses **F9**.

The default date was 01/01/1901. We typed in 12/12/1999 and pressed **F9**. But our handler picked up the original date, 01/01/1901. This is obviously not how we want our program to work!



To change this, we change the *Force Exit* column in the user event.



Using *Force Exit = Editing* will cause the engine to leave Edit mode, update the variable with the edited value, and recompute any values related to the updated variable, before executing any handlers.

Now, the date will get passed as expected.

**Hint:** The *Force Exit* property only applies to user events. If you need to add force exit to a system event such as a keystroke, you need to enter it as we did here, using the system event to trigger a user event.

**Note:** A calendar program is often attached to a date model using the Select program property of the model, in which case the date field is automatically passed in the “user modified” form. However, using logic units gives you more options. For instance, **F9** might bring up a plain calendar selection, but **F10** might bring up my whole Outlook schedule.

**See also:** Chapter 4, “How do I Work with the eDeveloper Engine as an Event-Driven Engine?” on page 55.  
eDeveloper Application Help: Force Exit.

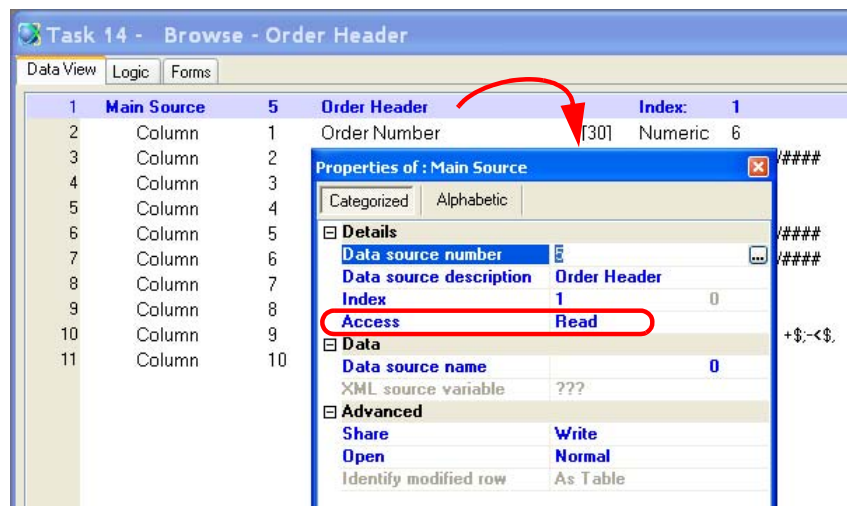
## How do I Prevent the End-User from Modifying Any Record in a Task?

By default, when you create an online task in eDeveloper, the task allows the user to do anything to a record: add, delete, and modify. The data source is opened in write mode, and records are locked if they are modified.

This is very useful for creating quick programs to maintain tables. However, in most applications, users are restricted from modifying certain records, and many screens will be for display only and you don't want to accidentally modify or lock records.

There are several levels of control you have over modification of data in eDeveloper. They are listed here.

### Setting the data source access mode



Setting the *Access* property of the data source is the most foolproof way to ensure the record does not get modified. If you do happen to accidentally modify a record that is opened in *Read*, then you will get a DB error, but the record won't be changed. The table will also open faster if it is opened in *Read*, so it is a good practice for tasks that are not supposed to update data.

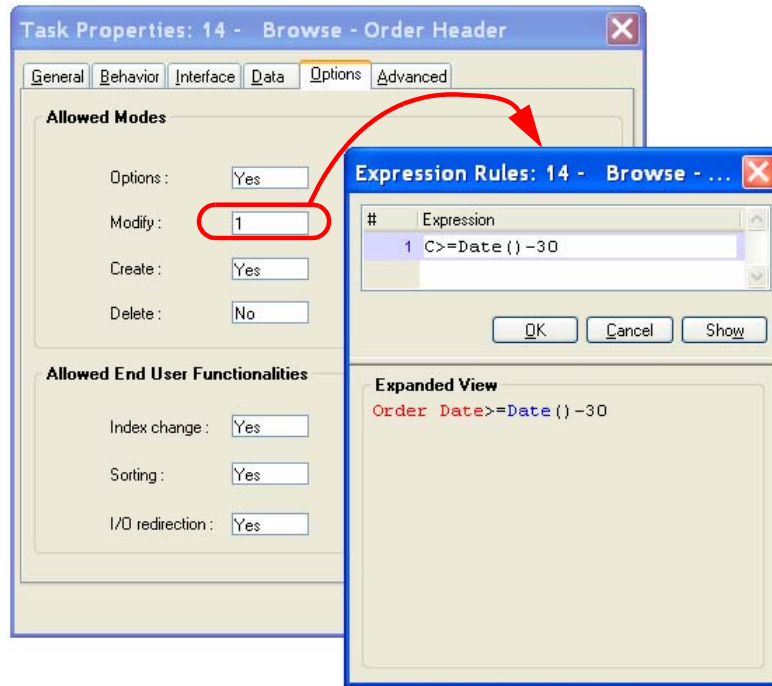
## Task Properties Initial Mode

The image displays two side-by-side screenshots of the 'Task Properties: 14 - Browse - Order Header' dialog box. The left screenshot shows the 'General' tab with the 'Initial mode' dropdown set to 'Query', which is circled in red. The right screenshot shows the 'Options' tab with the 'Options' dropdown set to 'No', also circled in red. Both screenshots show various other settings like 'Task name', 'Task type', 'End task condition', 'Evaluate condition', 'Return value', 'Selection table', 'Resident task', 'Task ID', 'Source file name', 'Allowed Modes', and 'Allowed End User Functionalities'.

Setting the *Initial Mode* in *Task Properties* to *Query* will cause the task to be opened in a non-modifiable mode. This is the mode you want for “lookup” or “view only” tasks.

However, the user can still change the mode from *Query* to *Modify* by selecting *Options->Modify Records (Ctrl+M)*, unless you prevent that by also setting *Task Properties->Options->Allow Options* to *No*, as shown in the frame on the right.

## Task Properties Options



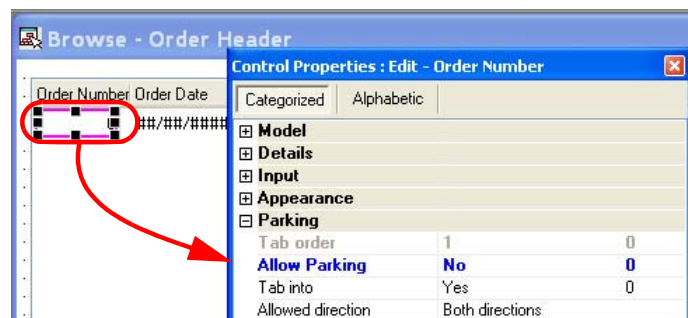
The **Task Properties (Ctrl+P)** option gives you more specific control what is allowed in this task. Each option can be set to Yes or No, or you can enter an expression which will determine whether or not that particular function is allowed, on a user-by-user or record-by-record basis.

For instance, in this case we never allow the user to delete a record on this task. But we allow them to modify a record if the order date is less than 30 days old.

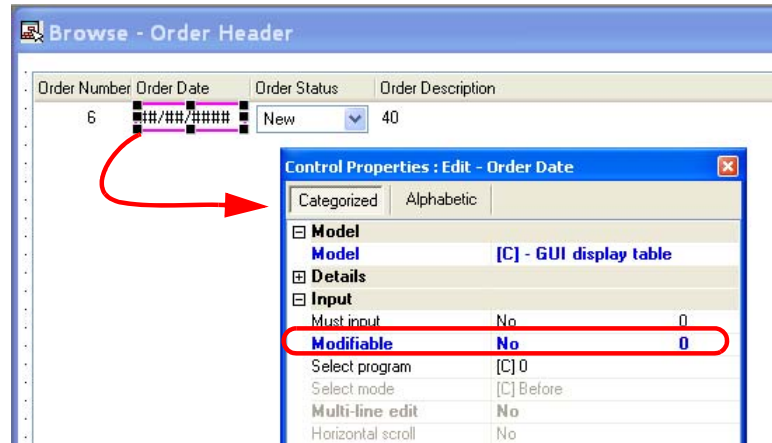
## Field Level Edit

In addition to the data source level control, you also have control at the field level.

- You can prevent the user from parking on a field by setting *Allow Parking* to *No* in the *Control Properties*.



- You can allow the user to park on a field but not modify it, using the control's *Modifiable* property. This can be set to Yes or No, or to an expression if you need boolean logic.



## How do I set the Engine to Execute the Record Suffix Logic Unit Even if the Record has Not Been Updated?

Normally, in an online task, record suffix is only executed if the record has been changed by the user. This is, in fact, what you generally want, because there is no point in writing the record unless there is a change to it.

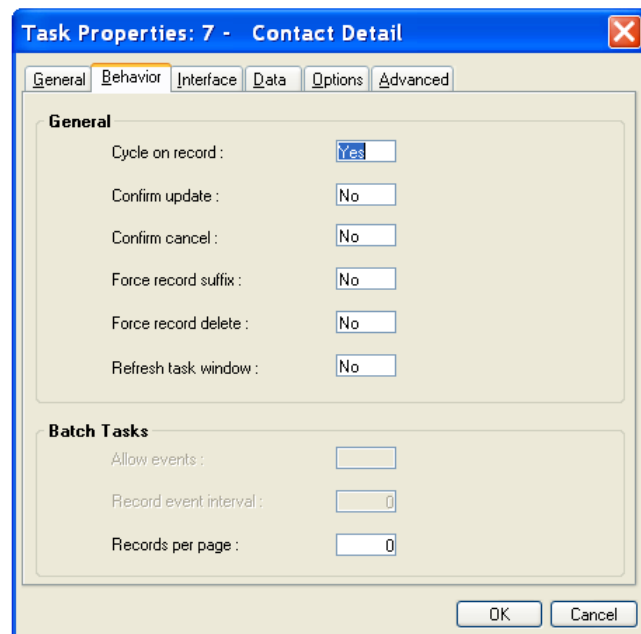
However, there are times when you will want to execute record suffix whenever the user leaves the record, even though nothing was changed. This might be the case, for instance, when you want to display a particular message or do some sort of computations or logging.

### Forcing record suffix to execute

1. Press **Ctrl+P** (Task->Task Properties).
2. Select the **Behavior** tab.
3. Set *Force record suffix* to “Yes”.
4. Alternatively, you can zoom (**F5**, double-click, or **Edit->Zoom**) to create an expression that determines when and when not to force record suffix to execute.

**Note:** You can use the control events to do much of what used to be done in record suffix. Control events give you a much finer level of control.

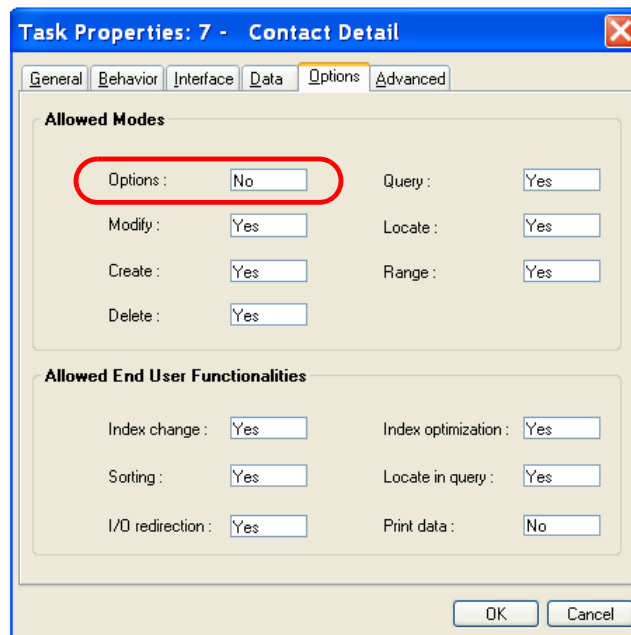
**See also:** Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537,



## How do I Prevent the End User from Changing the Task Modes?

By default, eDeveloper allows the end user to use the same screen to create, delete, modify, and query records. This is good for creating quick data viewing screens, however, most applications will have more restrictions on which users can do what and where. For instance, you may have a task that allows all users to view a record, but only Administrators to change the record.

### Preventing users from changing task modes

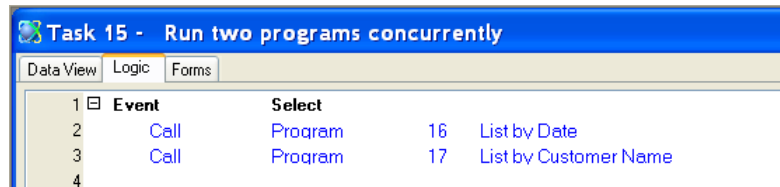


1. Press **Ctrl+P** (**Task->Task Properties**).
2. Select the **Options** tab.
3. Set **Options** to “No”. This will keep the user from changing the current mode of the task.  
Alternatively, you can zoom (**F5**, double-click, or **Edit->Zoom**) to create an expression that will evaluate to *true* when you want the user to be able to change the task mode.

**Note:** You can also use the other options on this tab to allow or disallow specific actions. For instance, if you set Delete to “No”, then the user cannot delete a record in this task.

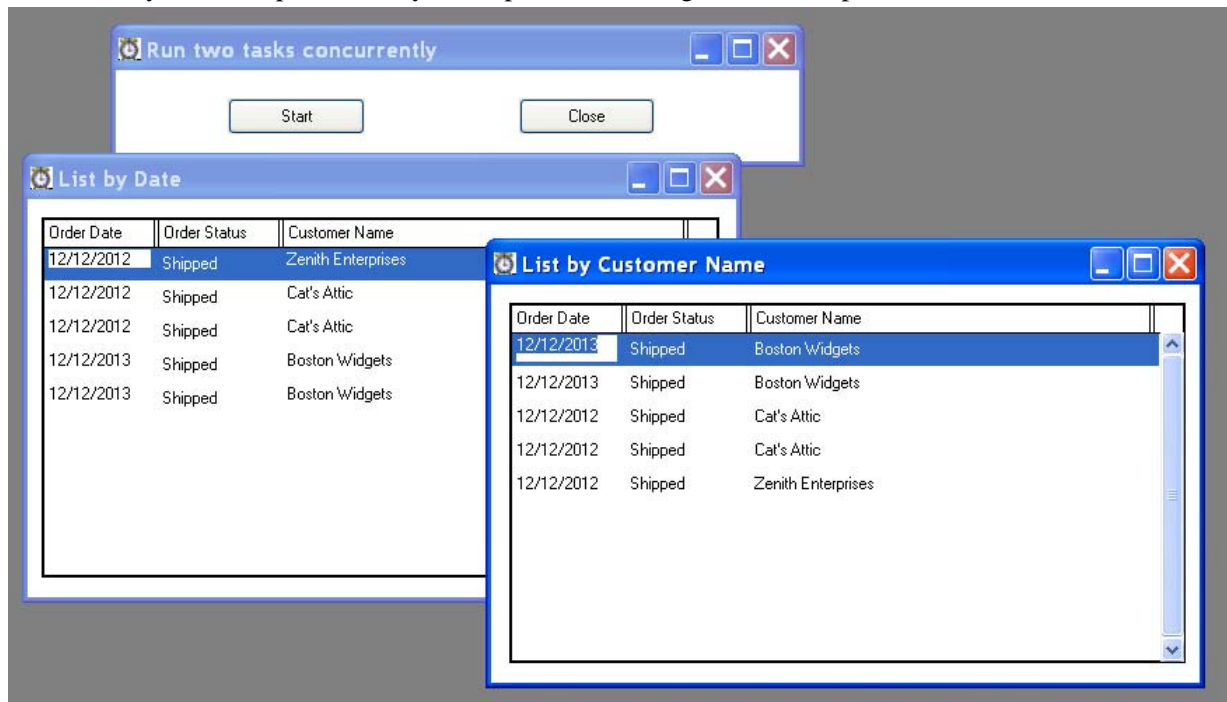
**See also:** Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537.

## How do I Run Two Tasks Concurrently?



Normally, when you call two programs as shown here, eDeveloper will run the first task. Then, when the user exits that task, the second will run.

However, you can set the programs up so that they will both run at the same time. This works a bit like keeping two windows open at the same time. They are both open, but they work independently from each other. Exactly how independent they are depends on settings in eDeveloper.



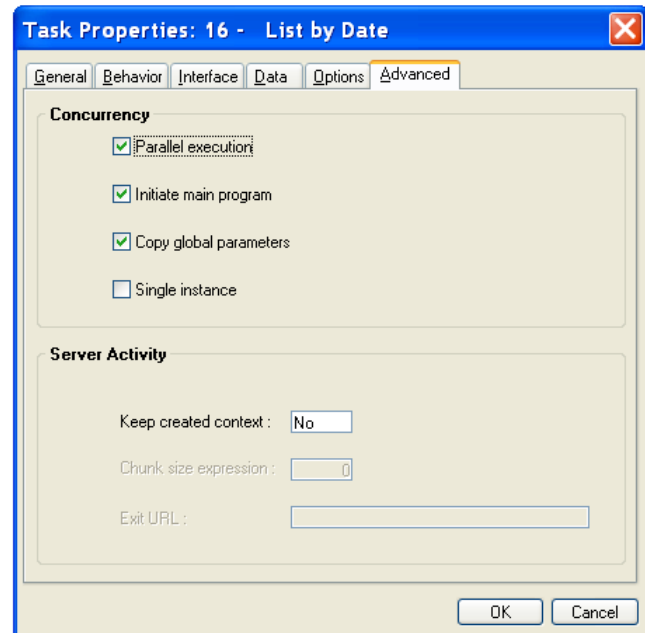
In this example, we set the two tasks that are called to run in parallel. So the first one is called, and then the second, but they both stay up. We can change the focus between the two tasks, and update them independently.

Note that there is nothing preventing locking issues here. The issues will be the same as if two users were accessing the same data. In this example, since the same records are displaying in two different windows, it would be safest if they were display-only lists.



### Setting a program to run concurrently

1. Go to **Task->Task Properties** (Ctrl+P).
2. Select the *Advanced* tab.
3. Check the *Parallel execution* check box.
4. Check the other options as needed.





# Chapter 5: Tasks

## How do I Define a Program Dataview?

Task 19 - Order Entry									
Data View									
Logic									
Forms									
1	<b>Main Source</b>		5	<b>Order Header</b>		<b>Index:</b>		1	
2	Column	1	Order Number	[32]	Numeric	6			
3	Column	2	Order Date	[2]	Date	####/####			
4	Column	3	Order Status	[33]	Alpha	U			
5	Column	4	Order Description	[5]	Alpha	40			
6	Column	5	Desired Date	[2]	Date	####/####			
7	Column	6	Shipped Date	[2]	Date	####/####			
8	Column	7	Customer ID	[26]	Alpha	U12			
9	Column	9	Total Line Items	[9]	Numeric	6C			
10	Column	10	Total \$	[7]	Numeric	9.2CZ +\$.-<\$.			
11	Column	11	Last Seq#	[9]	Numeric	6C			
12									
13	<b>Link Query</b>		4	<b>Customers</b>		<b>Index:</b>		1	
14	Column	1	Customer ID	[26]	Alpha	U12	<b>Direction: Default</b>		
15	Column	2	Contact First Name	[18]	Alpha	40	Locate: 2		To: 2
16	Column	3	Contact Last Name	[19]	Alpha	40			
17	<b>End Link</b>								
18	=== Parameters								
19	Parameter	1	pi.Order#	[32]	Numeric	6			
20	Parameter	2	pi.Mode		Alpha	U			
21									
22	=== Temporary hold fields ===								
23	Virtual	1	v.Total items entered		Numeric	6			
24	Virtual	2	v.Save order?		Logical	5			
25									
26	=== Push buttons ===								
27	Virtual	3	b.View Other Orders		Alpha	U			
28	Virtual	4	b.View Credit history		Alpha	U			

Whatever data is used by an eDeveloper task is declared in the Data View section. If you are using a particular data source, you do not need to bring in every column in that data source, only the columns you are using. You can also declare local variables and parameters as needed.

There are five basic types of data in the *Data View* section. The first three types of data are data sources:

- The *Main Source/Direct SQL*. This is not required, but if it is entered, the task will by default read the entire data source. That is, if it is an online task, the user can view all the items in the table, and if it is a batch task, then the batch task will cycle through every record in the table. However, you can use the Range columns to limit how many records are read, or to read only one record.
- *Linked sources*: These are sources that are linked one-to-one with other sources. In this example, we are bringing in the Customer record that matches this order, to display the first and last name of the customer contact. Note that the entire table does not need to be brought into the Data View; only a few columns might be selected.
- *Declare*: This lets you open a data source in this task, even though you aren't going to use it. This would be done to optimize speed, preventing necessary table open/close operations.

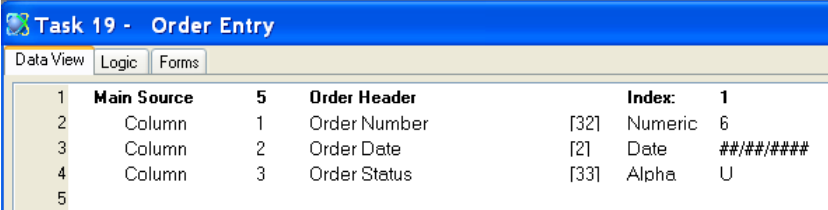
The last two types allow you to declare variables that are not attached to any data source:

- *Parameters*: Data that is passed to and from other tasks.
- *Virtuals*: Virtuals are temporary variables that will only exist for the life of this task.

The Main Source has to be at the top of the list, but otherwise you can create them in any order. However, there might be some dependencies to keep in mind (linked data sources should be below the data sources they are linked to, for example), and it's best practice to organize the data neatly. You can add comments and blank spaces too, which helps a lot with readability.

Below will give an overview on how to create these. There is a lot more to learn in this area though, and we suggest you read the eDeveloper manual and help files for more detail.

## Entering a Main Source



Task 19 - Order Entry						
Data View Logic Forms						
1	<b>Main Source</b>	5	<b>Order Header</b>		<b>Index:</b>	<b>1</b>
2	Column	1	Order Number	[32]	Numeric	6
3	Column	2	Order Date	[2]	Date	###/###/####
4	Column	3	Order Status	[33]	Alpha	U
5						

1. The **Main Source** header line always exists. You don't need to create it. Just tab past "Main Source" and enter the data source# you want to access, or **zoom** (F5, double-click) to select it from a list. Then **tab** again.
2. Now you will be on the data source name field. Notice that you can change this. Changing the name of the data source here does not affect its entry in the Data repository, and it doesn't affect how the engine deals with the data in this task, but it might make it easier to understand your program, especially if the same table is used several times in the task hierarchy. **Tab** again.

- Now you will be on the **Index** field. Type in the index number, or **zoom** to choose it from a list. Choosing the correct index is very important, as it effects the order the records show in on a list, and how fast the filtering works if the range is restricted.
- Last, choose **Properties** from the right-click mouse menu. Here you can choose the details about how the table is opened.

Once you have the main table set up, you can press **F4** on the lines below, and choose the columns you want from that source. These columns are where you will add *Range* criteria to your *Main Source* to restrict the data view. For instance, you may only want to show one order, or only the orders for one customer.

**Hint:** Any data view entries of type *Column* you create in the Data Source will always refer to the *Main Source*, unless they are located within a *Link/End Link*. This is true even if you have several linked files and lots of virtuals in between. For readability though, it is best to keep the *Main Source* header together with its columns.

## Entering a Linked Source

Entering a linked source is much like entering a Main Source.

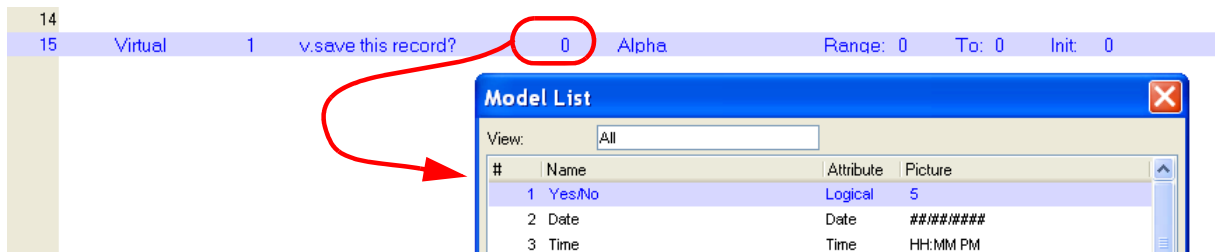
Index	Main Source	Column	Order Header	Index	Index
1	Main Source	5	Order Header	1	
2	Column	1	Order Number	[32]	Numeric 6
3	Column	2	Order Date	[2]	Date ##/##/####
4	Column	3	Order Status	[33]	Alpha U
5	Column	7	Customer ID	[26]	Alpha U12
6					
7			== Fetch Contact Name from Customer Record ==		
8	Link Query	4	Customers	1	
9	Column	1	Customer ID	[26]	Alpha U12
10	Column	2	Contact First Name	[18]	Alpha 40
11	Column	3	Contact Last Name	[19]	Alpha 40
12	End Link				

- First, press **Ctrl+H** to create the header line where you want your linked source.
- In the combo box at the far left, select the type of link you want to do. Each link type works differently, but the *Link Query* is probably the most common. *Tab* to the next field. An *End Link* will appear below your Link header automatically.
- Select the data source you want to link to, either by typing in its number, or zooming to select it from a list. *Tab* to the next field.
- In the *Index* field, select the index you want to use to search for the linked record. Any columns that participate in the link will automatically appear below the link line, so you can use them to locate the linked record.
- Select **Properties** from the right-click menu, to control the details about how the data source is handled.

Now, when you press **F4** between the *Link* and *End Link* lines, you will be able to choose the columns that you want to participate in the link. The *Locate From/To* properties are used to control which record is actually linked. In this example, we are selecting the record where the Customer ID on the Customer record matches the Customer ID on the Order Header.

## Entering a Virtual or Parameter

You can create variables that are not part of any data source. These variables can be of two types: *virtual* or *parameter*. The only difference is that parameters are used to receive data sent from another task, and the number and data types are checked against the sent parameters of the calling task.



1. Go to the line where you want to create the virtual or parameter, and press **F4** (*Edit->Create Line*).
2. Select *Virtual* or *Parameter* from the drop-down list. **Tab** to the next field.
3. You are now in the *name* field. You can type in any name you like. You may want to use some naming conventions, as we did here (using 'v.' as a prefix for a virtual). After you enter a name, **tab** to the next field.
4. Now you are in the *Model* field. You can *zoom* from here and select a field model from a list. Using a model will save you a lot of time, help prevent errors, and standardize your tasks. If you use a model, you just select the model here and you are done. Otherwise, **Tab** to the next field.
5. Now you are in the data attribute field. If you didn't use a model, select the data attribute (Alpha, Logical, Date, Time, etc.) by typing in the first character, or select it from the pull-down list. **Tab** again.
6. Type in the picture for this field, or press *zoom* to get a dialog to help you enter it. In the picture dialog, you can also press **F1** to get context-sensitive help, which will tell you all about entering data pictures.
7. Press **Alt + Enter** or click on the properties pane to enter the data properties for this field. The data properties specify not only the size and valid values of the field, but can also specify how it looks on the screen (the control it uses). If you used a model, these values may already be set up for you, but you can override the inherited values here if you need to.

**Hint:** One of the most common errors for new *eDevelopers* is creating fields with no picture. Once you select the data attribute, go immediately to the properties and enter the picture so the field has a length. If you run a program with a zero-length field, it will not run properly.

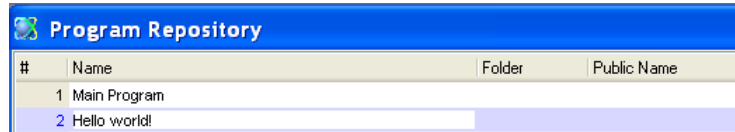
**See also:** Chapter 3, "How do I Define Reusable Data Objects?" on page 38.

## How do I Create a Simple Program?

In most programming books, you will find an example of a simple “Hello world!” program. Here are the instructions for “Hello world!” in eDeveloper.

**Hint:** While you are just starting out in eDeveloper, always remember the *F1 Help* key! There is a lot of context-sensitive information available just a keystroke away.

### Creating “Hello world!” in eDeveloper



1. Go to the Program repository (**Shift+F3**), to the spot where you want to create your new program.
2. Press **F4** (**Edit->Create Line**) to open up a new line.
3. Type in the name of your program, in this case, Hello World. eDeveloper does not use this name internally, so you can use any naming convention you like.
4. Press **zoom** (**F5**, double-click) to open up your new program.
5. Because this is a new program, the **Task Properties** dialog box will open up. You can just escape or press **OK**; the defaults will be fine for a simple program.



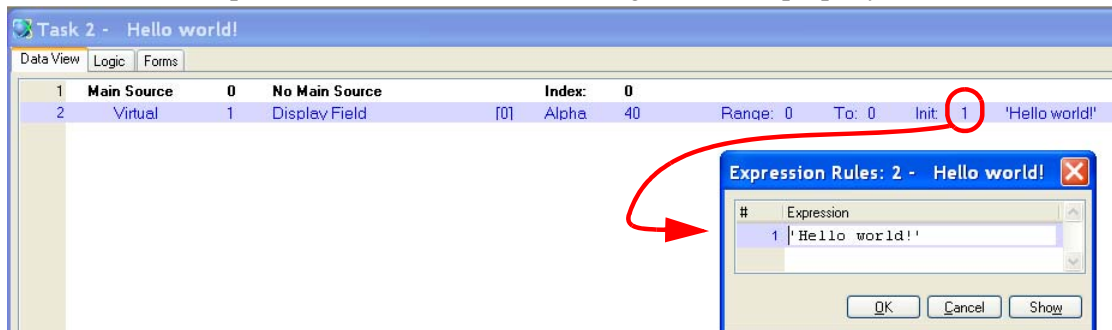
6. Next you will see your new program. Notice the three tabs at the top: *Data View*, *Logic*, and *Forms*. This is where you will do your coding in eDeveloper.

### Creating the Data View



1. First, let's set up the data view. Click on the *Data view* tab. Note that there is already a Main Source line. You can ignore that, because we are not using a data source in this example.
2. Press **F4**. This will open up a line.
3. In the pulldown box, select **Virtual**. This means we are creating a temporary variable. Tab to the next field.
4. Give your virtual a name. We called ours `Display Field`.

5. Tab twice, to the field that says *Alpha*. This is the data attribute, which is alpha by default. You can click on the pulldown here to see the other choices. Tab again. Now you will be on the *Picture* field. Type in 40. This means our alpha field will be 40 characters long. Tab to the property labelled *Init*.

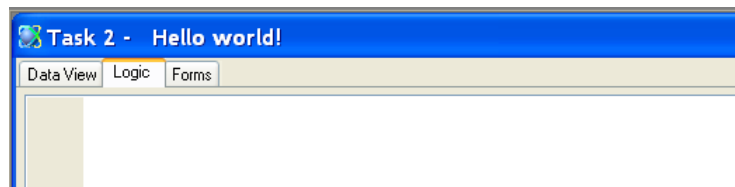


6. The *Init* property is where you will create values that are updated immediately when the task starts, or are recomputed during execution. In this case, we are going to initialize the field to Hello World. Press F5 to **zoom** to the *Expression Rules*.
7. In the *Expression Editor*, press F4 to open up a line, and type 'Hello world' (including the single quotes). Press OK.
8. The *Expression Editor* will close, and the number '1' will appear next to the Init: prompt. This means that the Init: is pointing to expression #1. You will also see all or part of the expression to the right of the Init: field.

**Hint:** For a simple expression like this, you can also just type = in the Init column. This will open up a small box, and you can type the expression there.

And that is all you have to do for this program in the **Data View** editor.

## Creating your logic

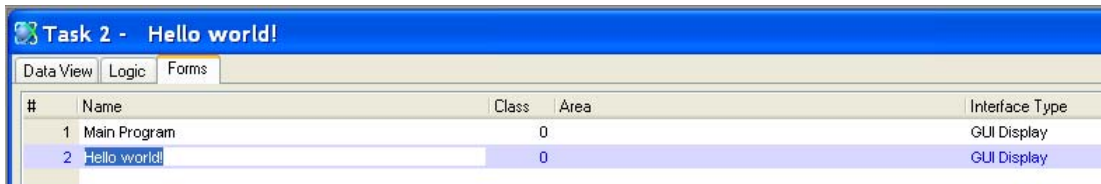


For this program, you don't need to do anything in the Logic Editor. eDeveloper will take care of most of the housekeeping for most programs, so there is usually not as much explicit logic as you would expect.

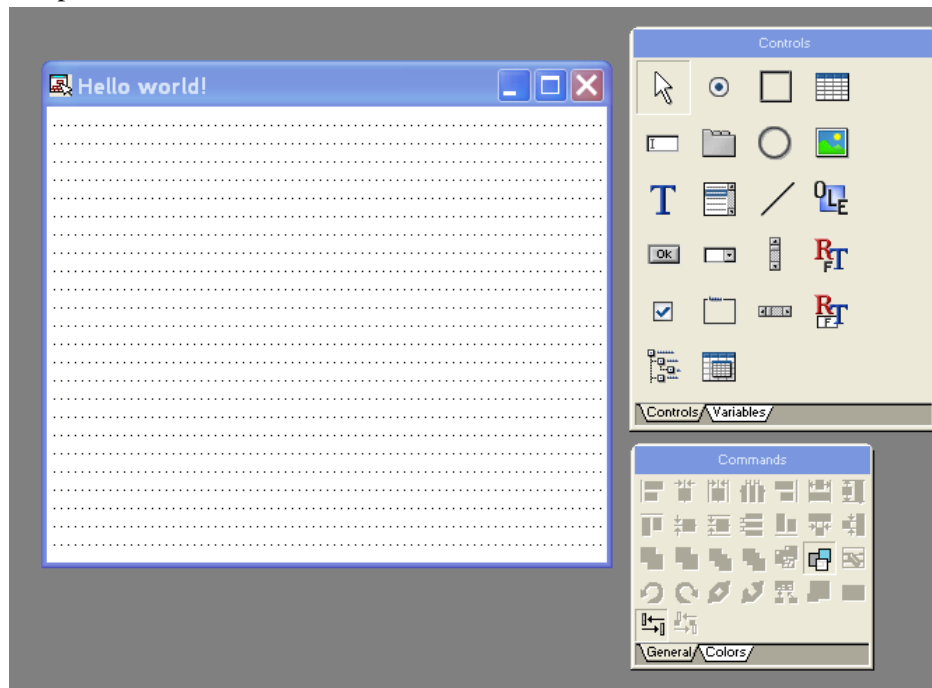
What you would be creating here would be certain kinds of field validations, calling other programs. See Chapter 4, "How do I Work with the eDeveloper Engine as an Event-Driven Engine?" on page 55 for more details about using logic and events in eDeveloper.



## Creating your display



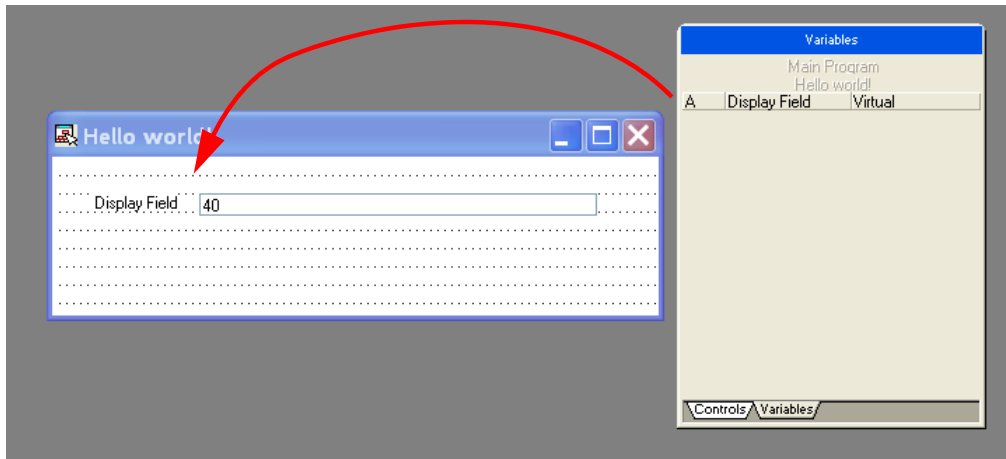
1. Click on the *Forms* tab. You will already see form created, named “Hello World!”. By default it will be named the same thing as your task, but you can rename it if you want. Also, by default, this is the text that will show on the title bar of the window.
2. **Zoom** (F5 or **double-click**) on the Name of the field. Now you will see your display. It is basically an empty window at this point. You can reposition it, or drag on the sides of the window to resize it. The properties pane (Alt+Enter) will also show you a lot of ways you can change this window. But for now, we’ll just accept the defaults.



3. Besides your display, you should see two extra boxes, labelled *Controls* and *Commands*. These are palettes you will use while editing forms in eDeveloper. The same options are also available on the overhead menu and via shortcut keys. It is good to get familiar with the options and the shortcut keys.

If for some reason the *Controls* and *Commands* are not showing in the workspace, select *View->Form Editor Palettes* (also available on an overhead icon). This toggles the palettes off and on.

4. Press the Variables tab on the Control palette. This will show you all the variables we have in our data view, which in this case is just one.

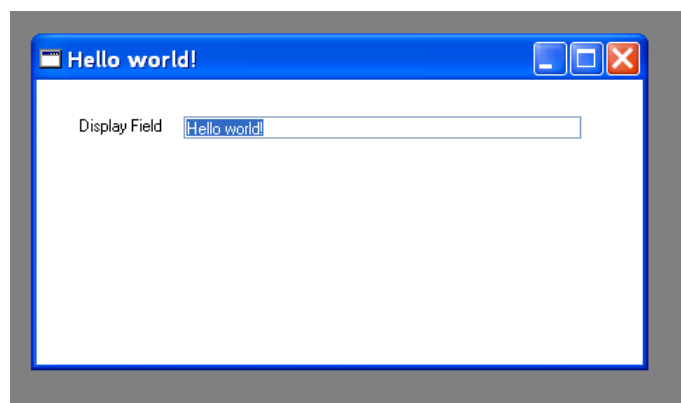


5. Click on the variable. The cursor will change to a box, indicating we are selecting an edit field. Click on the form to drop the variable on the form. Note that the field is pasted, and so is the field prompt, which by default is the name of the virtual. If you name your virtuals carefully, you can save time creating your form.

That's all you need to do here. Select *Options->Save and Close Object* to jump back to the Program repository.

## Running your program

1. Before you run your program, press **F8** (**Options->Check Syntax**). If the program is ok, you will get a message on the prompt line "Program is OK". Otherwise error messages will appear in the Checker pane. You should always fix the errors before running the program, because as with any programming language, some errors will result in unexpected behavior.
2. To run your program, select *Debug->Run* (**F7**). A new window will appear, running your program as it would appear in the eDeveloper runtime engine. While the program is running, your development environment is locked, in read-only mode.



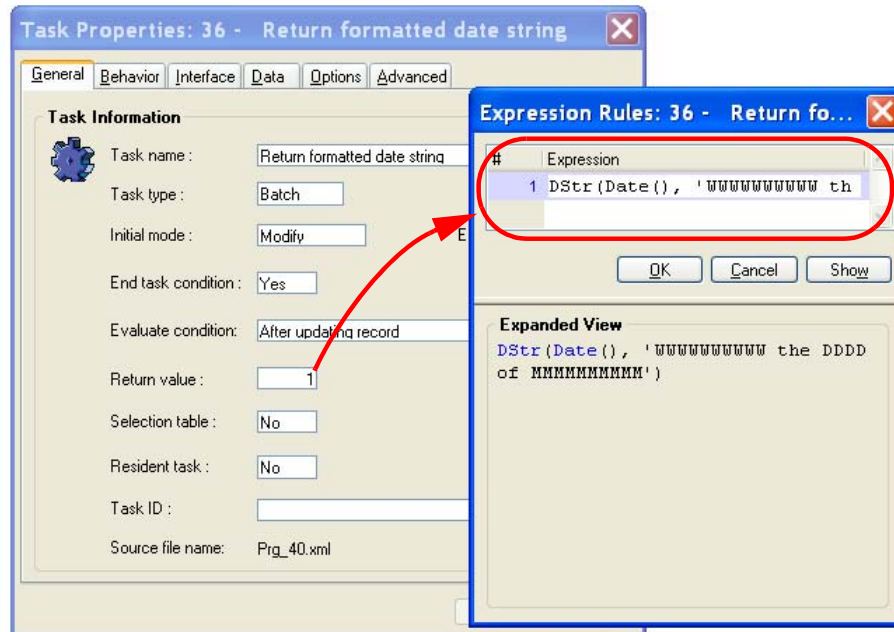
3. Here is your Hello world! program. When you exit out, you will be back in the eDeveloper Studio, and you can do more programming and re-test it. You can do incremental programming easily in the Studio this way.

There is also a full debugger available, so while you are running your program you can see what the eDeveloper engine is doing internally.

## How do I Set a Program to Return a Value to the Calling Program?

There are two ways that a program can return values to a calling program. The first is to use parameters, which are covered in Chapter 5, “How do I Synchronize Parameters Between Called Program and Calling Program?” on page 101. The other way is to use a return value.

The return value is what you need when you call the program from an expression and certain other kinds of calls. Here is how you enter it.

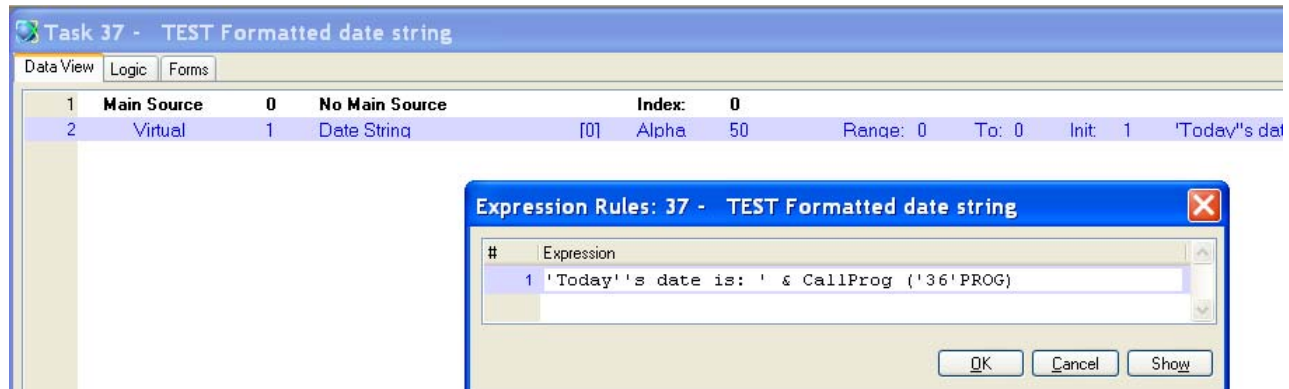


### Creating a return value

1. In your called program, zoom on **Task Properties->General->Return Value**. Create an expression for the value you want to return.

In this case, we have a batch task that iterates once, and returns today's date, formatted for printing on certain reports according to our company's standard. We do this in a program so that if the company standard changes, we only have one routine to change.

## Using a return value



1. Now, you can use your program anywhere you can enter an expression. In this example, we use it in an init field. The function **CallProg** calls our program, and the value is automatically returned into the string we are creating.

Note that **CallProg** calls the program by number. What happens if the program moves from position 36? You don't need to worry about that, as long as you code the literal as shown here, with the letters PROG following the number in single quotes. eDeveloper will keep track of the number if it changes.

However, you can also use **CallProg** with the *public name* of the program, which is more readable. The syntax for that would be:

```
'Today's date is: ' & CallProg(ProgIdx('DateString','TRUE'LOG))
```

See the F1 Help for the function **CallProg** and **ProgIdx** for more information.

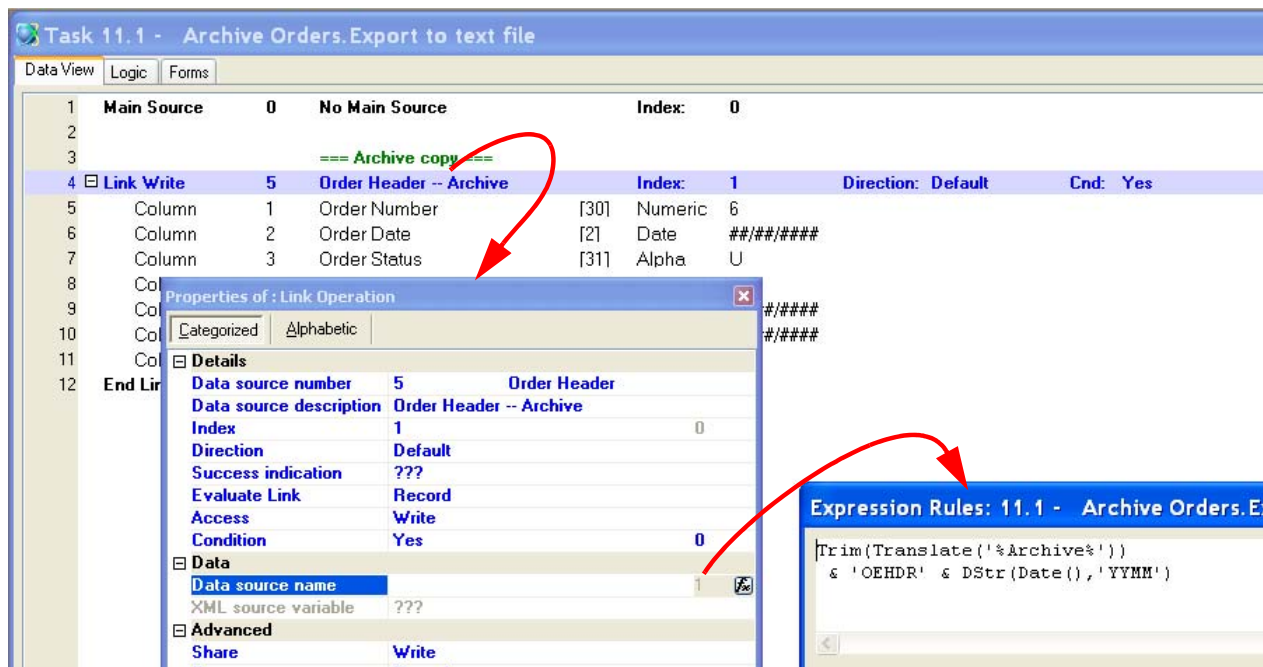
**Note:** eDeveloper gives you a lot of options for standardizing your programs. In this case, we could also have used a model to standardize our date format, or a function in the Main Program.

## How do I Open a Data Source Using a Physical Name that is Different than the one Specified on the Data Repository?

Usually, the data source names are set in the Data repository. This makes maintenance easier; you can find the data source names easily and change them easily in one central place.

However, there will be times when you want to change the data source name at the task level. For instance, you might want to use the same table description to define separate tables for different users, or to create archival copies of data named by date/month.

### Renaming a data source at the task level



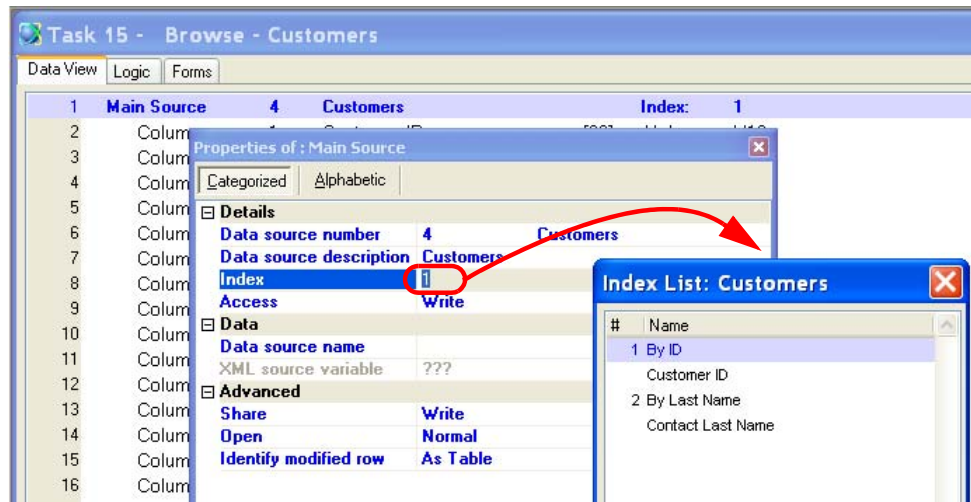
The screenshot shows the 'Task 11.1 - Archive Orders.Export to text file' interface. The main window displays a table with columns: Main Source, Index, Order Header -- Archive, Index, Direction, and Cnd. A red arrow points from the 'Order Header -- Archive' column to the 'Properties of : Link Operation' dialog box. The dialog box has tabs for Categorized, Alphabetic, and Details. The Details tab is selected, showing properties like Data source number (5), Data source description (Order Header -- Archive), Index (1), Direction (Default), Success indication (???), Evaluate Link (Record), Access (Write), Condition (Yes), and Data source name (1). A red arrow points from the 'Data source name' property to the 'Expression Rules: 11.1 - Archive Orders.E' window, which contains the expression: Trim(Translate('%Archive%')) & 'OEHDR' & DStr(Date(), 'YYMM').

In this example, we have a subtask that creates a copy of the Order Header record. It opens the same item in the Data repository, but uses the Data source name property to override the name. The data will be added or modified in a file named after the year and month ('0712' for Dec. 2007, for instance), in the directory represented by the logical name%Archive%.

**Note:** You cannot name the same data source two different names in the same task. In this example, we create the archived record in the *subtask*, because the parent task opens the same data source object under the default name.

**See also:** Chapter 18, "How do I Access an Existing Database Table?" on page 464.

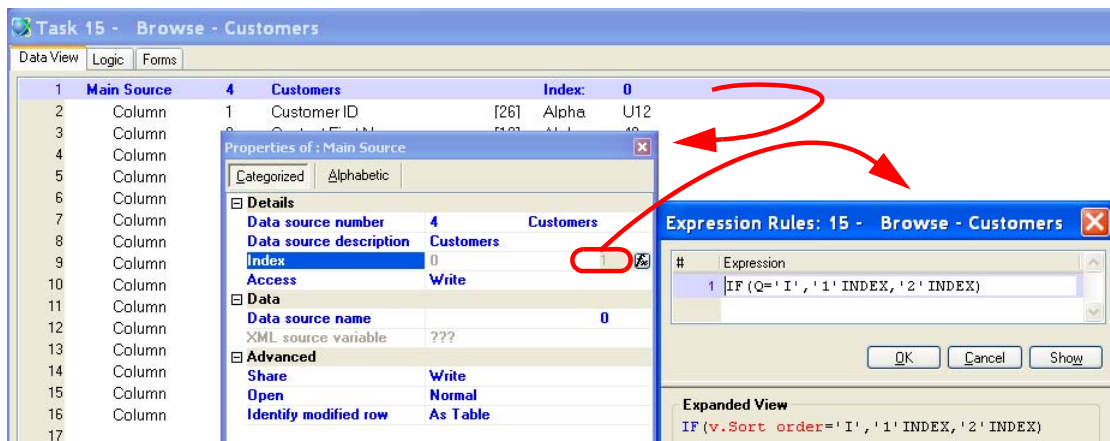
## How do I Dynamically Change the Display Order of Records in a Program?



The record display order can be determined two ways: by creating a run time sort in the sort repository, or, more commonly, by selecting different indexes in the properties of the *Main Source*. In this example, we selected Index 1, so the records will be sorted by Customer ID.

However, you can also set the index dynamically, using an expression. This is commonly done so the user can change the display order of a table, or to optimize processing time depending on what filters the user chose for a report.

### Using an expression for an index



1. Go to *Data View*, and position on the *Main Source*. Press **Alt+Enter**. You will be on the **Properties of Main Source**.
2. Type 0 (zero) in the Index field. That will allow you to tab over to the right, where the index expression can be coded.

3. *Zoom* from the index expression, or press the *fx* button.
4. Enter your index expression. Here we are selecting index 1 if the sort order is 'I' (for ID); otherwise we use index 2. Using the format ' 1 ' INDEX ensures that if the indexes are moved in the data source, the expression will remain correct.
5. Press OK, which will bring the expression number back to the properties pane.

**Note:** Users also have the option of clicking on a column header to change the sort order of displayed records, or using the runtime option **Ctrl+K** (**Options->View by Key**), or creating their own sort order using **Ctrl+T** (**Options->Sort Records**).

**See also:** Chapter 20, “How Can I Dynamically Set the Order of the Retrieved Records in a Task?” on page 501.



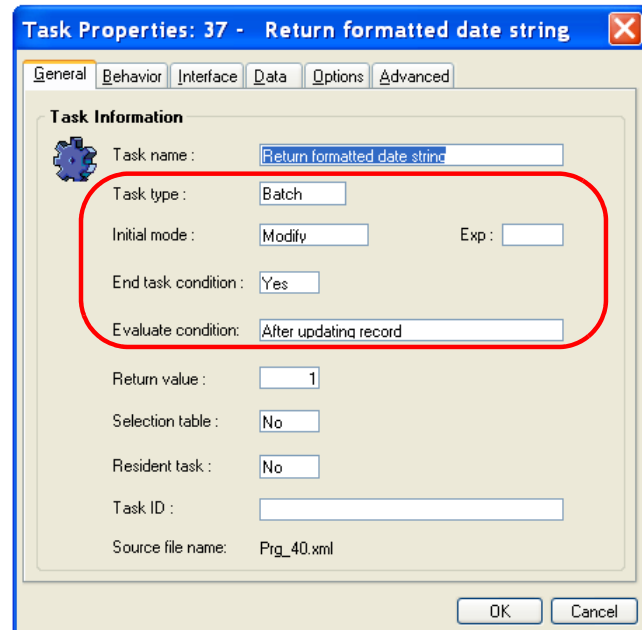
## How do I Create a Simple Batch Program?

There are two kinds of simple batch programs:

- Programs to cycle through all the records in a data source, commonly to export them to a file. You can create these easily using the *program generator* (*Ctrl+G*). Chapter 5, “How do I Create Tasks that Dump Data Records Into Flat Text Files and Vice Versa?” on page 97.
- Programs that cycle once, usually to return a value. We’ll cover how to create these kinds of programs here.

### Creating a simple batch program

1. In the Program repository, press **F4** to open up a line.
2. Give your program a name. This name is not used by eDeveloper, so you can use any naming convention you like.
3. **Zoom (F5)** on the program name. Because this is a new program, you will see the **Task Properties** dialog box.
4. By default, the *Initial mode* is *Modify*. This is generally what you want. If you use Query, no records will be updated.
5. Select *Task type Batch*.
6. Select *End task condition Yes*. This will prevent the task from looping forever. Instead, it will only loop once. If you want it to loop some number of times, you can zoom here and enter an expression to control when the task exits.
7. Select *Evaluate condition After updating record*.



Since this is a batch program, by default no screen will show to the user. If it is a long process, you may want to show some “processing ...” window.

Now you can create your logic to do whatever you want to do. You may want to update records, or pass back data in parameters or, as shown here, in a return value.

**See also:** Chapter 22, “How do I Export Data into a Text File?” on page 569.  
Chapter 5, “How do I Set a Program to Return a Value to the Calling Program?” on page 88.

## How do I Stop a Batch Task from Running Forever?

By default, a batch task will run forever.

To keep it from running forever, you need to one of the following:

- Select a main source in the data view. If a main source is selected, the task will cycle through each record in the main source.
- Enter an end task condition in task properties, as shown on the right. If you only want the task to cycle once, enter

*End task condition:* Yes

*Evaluate condition:* After updating record

- If you want the task to end based on some other criteria, **zoom** from the *End task condition* field and enter an expression. When the expression evaluates to true, the task will end.

Task Properties: 37 - Return formatted date string

General Behavior Interface Data Options Advanced

**Task Information**

Task name: Return formatted date string

Task type: Batch

Initial mode: Modify Exp:

End task condition: Yes

Evaluate condition: After updating record

Return value: 1

Selection table: No

Resident task: No

Task ID:

Source file name: Prg\_40.xml

OK Cancel



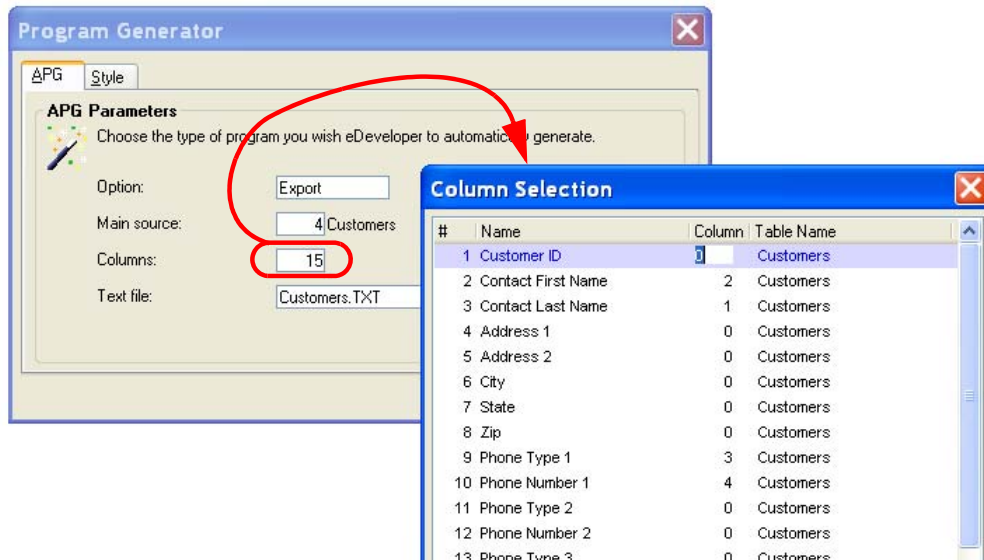
Now, when this task is run, the records that meet the range criteria will be deleted from the data source.

## How do I Create Tasks that Dump Data Records Into Flat Text Files and Vice Versa?

One very common programming task is to dump data into a flat file, or to read data from a flat file. Fortunately eDeveloper has a very nice facility for doing most of the work for you.

### Dumping records to a text file

1. Open up a new line in the Program repository by pressing **F4** (**Edit->Create Line**).
2. Press **Ctrl+G** (**Options->Generate Program**).



3. The *Program Generator* dialog box will appear. Select:

*Option:* Export

*Main source:* Whatever data source you want to export. You can zoom to select from a list.

*Columns:* Zoom here to select which columns will export. By default, all columns will export, but in this example, we set most of them to zero so only the Last Name, First Name, Phone Type 1 and Phone Number 1 will export.

*Text file:* You can enter another file name here if you want. You can also change it within the generated program to use a variable or logical name path.

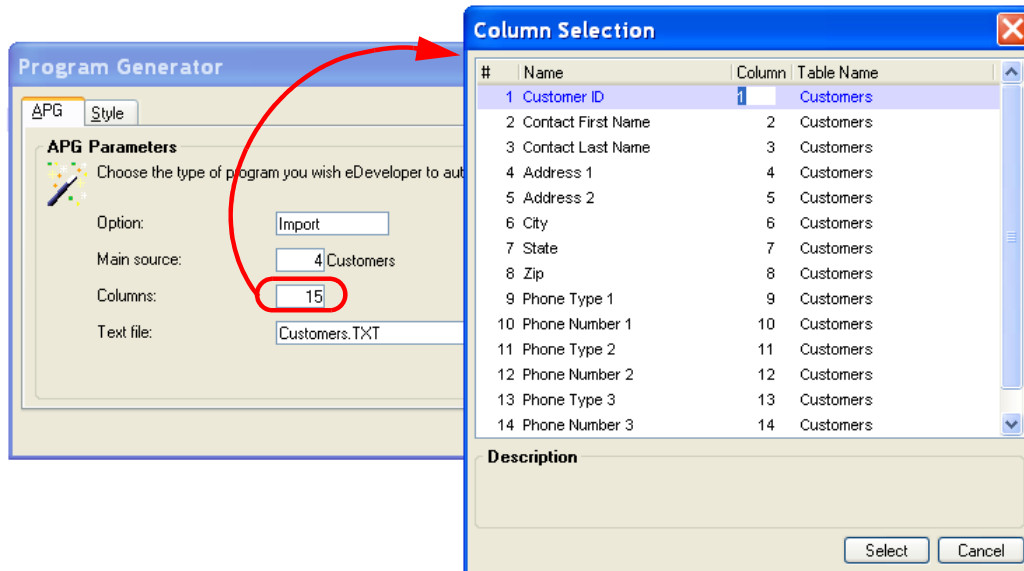
4. Click OK.

Your program is now created, and when you run it, it will export every record from the main source. You can edit the program as you would any eDeveloper program, changing the output format and adding ranges to limit which records are exported.

### Reading records from a text file

1. Open up a new line in the Program repository by pressing **F4** (**Edit->Create Line**).

2. Press **Ctrl+G** (Options->Generate Program).



3. The *Program Generator* dialog box will appear. Select:

*Option:* import

*Main source:* Whatever data source you want to import. You can zoom to select from a list.

*Columns:* Zoom here to change which columns will import. By default, all of the columns will import, as shown here.

*Text file:* You can enter another file name here if you want. You can also change it within the generated program to use a variable or logical name path.

4. Click **OK**.

Your program is now created, and when you run it, it will import every record from the flat file. You may have to edit the import form to match your flat file, depending on the data layout.

Note that the automatically generated program does not check for duplicate records, so it is up to you to either run this on an empty data source, or make sure the data isn't duplicated, or you will get error messages from the DBMS.

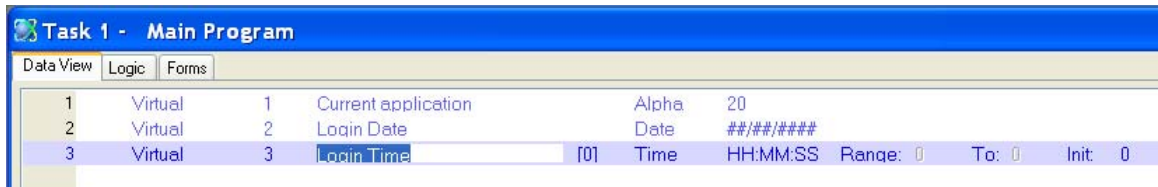
**Hint:** If you accept the defaults for the column layout, the export program eDeveloper creates is a perfect match for the import program, so you can use the generated programs as a quick way to copy data from one location to another.

# How do I Define Global Values that Can be Dynamically Defined and Accessible by Various Programs?

When you define variables and declare data sources in an eDeveloper task, those values are only visible to that task and to subtasks within the same program. However, sometimes you will need values that are available to all tasks within the project.

These global variables are entered the same way as any other variable, except that they are entered in the *Main Program*.

## Defining global variables



- 1. Go to the Program repository (*Shift+F3*). Go to program #1, the *Main Program*.
- 2. Zoom (*F5*) to open the program.
- 3. Click on the data source tab.
- 4. Proceed as you would for creating the data view in any other task.

**Hint:** You can link data sources in the main program, but if you do, be careful it is not a table that is used by a lot of users, because the main program stays resident for the user's entire session. It is safer to use virtuals in the main program, and fetch or store the data in Task Prefix and Task Suffix.

**See also:** Chapter 5, "How do I Define a Program Dataview?" on page 79.

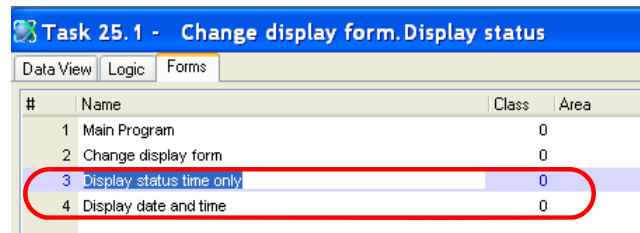
## How do I Dynamically Instruct the Task to Open a Different Form Than the Main Display Entry?

Normally, each task has its own form which may display to the user. However, you can change which form is displayed by instructing eDeveloper to use a different form at runtime.

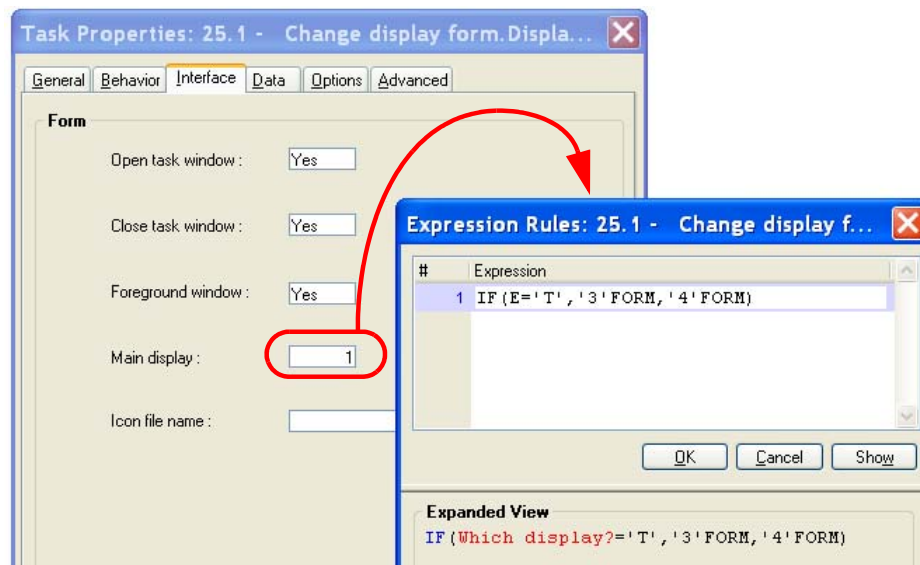
### Choosing forms at runtime

**Prerequisite:** The forms must already be created, and be located in this task (not a parent task).

1. Click on the *Forms* tab. You will see a list of forms. Some of the forms belong to this task, and some belong to parent tasks. In this case, the lower two forms, 3 and 4, belong to this task. One displays the login date and time. The other just displays the time.
2. Press **Ctrl+P** (**Options->Task Properties**). This will bring you to the task properties dialog box. Click on the *Interface* tab.
3. Zoom from the *Main display* field to enter an expression which, at runtime, will evaluate to one of the forms in this task. In this case, form 3 and form 4 are both valid choices.



#	Name	Class	Area
1	Main Program	0	
2	Change display form	0	
3	Display status time only	0	
4	Display date and time	0	



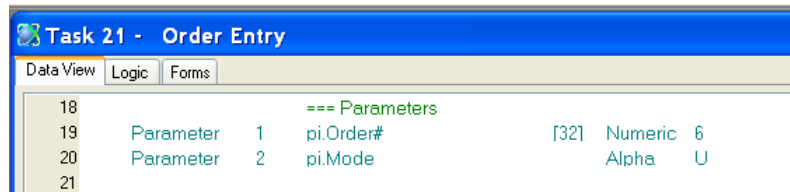
The expression needs to use the format shown here, with the number in single quotes followed by the literal FORM. This signals eDeveloper to keep these numbers in sync if more forms get added to the form list.

Now, in our example, two different forms will be used, depending on which option the user chose.

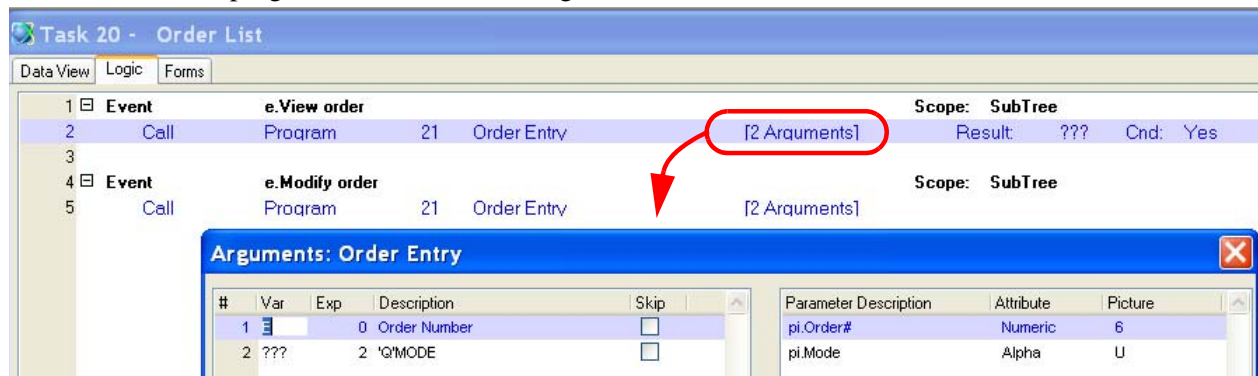


## How do I Synchronize Parameters Between Called Program and Calling Program?

When you define variables as *Parameters* in your data view, eDeveloper will automatically try to synchronize the values between the called and calling program. For instance, suppose we have a task that has three parameters, defined as shown:



When we call this program, and click on the arguments we see:



If the two lists don't match in the Arguments dialog box, then we will get an error message. If one of the parameters is optional and we don't want to send it, we can check the *Skip* option for that parameter, so the lists still match.

Note that you can see the name, attribute and picture of the parameters in the called program.

**Hint:** You may see programs that are called with parameters, but do not have any parameters in their data view section. This is allowed for backward compatibility, but is not a good idea. You should change the virtuals in those programs to parameters, and check all programs that call that program using **Ctrl+F (Edit->Find and Replace->Find Reference)** to make sure the parameters are in fact being passed correctly.

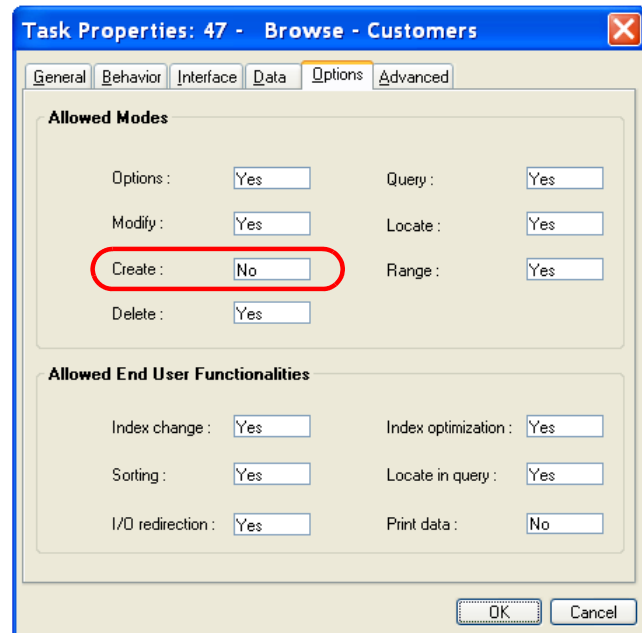
## How do I Prevent the End-user from Adding New Records?

By default, an eDeveloper task allows users to add, delete, and change records.

If you want to prevent the user from adding new records, do the following.

### Preventing the user from entering create mode

1. Go to task properties (**Ctrl+P**).
2. Click on the *Options* tab.
3. In the *Create* field, enter *No*.
4. Alternatively, you can **zoom** (**F5**, double click) from the *Create* field to enter a boolean expression. If the expression is false at runtime, the user will not be able to create a record.



## How do I Prevent the End-user from Deleting Existing Records?

By default, an eDeveloper task allows users to add, delete, and change records.

If you want to prevent the user from deleting records, do the following.

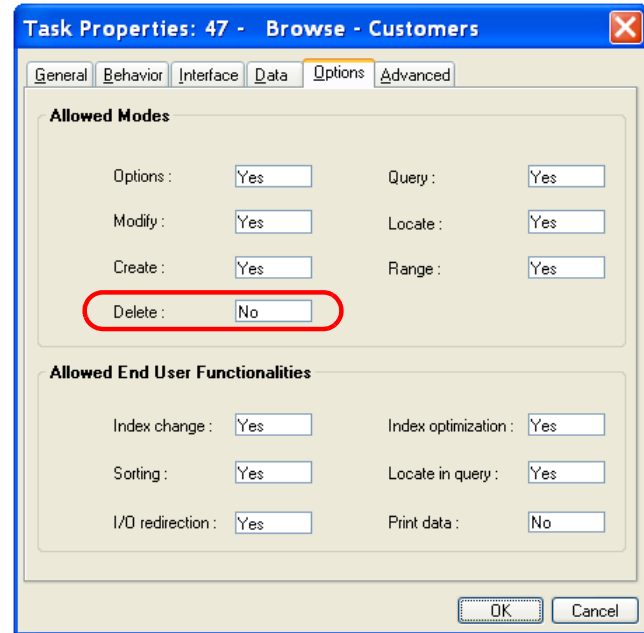
### Preventing the user from entering delete mode

1. Go to task properties (**Ctrl+P**).

2. Click on the *Options* tab.

3. In the *Delete* field, enter *No*.

Alternatively, you can **zoom** (**F5**, double click) from the *Delete* field to enter a boolean expression. If the expression is false at runtime, the user will not be able to delete a record.



## How do I Prevent the End-user from Modifying Existing Records?

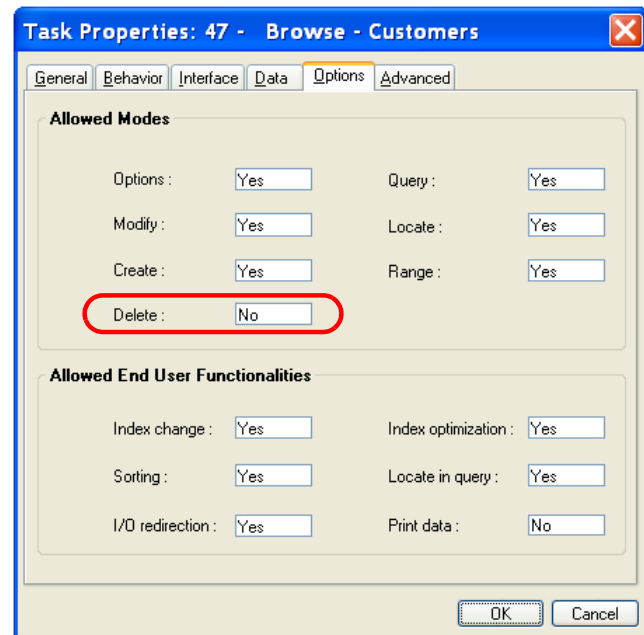
By default, an eDeveloper task allows users to add, delete, and modify records.

If you want to prevent the user from modifying records, do the following.

### Preventing the user from entering modify mode

1. Go to task properties (**Ctrl+P**).
2. Click on the *Options* tab.
3. In the *Modify* field, enter *No*.  
Alternatively, you can **zoom** (**F5**, double click) from the *Modify* field to enter a boolean expression. If the expression is false at runtime, the user will not be able to modify the record.

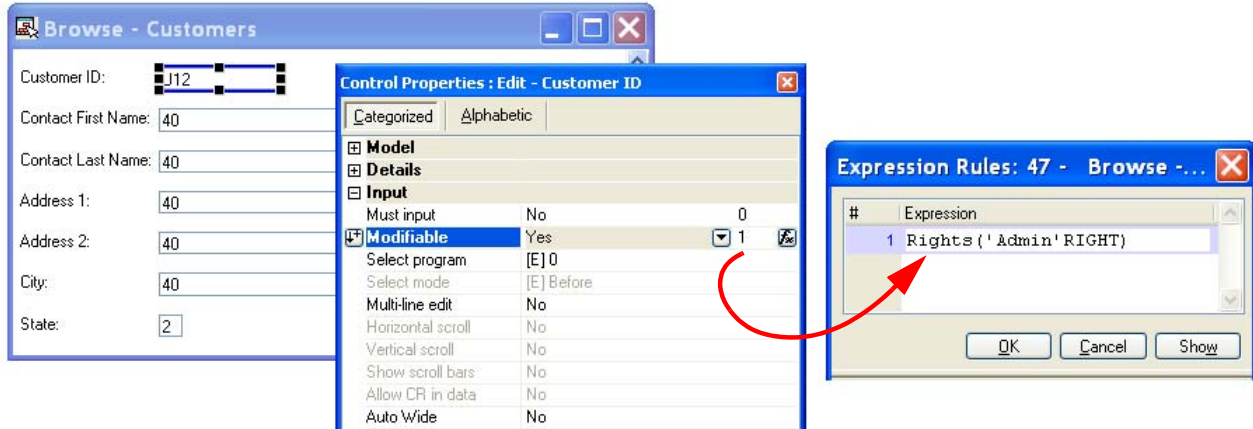
**Note:** This prevents the user from switching from Query mode to Modify mode and so changing records. However, it applies globally, to all records accessed by the task. If you need more control, say on a record-by-record basis, you need to use a different method, either by setting the allowed mode before the record is opened, or by disallowing changes at a field level (such as is described in Chapter 5, “How do I Prevent the End-user from Modifying Existing Records?” on page 104).



## How do I Prevent the End-user from Modifying the Data in Specific Fields?

By default, any data field you put on a form is editable by the user, if the task is in Modify mode. You can, however, prevent the user from changing the data in any field on a field-by-field basis.

### Making a field non-modifiable



1. Position the cursor on the field you want to change, or, select multiple fields to change them all at once (using **Ctrl+Click**).
2. Press **Alt+Enter** to go to *Control Properties* (or just click on the properties pane, if it is open).
3. Go to the Modifiable field and select No if you want the field to never be modified.

Alternatively, you can **zoom** from the expression field to the right (or press the **fx** button) and enter an expression that will evaluate to *true* when the user should be able to modify this field. In this case, we used the Rights function so that the user can only modify the field if they have the “Admin” right.

Now, the user can park on the field, but not modify it unless the expression evaluates to true.

**See also:** Chapter 13, “How do I Enable the End-User to Park on a Control Only by Mouse?” on page 287.

## How do I Create a Selection List Program?

One of the more common types of programs is the selection list type of program. In this type of program, a list of items is presented to the user. The user can scroll up and down or use keystrokes to locate the desired item. Pressing **enter** selects the item that is currently parked on, and “brings it back” to the current field.

These programs are very easy to create in eDeveloper.

### Creating a selection list

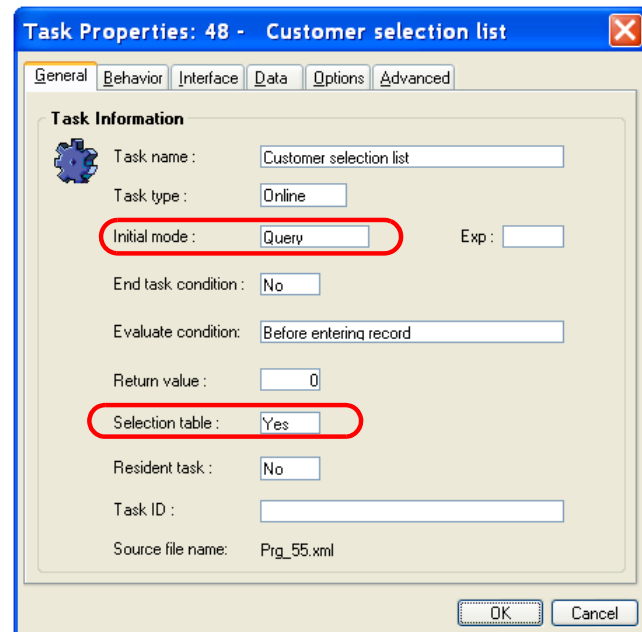
1. In the Program repository, press **F4** to open up a line.
2. Give your program a name. This name is not used by eDeveloper, so you can use any naming convention you like.
3. **Zoom (F5)** on the program name. Because this is a new program, you will see the **Task Properties** dialog box. Select

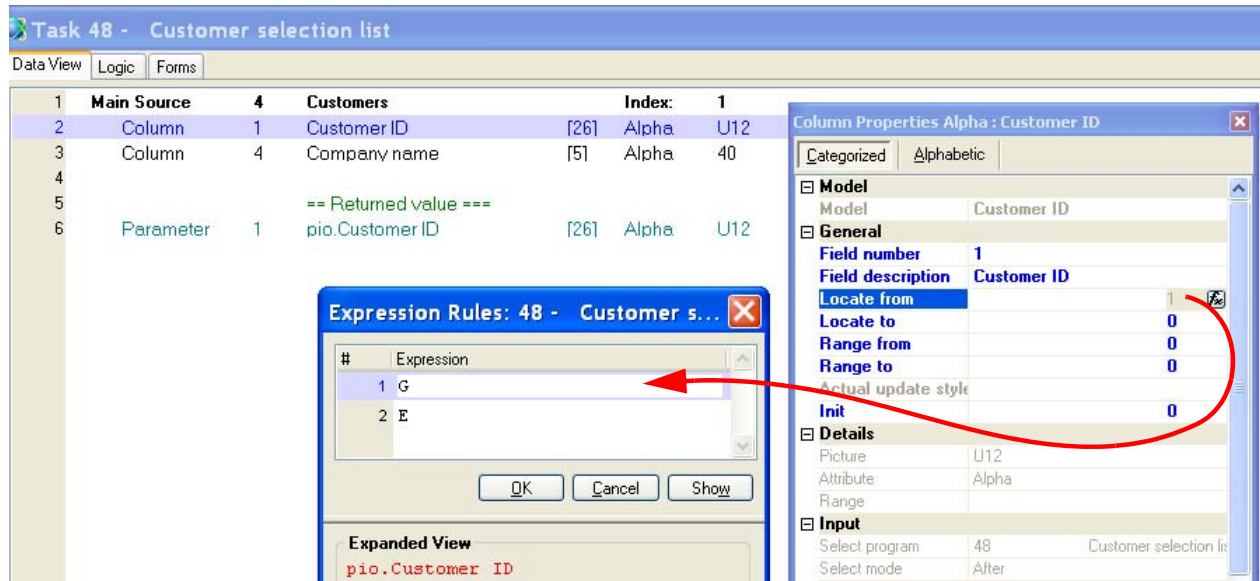
*Initial mode:* Query

*Selection table:* Yes

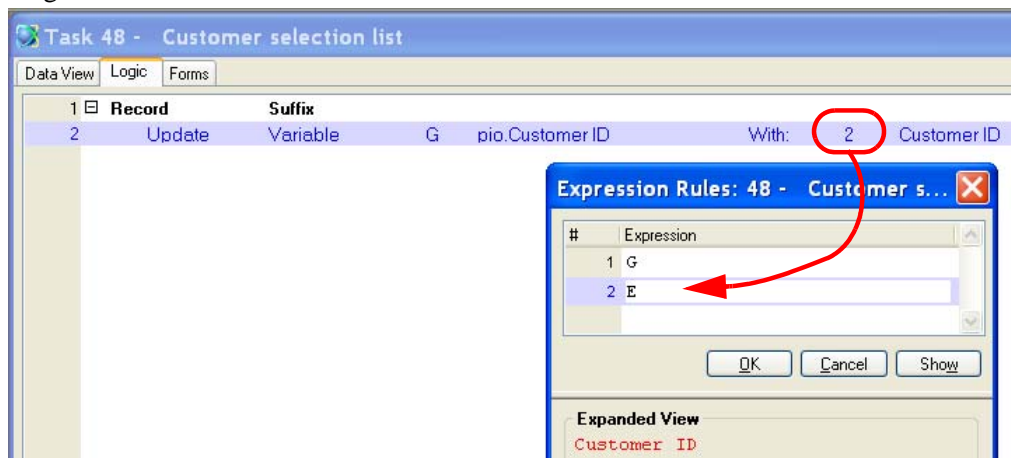
You want query mode because you don't want to be modifying the data accidentally while searching for an item.

The selection table option changes way the engine operates. When Selection table is Yes, then when the user presses Enter, record suffix will execute and the task will end.



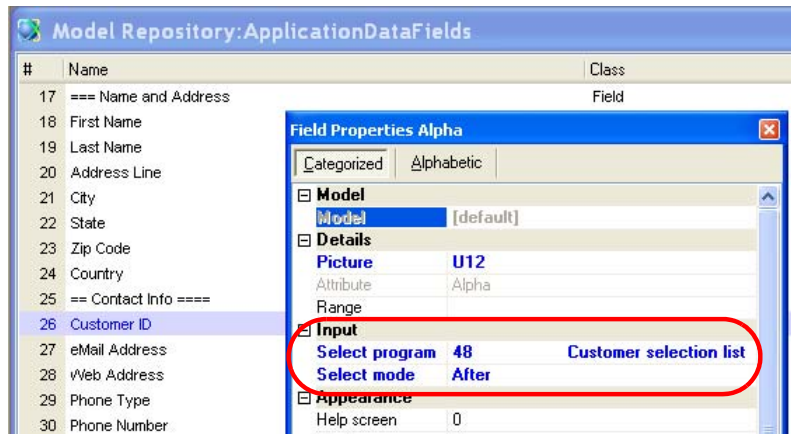


4. In data view, select the data source you want to list. In the data source properties, set access to read so the records won't get locked.
5. Also in the data view, create one parameter. This parameter should use a data model for the value you are sending back. In this example, we are using the Customer ID, and using the Customer ID model (#26). Note that the name of the parameter is similar to the name of the value we are selecting; it's a good idea to name them differently so you don't update the wrong variable. In this case, we used the prefix "pio." to mark the one that is the parameter.
6. Use the parameter as the Locate From property for its counterpart in the table. That is, in this case, we are passing in a Customer ID and we will locate on the Customer ID in the data source.



7. In the Logic Editor, create a Record Suffix logic unit. In that logic unit, update the parameter to the value in the data source. This is where the selected value gets sent back to the calling program.
8. Last, create your form, which will be a simple table control. You can do most of the work by using **Ctrl+G** to generate the form (see Chapter 5, "How do I Automatically Generate a Default Form Layout?" on page 149).

## Using a selection list



The easiest way to use a selection list is to attach it to a model or control, in the Select program property. Notice that there is no explicit parameter passed here. The Customer ID is passed implicitly, whenever the model is used.

**Note:** If you are selecting from only a small set of data, using a combo box that is attached to a table is a good alternative to a selection list.

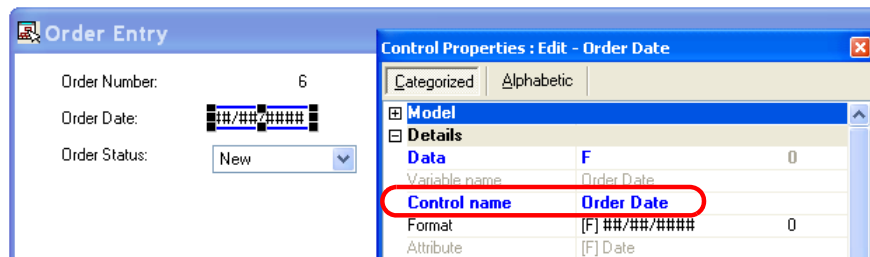


## How do I Properly Validate the Data Entered by the End-user?

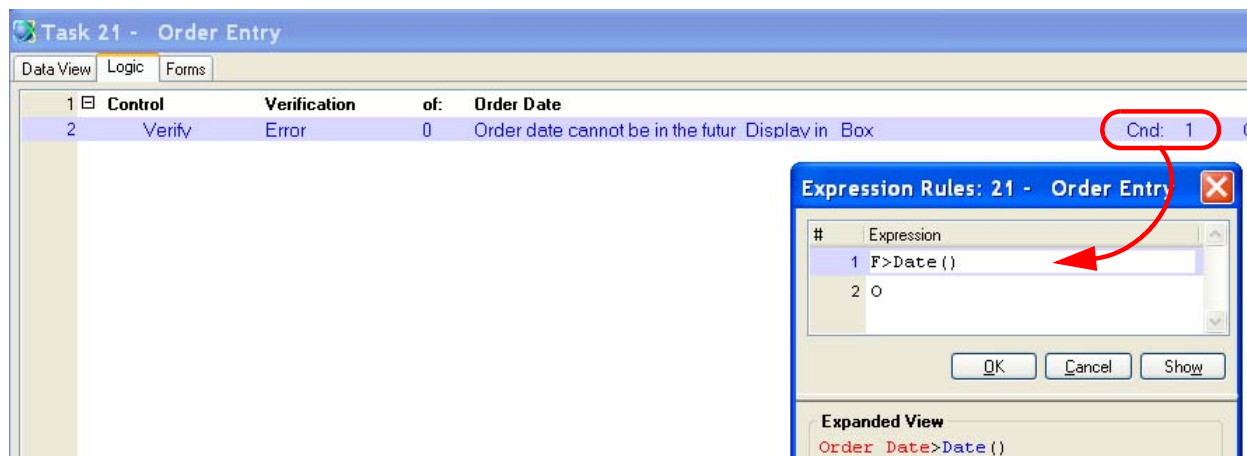
Much of the data validation in eDeveloper can be done when you set up the field model. For instance, if a field is supposed to be a number between 1 and 100, or can only contain 'A', 'B', 'C' or 'X', that can be entered into the model properties and the validation will be done with no further work on your part.

However, if you have more complex validations to do, you need to do these in a Control Verification event.

### Using a control verification event



1. First, make sure the control you are validating has a control name. This one is called *Order Date*.
2. In the Logic Editor, create a control verification logic unit. Select your control as the control to validate. You can zoom to select the control from a list of all controls on your form.



3. Under the control verification event, create a *Verify* operation. The steps to do this are:
  - Press **F4** to open up a line.
  - Type 'E' or select Verify from the pulldown list.
  - Type 'E' or select Error from the next pulldown list. Tab.
  - In this field you can zoom to enter an error message to give the user, using variables, or ...
  - tab again to enter the error message with no variables.
  - After the *Display in* field, select *Box* (the default), unless you want the error message on the prompt line at the bottom of the screen.

- In the *Cnd* column, *zoom* to enter an expression. You want the expression to evaluate to *true* when the data is in *error*.

Now, whenever the order date is in the future, the user will get an error message and cannot move past that field until either 1) the error is fixed (the condition becomes false) or 2) the user cancels all changes to the record (**Ctrl+F2**, or **Edit->Cancel**).

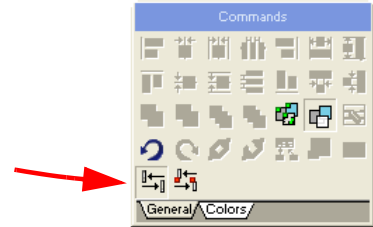
**Hint:** *It is possible to create an error message that is impossible to fix, usually due to improper coding. For instance, if you leave the **Cnd** column hard coded as **Yes**, you cannot stop the error message. If that happens, it is impossible to exit the task. For instance, if you incorrectly enter a Verify Error operation, sometimes you really cannot exit the task. If that happens while you are testing, select Debug->Stop or press the red box on the toolbar.*

## How do I Set the Tabbing Sequence of the Controls on the Form?

By default, the cursor on an eDeveloper form will move about as you expect it to, from top to bottom and left to right. However, you can override the default tabbing sequence if you need to, by following the instructions below.

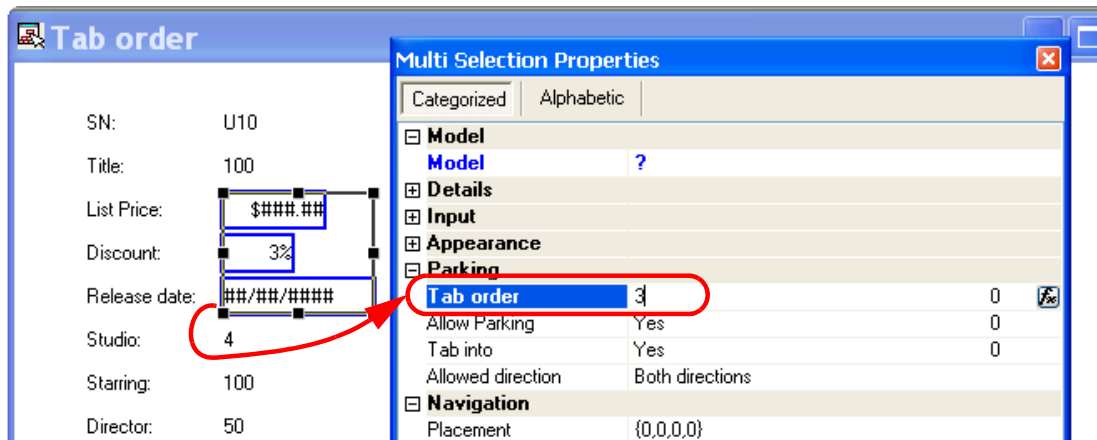
### Setting the tab order

1. Set *Automatic Tab Order* to *off*. It is *on* if the icon is indented, as shown here (it is also on **Drawing->Order->Automatic Tab Order**).
2. Now, in the *Control Properties* of each control, you can enter a number in the property *Tab Order*, or create an expression that will set the tab order at runtime. If the tab order property shows up in grey and you can't change it, go back to step 1.  
(Note: Controls that use an expression rather than a variable will always show in grey, because you can't tab to them.)
3. You can view the tab order by pressing the *Display tab-order* icon, which is right next to the Automatic Tab Order icon on the Commands palette. Items you can tab to are shown in red, others in grey.



When you change the tab order of a control, the form editor keeps the tab order in tight sequence. For example, if you change the tab order '1' to '2', then the form editor will change the existing '2' to '1'.

### Setting the tab order for several controls at once



You can also set the tab order for several controls at once.

1. First, turn Set *Automatic Tab Order* to *off* as explained above.
2. Hold down the **Ctrl** key while clicking on the controls in the desired tab order.
3. Enter the number for the first tab in your selection.

In this example, we selected, in this order: List Price, Discount, and Release date, then entered '3' for the tab order. The tab order would then be: (3) List Price, (4) Discount, and (5) Release date.

**See also:** Chapter 11, "How do I Make the cursor jump to a Specific Control?" on page 257.  
Chapter 11, "How do I Condition an Operation to be Executed only When the End-user Tabs from one Field to Another in a Certain Direction?" on page 260.

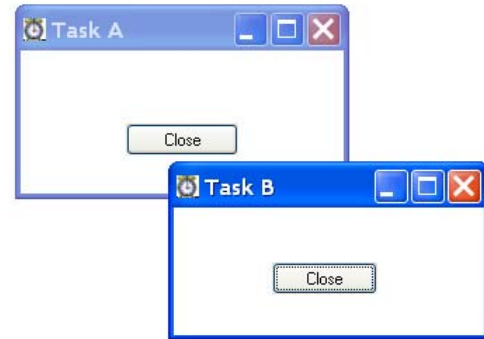
## How do I Exit a Program from a Subtask Level?

Very often, you will have a stack of related tasks running together. Sometimes these tasks even look like one screen, so when the user presses an Exit button, they expect the entire stack of tasks to exit.

The best way to do this is to use your event handler as shown below.

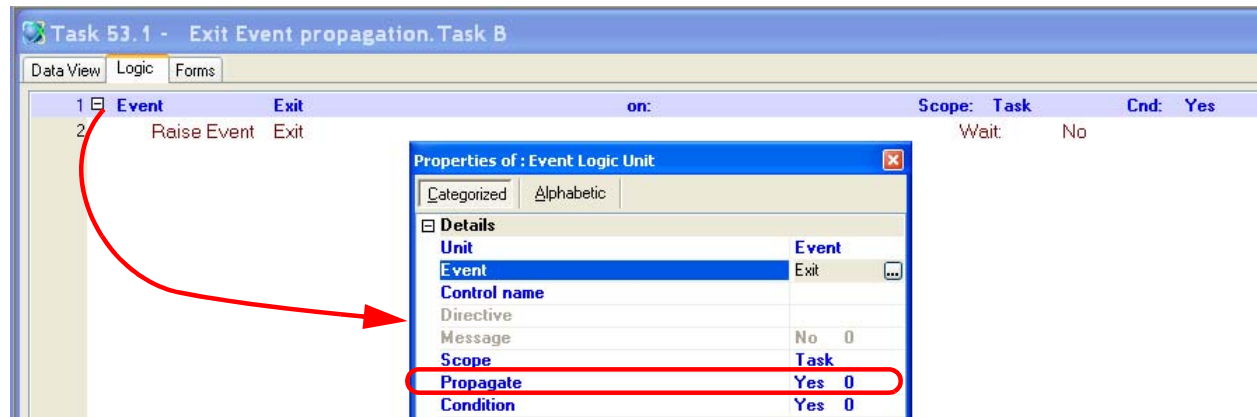
### Allowing an Exit event to propagate

Here we have Task A and Task B. When you click on the Close button in Task A, what normally will happen is that it closes both tasks. But when you close Task B, it will close Task B and control will revert to Task A.



So, how do we make closing Task B also close Task A?

1. First, make the close button raise an **Exit** event. This is the same event that gets raised if the user clicks on the X or presses **Esc**, so we cover all the bases that way.
2. Next, create a handler that is triggered by the **Exit** event. The handler will be triggered by **Exit**, but will also allow it to propagate, which will close Task B.
3. In the handler, raise another event, again, an **Exit** event. This event will then close Task A.



While we showed two separate tasks here for clarity, usually this would be used when Task B is a subform task of Task A.

**See also:** Chapter 8, “How do I Automatically Return Back to the Parent Form by Tabbing Out of the Last Control of the Subform Display?” on page 204.

## How do I Save My Changes in the Task Editor While Remaining in it?

It is always a good idea to save your changes while working on a computer. This is especially true when editing a complex program! That way, if you make a mistake or your system crashes, you can get back to the last saved version.

### Saving changes while editing a program

1. To save changes while editing an eDeveloper program, press **Ctrl+S** (**Options->Save program**).

Now, any changes you have made will be written to disk.

## How do I Create Logic in a Task?

The logic in a task is entered in the Logic Editor. Clicking on the **Logic** tab will get you there.

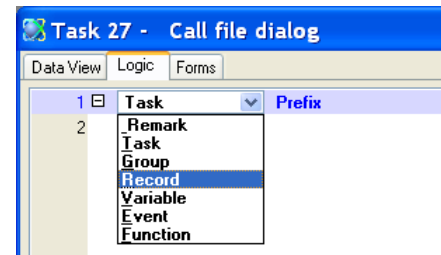
Task logic is held in blocks of code called *logic units*. A logic unit consists of a *header line*, which represents a handler, and zero or more *detail lines*, which represent operations.

The *handler* itself is non-procedural. That is, it is triggered in response to some event, not according to its location in the Logic Editor.

The *operations* within in logic unit, however, are in fact procedural, and execute from top to bottom. These operations are the “code” that executes in your eDeveloper program. They are very powerful, and you will notice that it takes far fewer “lines” to do work than you may be used to in other languages.

### Entering a Header line

1. Inside the Logic Editor, move to the line where you want to create the header.
2. Press **Ctrl+H** to create a header line.
3. Select the type of header you want: Task, Record, Variable, Control, Event, or Function.
4. Continue entering the header properties depending on the type of header.



### Entering an Operation

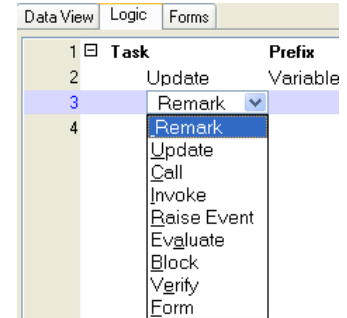
1. Now, inside your logic unit, press **F4** to continue adding operations to your procedural logic. Every operation is entered a little differently, but there is context sensitive help to get you going. The operations are covered in overview in Chapter 5, “How do I Create Operations in a Task?” on page 116.

## How do I Create Operations in a Task?

In each logic unit, there are lines of operations. You create an operation by:

1. Pressing F4 on the line above where you want the operation
2. Selecting the operation by typing in the shortcut letter, or selecting it from the pulldown list.
3. Continuing to fill out the operation according to what it does.

There are 8 operations you can enter. Here is an overview of how to enter each one.

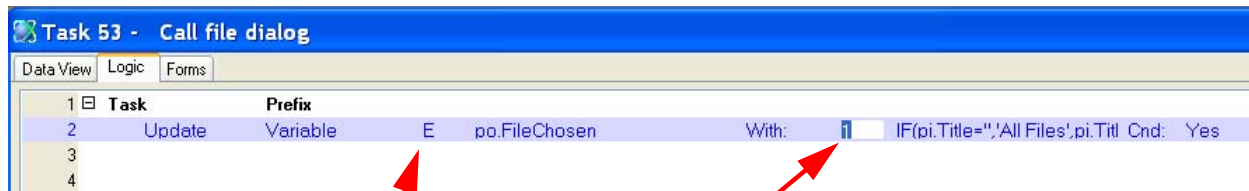


### The Cnd: field

All the operations have a *Cnd* (condition) field at the end. This is basically an “if” statement. If the *Cnd* is “yes” or it points to an expression that evaluates to true at runtime, then that operation will be executed.

- If the *Cnd* is *Yes*, then the operation always executes.
- If the *Cnd* is *No*, then the operation never executes.
- If the *Cnd* is a number, then it points to the expressions rules. To enter a condition expression, just zoom from the *Cnd* field, type in the expression, and press Enter to bring back the expression number.

## Update



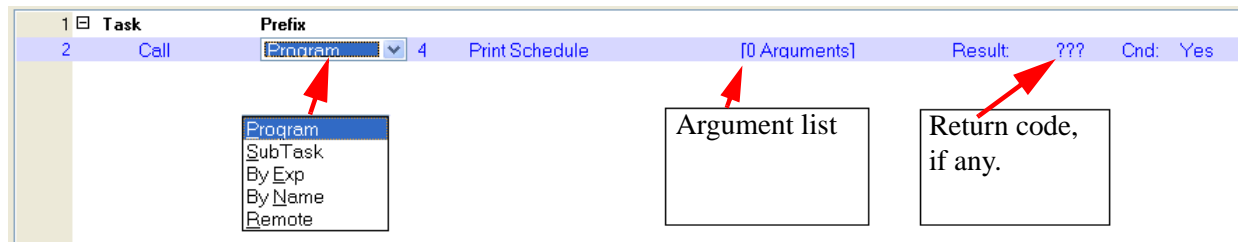
What will be updated. Zoom for a list of variables

What it will be updated to. Zoom to enter an expression.

The Update operation is equivalent to the assignment operator in many languages. It copies the data from an expression into a variable.



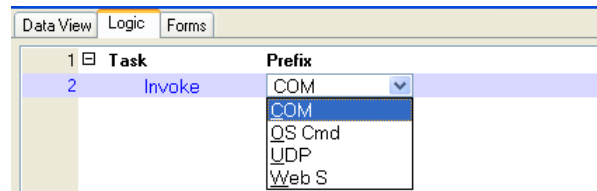
## Call



The *Call* operation is how you execute another eDeveloper task. You have different options, depending on what you want to do. For instance, *Call Program* calls another program in the Program repository, by number. *Call Subtask* calls only subtasks of this program. *Call By Name* calls programs by their public names.

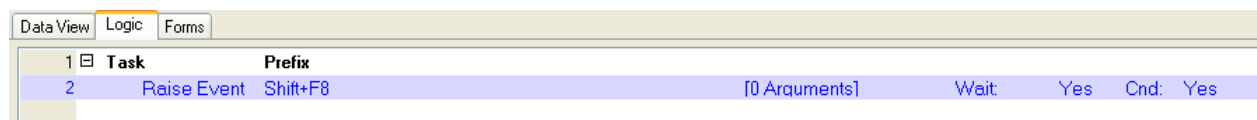
The *Argument* list is used if the called program is expecting parameters. Similarly, the *Result:* field allows you to enter a variable to catch any return value the program sends.

## Invoke



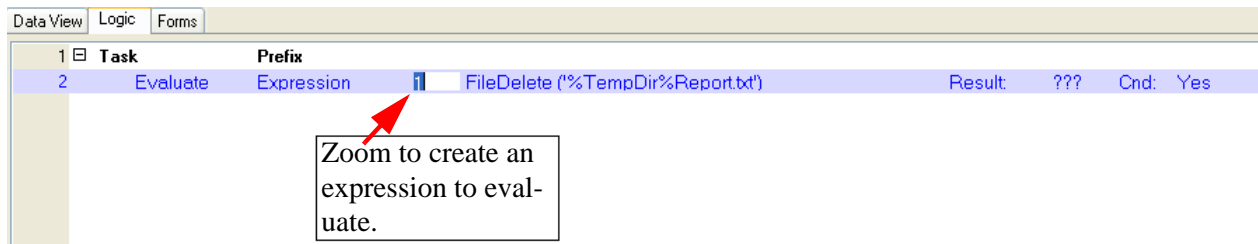
*Invoke* is what you use to call programs that are not in eDeveloper. You can call operating system scripts, web services, ActiveX objects, and more. Each of these is entered a little differently, by changing the settings in the operation properties.

## Raise Event



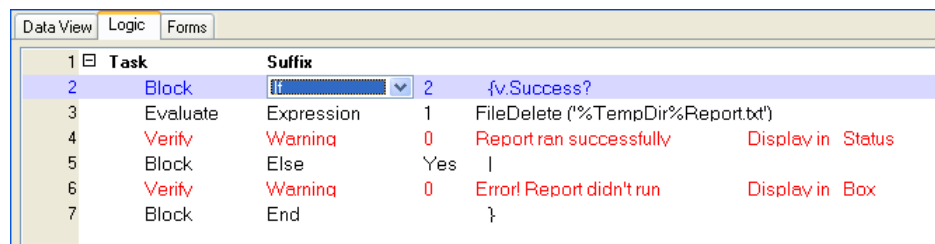
The *Raise Event* operation does just that: it raises an event. The event it raises will be handled in the same way as if the user had pressed a key (system event) or an event was raised by a push button.

## Evaluate



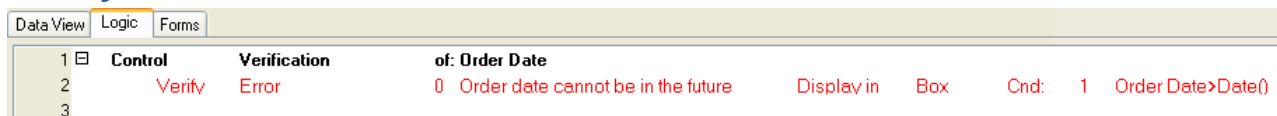
Evaluate is used to execute expressions that may not set any variables. For instance, here we are executing a **FileDelete()**.

## Block



The Block operation is used to group operations, as the block operator does in most computer languages. Here we have a Block if/then/else, based on a return code from a report.

## Verify




There are two kinds of Verify operations:

- *Verify Warning* gives a message to the user.
- *Verify Error* gives a message to the user, but also stops processing from continuing, until the error is fixed.

*Verify Warning* is handy, but *Verify Error* is powerful. Verify Error can stop processing altogether, preventing the user from saving an invalid record. You can learn more about this in Chapter 5, “How do I Properly Validate the Data Entered by the End-user?” on page 109.

## Form

9  Record  
10 Form

Suffix  
Output  Customer Detail

The form to output.  
Zoom to select the  
form.

To:  Report

The IO device to output  
to. Zoom to select from  
a list.

Tasks

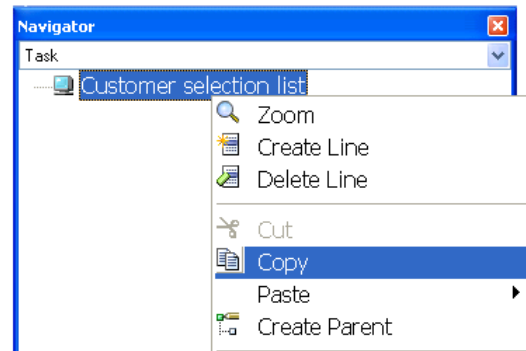
The *Form* operator is used to output a form, or to input a form. Typically if you are outputting a form, it is going to be to a printer, for a report, or to a text file to be read by another program (like Excel), or to an HTML file for internet screens. Input forms are usually used to read text files.

## How do I Copy a Task as a Subtask?

Sometimes you will have a program that is actually only called by one other program. In this case, it is often easier to understand the program flow if you make that program into a subtask. Here is how you do it.

### Copying a program to become a subtask

1. In the *Navigator* pane, select the task you want.
2. On the right-click menu, select **Copy**.
3. Go to the program you want to copy this task into.
4. In the *Navigator* pane, select the task that will be the parent or sibling of the task you are going to copy.
5. On the right-click menu, select **Paste (Ctrl+V)** to paste the task as a child, or *Paste as Sibling* to paste it as a sibling.



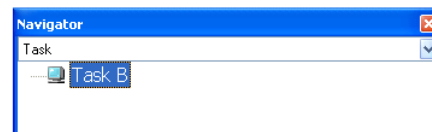
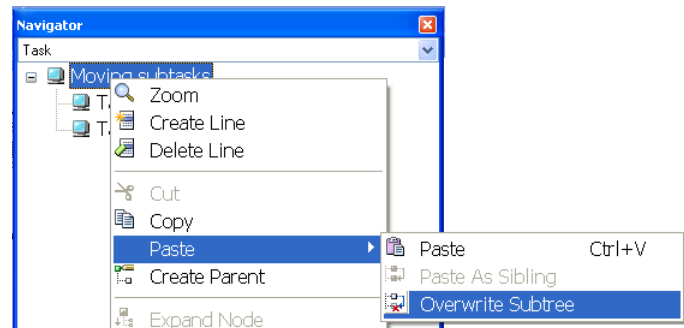
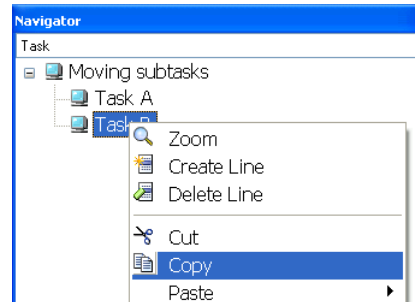
When you are done, the program will be copied as a subtask. You will likely want to do a little cleanup as far as parameters and window positioning, but you will be mostly done.

## How do I Make a Subtask Become the Top Level Task?

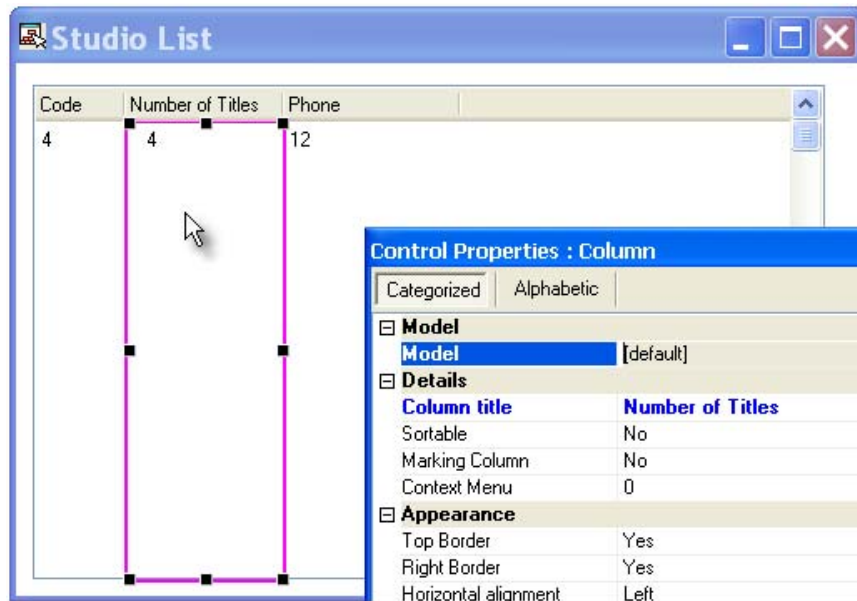
Sometimes you will have a subtask that you want to make into a standalone program. Or, you might have a subtask that has a “dummy” start program that you want to get rid of. In either case, the trick is to move the subtask to the top level. This is easily done, as explained here.

### Moving a subtask to the top level

1. In the Navigator pane, position the cursor on the task you want to move.
2. On the right-click menu, select Copy.
3. Position the cursor on the top level task.
4. On the right-click menu, select **Paste->Overwrite** subtree.
5. Answer Yes to the Confirm Overwrite dialog.
6. Now the lower level task has overwritten the top level task.



## How do I Select a Column of a Table Control?



Normally, when you click on a table control, the entire table is selected. This is because the table is grouped together.

If you want to select only a column on the table control, position the cursor somewhere below the first table line, and press **Alt+Click**. Once you have one column selected, you can use Tab to move from column to column.

**Hint:** You can also select multiple columns by using **Alt+Shift+Click**. This is very useful if you want to change the properties for a number of columns at the same time.

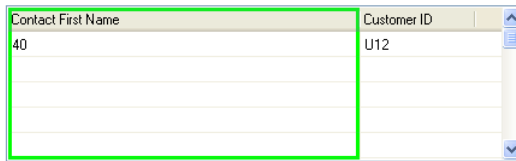
## How do I Drop a Control or Variable on a Table to Create a New Column to the Left of the Highlighted Column?

Normally, when you drop a control or variable on a table, it appears to the right of whatever column you dropped it on. If you want the control or variable to appear to the left, press down the **Shift** key while you drop it.

## How do I Place Several Controls on the Same Column in a Table Control?

Normally, when you place a control on a table control, a new column appears to the right of the column you were over, with the control placed in that column. However, you can cause the control to be dropped in your selected column, by keeping the Alt key held down while dropping the control.

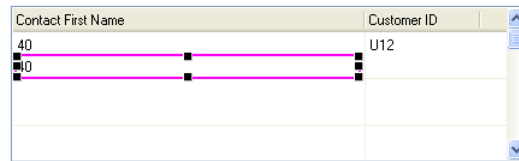
### The effect of dropping a virtual while holding down the Alt key



A screenshot of a table control with two columns: 'Contact First Name' and 'Customer ID'. The 'Contact First Name' column contains the value '40'. The 'Customer ID' column contains the value 'U12'. A green rectangular border highlights the 'Contact First Name' column, indicating it is the selected column.

Contact First Name	Customer ID
40	U12

Before dropping the next field



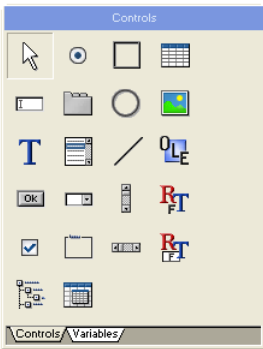
A screenshot of the same table control after a field has been dropped into the 'Contact First Name' column. The 'Contact First Name' column now contains two rows, both with the value '40'. The 'Customer ID' column remains unchanged with the value 'U12'. Two horizontal pink lines with black square handles at their ends are drawn across the table, highlighting the two rows in the 'Contact First Name' column.

Contact First Name	Customer ID
40	U12
40	

After dropping the field.

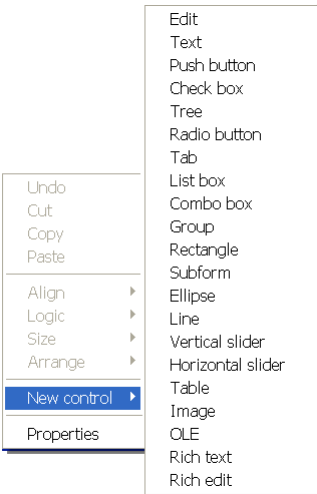


How do I Add New Controls to a Form?



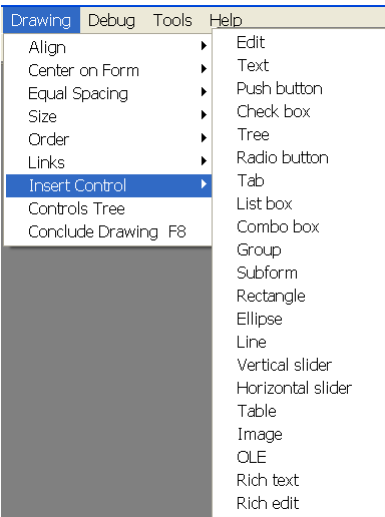
The control palette

To select a control, click on it. Then click on your form where you want it to go.



The context menu

To select a control, click on it. It will then appear on your form.

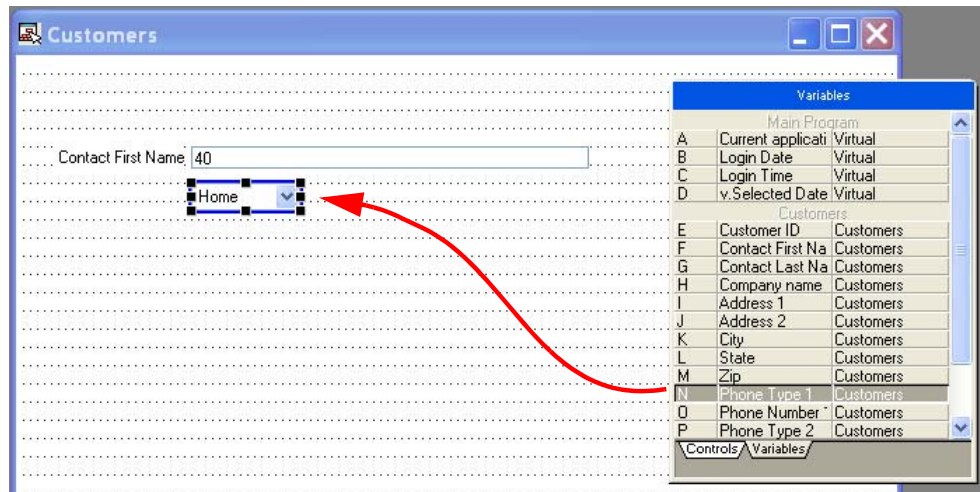


The pulldown menu

To select a control, click on it. It will then appear on your form.

There are three ways to add new controls to your form, as shown above. Each of these will create an “empty” control, that is, one that is not attached to any variable in your program. Some controls, the static controls such as lines, rectangles, and text, are never attached to any variable. For the other controls, you can specify the attached variable in the control properties after it is on the form.

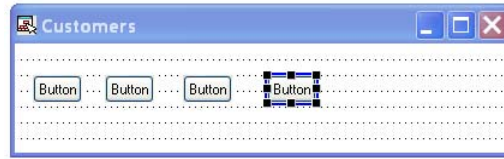
## Using the Variable Palette



The other way you can put controls on your form is to use the *Variables Palette*. Here, you just click on the variable, then click on the form where you want the control. This gives you a control that also is attached to a variable.

When the data was defined (in the field model, data source, or data view) the control style to use was also specified, in the column properties *Style* section. In this case, the model for “Phone Type” uses a combo box, so when we drop variable N, Phone Type 1, it appears in a combo box.

## How do I Drop a Control Multiple Times Consecutively?



Normally, when you drop a control, the cursor changes back to its usual shape, and if you want to select another control, you have to go back to the menu or Control palette.

However, if you want multiple copies of the same control, just hold down the **Ctrl** key while you are dropping the controls.

**Hint:** You can press *Esc* to get the cursor back to normal shape at any time, if you change your mind about dropping a control.

## How do I Select a Container Control (Like Tab or Table) Without Selecting the Controls that are Attached to it?

Normally, when you click on a container control, the control is selected along with all the controls that are attached to it. This is how attached controls work.

However, if you want to select only the container control, not the items attached to it, hold down the **Ctrl** key when you click on the control.

## How do I Keep the Design of a Form for Future Re-use?

Forms are very easy to create in eDeveloper. However, if you want to standardize forms, or save complex forms for re-use, you can do this by using form templates.

### Saving a form as a template

1. Open up the form you want to save as a template.
2. Select **Options->Create Template**.
3. When the file dialog box appears, give the file a name. The default extension is *.mft*.
4. Press **Save**.

### Using a template form

1. Go to the form in which you want to add the template
2. Select *Options->Load template*.
3. When the file dialog box appears, select the template you want to use.
4. Press **Open**.

The template will be loaded on your form. The controls will all be on the form as in the original form. You will still need to attach them to the variables that exist in this task.

**Note:** Another good way to standardize your forms is to use form and control models. These not only allow you to build forms that all look alike, it also gives you the ability to change your standards and have all the existing forms and controls change automatically to meet those standards.

## How do I Select Several Controls at the Same Time?

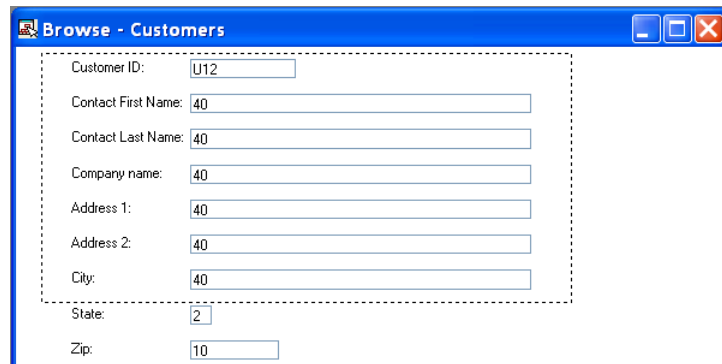
Clicking on a control selects that control. However, sometimes you will want to select several controls at the same time. This is useful for moving the controls as a group. It is also useful for changing shared properties of the controls. For instance, you can select a group of similar controls and attach the same model to all of them, or change the font on all of them.

There are two ways to select a group of controls, using the “rubber band”, and using *Ctrl+Click*.

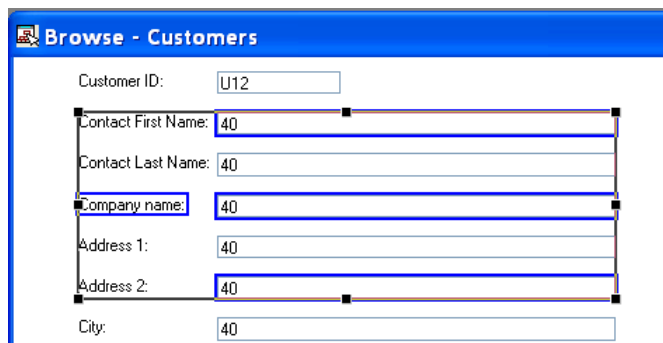
### Selecting controls using a “rubber band”

1. To select controls using the rubber band method, click on a spot outside all the controls you wish to select.
2. Drag the rectangular box around all the controls, then let go of the mouse.

All the controls within the area will be selected. If a control is intersected by the rubber band, it will be selected also.



### Selecting controls using Ctrl+Click



You can also select controls one by one, by using **Ctrl+Click**.

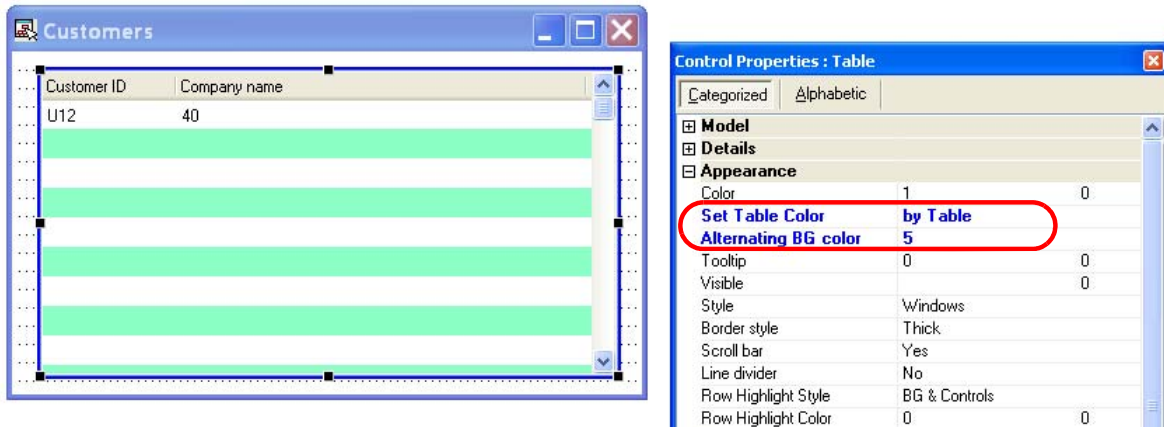
1. Press down the Ctrl key, and click on a control.
2. Still holding the Ctrl key down, click on another control.
3. Repeat as many times as you like.
4. If you want to de-select an item, keep the Ctrl key down, and click on it again.

**Hint:** You can also deselect the controls at any time, by pressing the **spacebar**.

## How do I Show a Table Control with Alternating Colors?

Sometimes, on a large table control, it is hard for the eye to follow the rows. One solution to this has been to use alternating row colors.

### Using alternating colors on a table



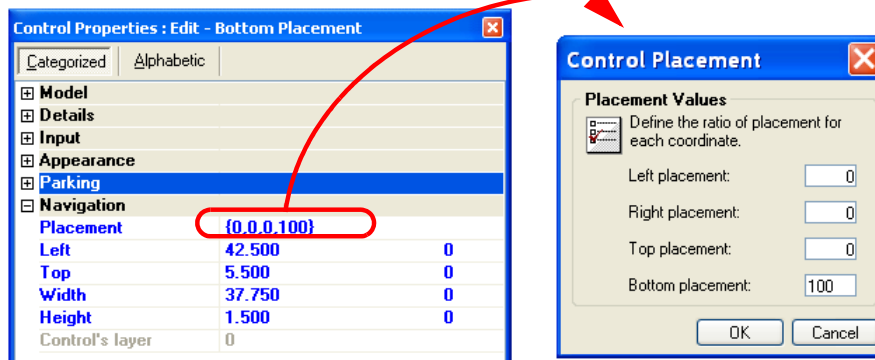
1. Select the table control by pressing **Ctrl+Click**.
2. In the *Control Properties*,
  - Select *Set Table Color by Table*.
  - Select an alternating background color. You can type in the number, or zoom to select it from the color list.

Now the table will show in alternating colors. The base color will be the *Color* property ('1', in this example), and the alternating color will be the *Alternating BG color* ('5', in this example).





## How do I Make Form Controls Fit the Form when it is Resized?



One of the nice things about the GUI environment is that the windows can all be resized by the user. Unfortunately, that makes it difficult sometimes to design a form that looks good for different users.

Placement can help with that. The placement property allows you to specify on each control, that the control will resize itself according to how the form is resized. Entering 100 in one of the placement fields means that the control will resize 100% of the stretching of one of the window's sides; 50 means it will resize half the amount. You can experiment around with the values to get the effect you are looking for.

Using placement on table controls gets particularly interesting. When the bottom placement on a table is 100%, the table adds rows as it resizes. If the right placement is 100%, the columns resize and stretch so more data is visible.

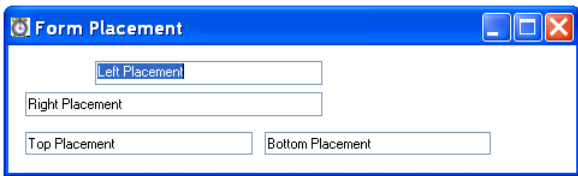
**Hint:** *The controls will never get smaller than their original size. So if you are expecting a form to be resized, create each control at the smallest required size.*

The chart below shows the effect of 100% placement. Each edit field had one placement field set at 100%, and you can see the results when the window is stretched.

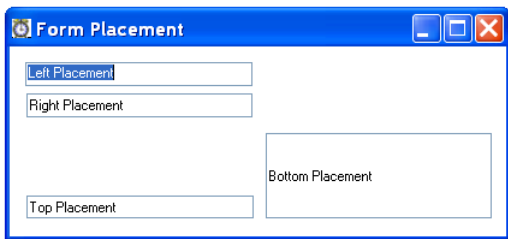
TABLE 0.1. The effect of 100% placement



The original window, unstretched



Stretching to the right. Both 100% Left and 100% Right cause the right hand border to move with right border of the window. But for 100% Left placement, the left border moves also, so the field does not “stretch”.



Stretching to the bottom. Both the 100% top and 100% bottom placements caused the bottom border to stretch with the bottom of the window. But 100% top placement also caused the top to move, so the field does not “stretch”.

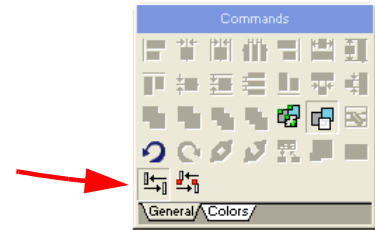
## How do I Change the Tabbing Order of a Control?

By default, the cursor on an eDeveloper form will move about as you expect it to, from top to bottom and left to right. However, you can override the default tabbing sequence if you need to, by following the instructions below.

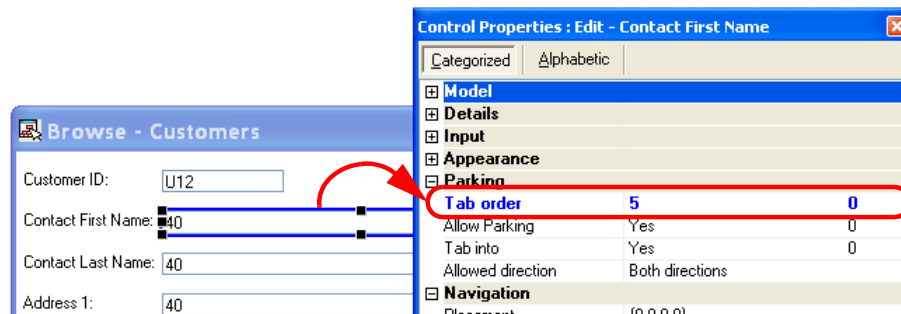
### Setting the tab order

1. Set *Automatic Tab Order* to *off*.

You can tell it is *on* if the icon is indented, as shown here (it is also on **Drawing->Order->Automatic Tab Order**).



2. Now, in the *Control Properties* of each control, you can enter a number in the property *Tab Order*, or create an expression that will set the tab order at runtime. If the tab order property shows up in grey and you can't change it, go back to step 1.



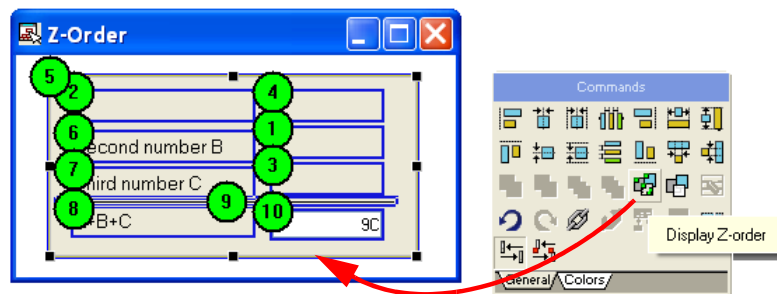
3. You can view the tab order by pressing the Display tab-order icon, which is right next to the Automatic Tab Order icon on the Commands palette. Items you can tab to are shown in red, others in grey.

## How do I Display a Control on Top of Another Control?

When creating GUI screens, there is an issue when one control occupies the same physical space as another, namely, which control will be on top? This is handled in programs by using something called z-order, which refers to the order in which the controls are painted on the screen. Items with a low z-order will be perceived as being “behind” the ones with the higher z-order.


eDeveloper, by default, handles the z-order of the controls automatically, and usually that works fine. There will be occasions where for one reason or the other you need to manually set the z-order.

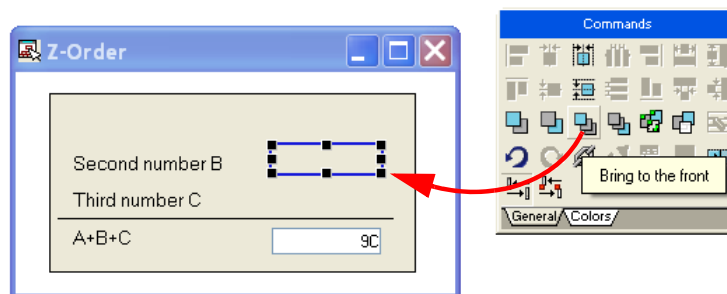
### Viewing the Z-order



In order to see what the current z-order is, you can press the *Display Z-order* icon on the *Commands* palette (**Drawing->Order->Display Z-order**). In this instance, we have several fields that are “behind” the group box. You can see this from their z-order: the group box is a 5, while the disappeared fields are 1-4.

### Manually Using Z-order

1. To fix the z-order, you first need to turn off the Automatic Z-Order feature in eDeveloper. Do this by clicking on the  icon on the commands palette, or selecting **Drawing->Order->Automatic Z-order**.




2. Now, when you select a control, you will see some new options available on the command palette. You can use either *Bring to the Front* or *Bring forward one level* to make your controls visible.

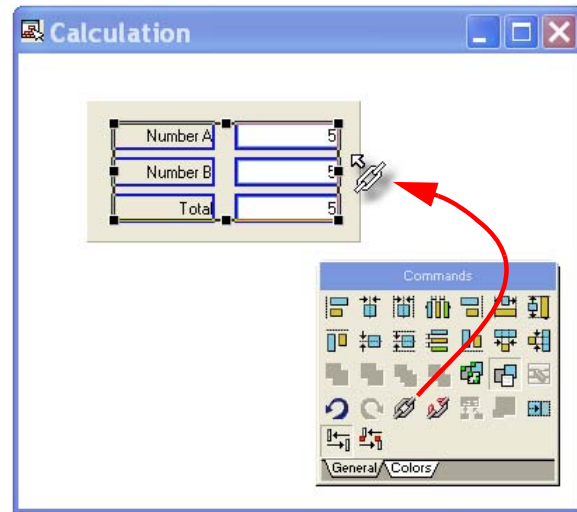
**Hint:** To select controls that you can't see, keep pressing the tab key. You will see the selection box even though you can't see the control itself. Or, you can select **Drawing->Controls Tree** to get a quick view of all the controls on the form.

### Attaching the control

When you are displaying one field as part of another field, for example text that is on a group box, you can also just attach one control onto the other. When you do this, the attached control will always be visible and will not disappear behind the background control, even without dealing with the z-order. To attach one control onto another, do the following:

1. Select the items you want to attach, by holding down the **Ctrl** key while clicking on the items.
2. Click on the link symbol .
3. Click on the background control (a rectangle, in this example).

Now the controls will be attached to the background.



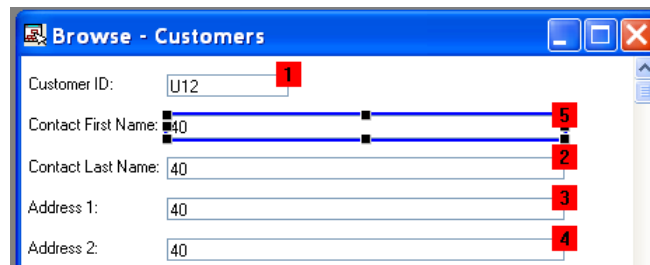
## How do I Jump to the Main Form of the Task?

In GUI programming, you will find yourself working a lot on the main form of the task, which is the part the user sees. You can get to the main form by opening the form editor, then scrolling to the main form, and pressing **F5** to open it.

However, the shortcut way to get to the main form is to just press **Ctrl+M** (**Options->Edit Main Form**). This causes you to jump right into the opened main form, ready for editing.

**Hint:** *To jump out of the main form and back to the task, you can use another shortcut, F8 (Drawing->Conclude Drawing).*


## How do I View the Tab Order of All Controls?



You can view the tab order by clicking on the  icon on the *Commands* palette, or by selecting **Draw->Order->Display Tab order**.

## How do I Undo the Last Modification in the Form Editor?

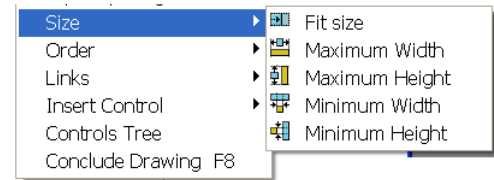
eDeveloper allows you to work fast. Sometimes that means it's easy to make a mistake fast too! Fortunately there is a really good multiple-level Undo function available in the form editor.

To undo your latest modification, just press **Ctrl+Z**, or **Edit->Undo**. You can also click on the  icon on the toolbar or on the commands palette.

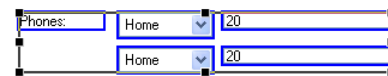


## How do I Make Some Controls Have the Same Height or Width?

To make your controls line up neatly on the form, you often need to make them the same height or width. For instance, if the text prompt for a field is a different height than the field itself, the two will not look even, even if they are aligned at the top or bottom.



The easiest way to make controls have the same size is to use the maximum/minimum commands. These are available on the overhead menu and in the upper right corner of the command menu.



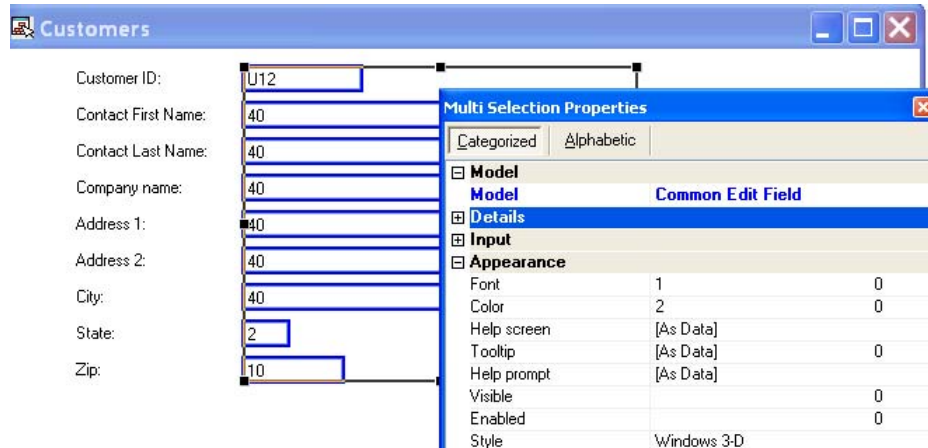
Before



After

## How do I Set a Property on Multiple Controls at the Same Time?

Usually, on a given form you will want most of the controls to look more or less the same, so the controls will likely share the same properties. You can set the properties on each control individually, by clicking on that control and changing the properties in the property pane. But you can save a lot of time by changing the properties for a lot of controls at the same time. Here's how to do it.



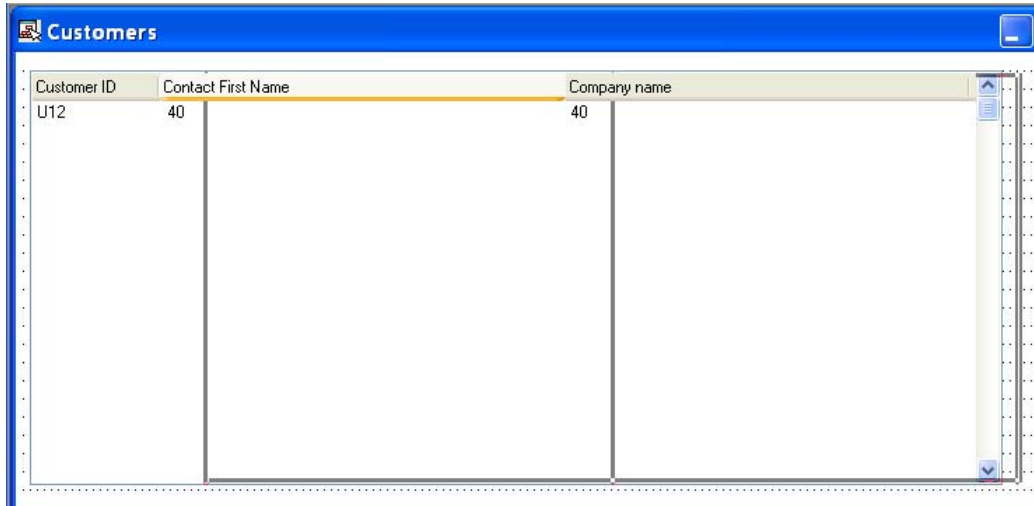
### Changing properties on a group of controls

1. Select the controls you want to change. You can do this by either rubber-banding them, or by using **Ctrl+click**.
2. Go to the properties pane (**Alt+Enter**) and change the property you want to change.

Note that when you select a group of controls, the property pane will change. The header will say "Multi Selection Properties" and the pane will only show the properties that are held in common between all the controls you selected. If the controls you selected were very different, say, some being static controls and some edit controls, then there will not be many properties in common.

**Hint:** This is particularly useful for attaching models to controls, especially on older, inherited programs where control models were never used originally. You can select a column of data, and attach the same model to all the edit fields, or all the text fields, very quickly.

## How do I Change the Width of a Table Control Column?



Customer ID	Contact First Name	Company name
U12	40	40

Normally, when you drag on a table column, it moves the column divider, but the data stays where it is. If you want the entire column to be made wider and move all the other columns off to the right, do the following.

1. Position the cursor on the column divider, up on the table's header area. The cursor will change to a cross shape with arrows on either side.
2. Hold down the **Ctrl** key while you are dragging the column to the right. You will see grey bars appear for each column, as shown above, indicating the movement of the columns.

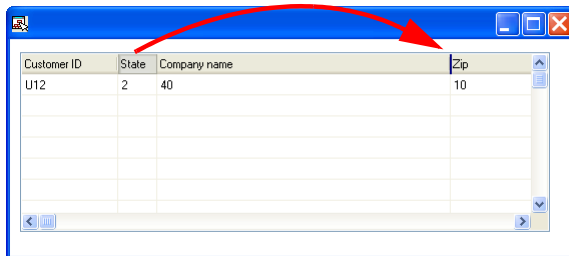
When you are done, the edit fields will be in their proper places in the shifted columns.

## How do I Move a Table Control Column?

If you need to move a table control column from its current position to a new position, that is easily done.

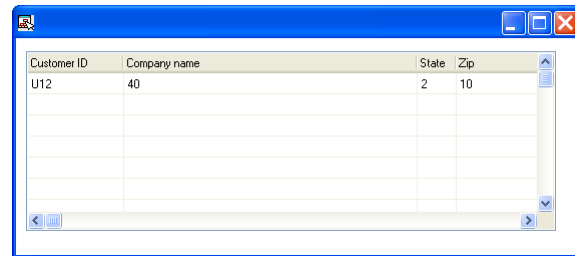
### Moving a Table Control Column

1. Move the cursor onto the header of the column you want to move.
2. Holding down the mouse button, drag the header to the new position. You will see a little black line appear where the column will appear.
3. Release the mouse button.



Customer ID	State	Company name	Zip
U12	2	40	10

Before



Customer ID	Company name	State	Zip
U12	40	2	10

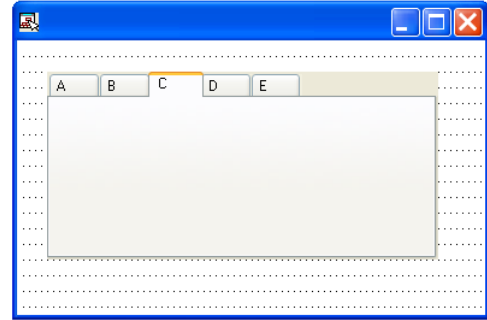
After

## How do I Move Between Tabs While Editing a Tab Control?

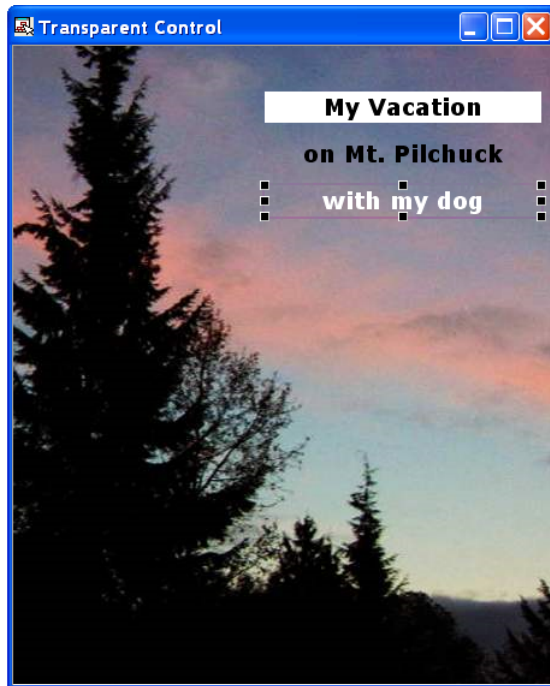
When you are working with a Tab control, clicking on the control selects the entire Tab control, plus any items that are attached to it.

If you want to select just one tab, and see what items are attached to that tab, then you need to do one of the following:

1. Hold down the **Shift** key, then click on the header you want. In this case, we clicked on header “C”.
2. Or, you can select the entire Tab control, then press the **Enter** key. Pressing **Enter** repeatedly will cycle through the tabs.



## How do I Make a Control Transparent?



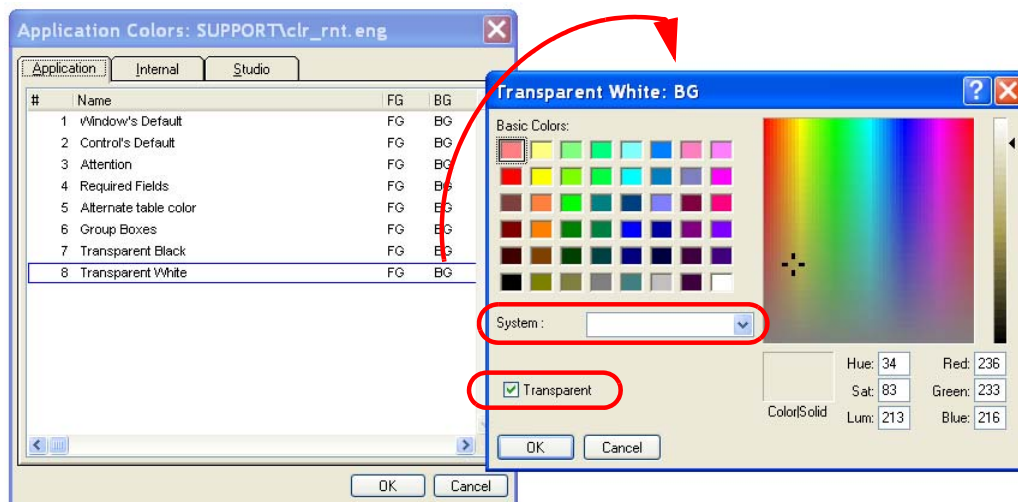
Usually, the background color of a control is opaque, or else it is a 3-D color inherited from Windows. There are circumstances though, where you will want the background color to be transparent, as in this case, where we want the background image to show through.

The first Text control, “My Vacation” is a 2D Text control with a white background. The second two both have transparent backgrounds, and different color foreground texts.

How do we do this? Basically all we do is select a color for the control which has a transparent background color. Below we will show you how to do that.

### Setting a transparent color

1. Go to **Options->Settings->Colors**.
2. Click on the color you want to change, or press **F4** at the end of the color table to create a new color.
3. Zoom on the background color.



4. Set the system color to blank. You do this by selecting the empty line at the top of the drop-down list.
5. Check the *Transparent* box
6. Press **OK**
7. When you get back to the color list, you can set the foreground color the same way, except do not check the *Transparent* box. Instead, select the color you want the text to be (black or white, usually).

8. Press **OK** to exit the Application Colors dialog box.

Now, whenever you use this color on a 2D control, the background will be transparent.

## How do I Set a Default Push Button for the Form?

If you have several buttons on a form, you may want to designate one of them to be the default that is pressed when the user presses **Enter**.

For example, on this screen, we have the Close button selected by default. Although the cursor is positioned on Customer ID when the window opens, press Enter will cause the task to close.

Here is how you do it.

### Setting a Default Push Button

1. Create your push button, and give it a control name.
2. Go to *Form Properties* (*Alt+Enter* when no control is selected).
3. In the Input section, go to the Default Button entry.
4. Zoom from the first field to select your button control name.
5. Or, zoom from the expression area to the right to create an expression that, at runtime, evaluates to a valid button control name.

Now, when you run your program, the button you chose will be the default button.

Form Properties GUI Display -		
Categorized   Alphabetic		
Model		
Details		
SDI		
Input		
Title bar	Yes	0
System menu	Yes	
Minimize button	Yes	
Maximize button	Yes	
Average palette	No	
Default Button	Close Button	0
Appearance		
Split		
Navigation		

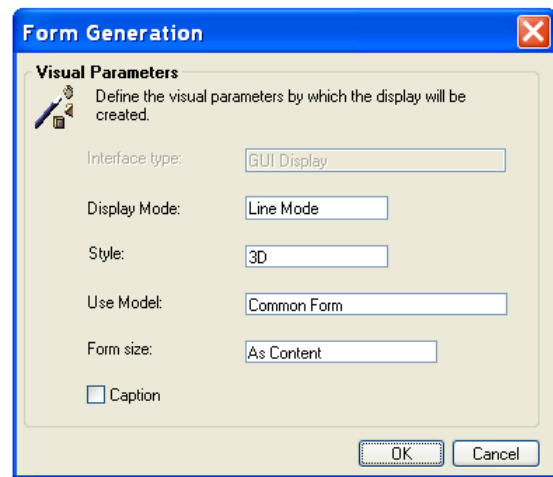


## How do I Automatically Generate a Default Form Layout?

When you have your data view all selected, and the next step is to put the variables on the screen, there is a quick shortcut you can use to get started. It's called the Form Generator, and it will put all the variables in your data view on the main form for you.

### Using the Form Generator

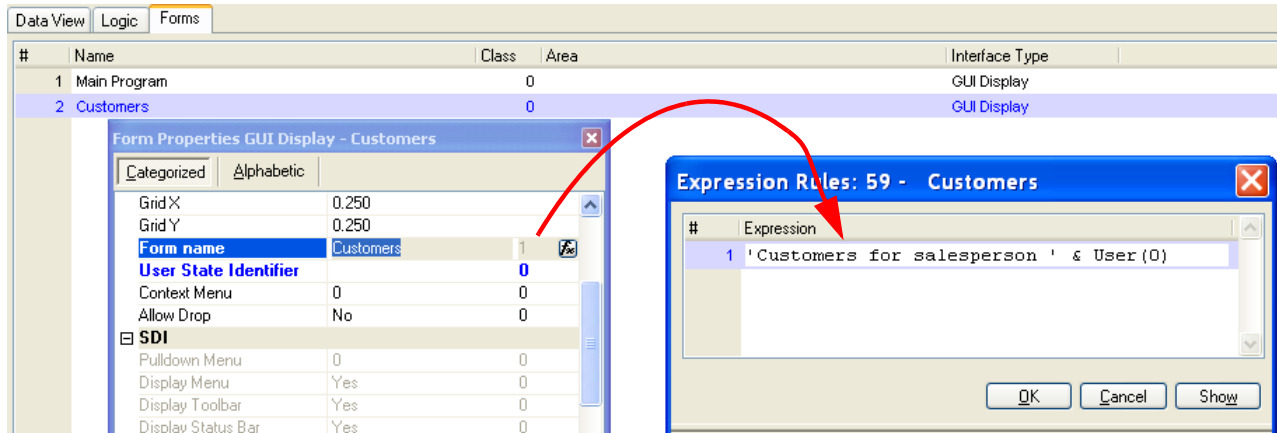
1. Open the task for which you need the form built.
2. Click on the *Forms* tab.
3. Press **Ctrl+G** (**Options->Generate Form**).
4. You will get a dialog box that says "Overwrite Current Display". Press the Yes button.
5. The Form Generation dialog box will appear. Fill out the options according to what you want.
  - *Display Mode*: Select Line Mode if you want the data in a table control, Screen Mode otherwise.
  - *Style*: 3D or 2D controls.
  - *Use Model*: If you select your model here, that will be used as the form model.
  - *Form Size*: to fit the content, to fit the whole screen (MDI), or to fit the model.
6. Press OK. Your form will be populated with all the items in your data view.



**Hint:** Be sure you press **Ctrl+G** from the forms list. If you press it while in the data view editor or Logic Editor, you will start the Program Generator rather than the Form Generator, and then you will overwrite your entire program.

## How do I Show a Dynamic Form Title?

Windows that have a title bar typically have text written across the title. By default, eDeveloper shows the form name on the title bar. However, you can change the title bar dynamically, at runtime, using variables in an expression.



### Using an expression in a form title

1. Go to the form editor.
2. Open up the form properties (**Alt+Enter**).
3. Go to the *Form Name* property.
4. Zoom on the expression column (or press the *fx* button).
5. Create an expression that will evaluate to the title you want. In this case the title will read 'Customers for salesperson' and the userid.

Now, this expression will show at runtime in the title bar. Notice that the old form name is still there: it will show while we are working on the form in the Studio.

**Hint:** This is a nice technique to use even for static titles, when the title is too long for the form name field.

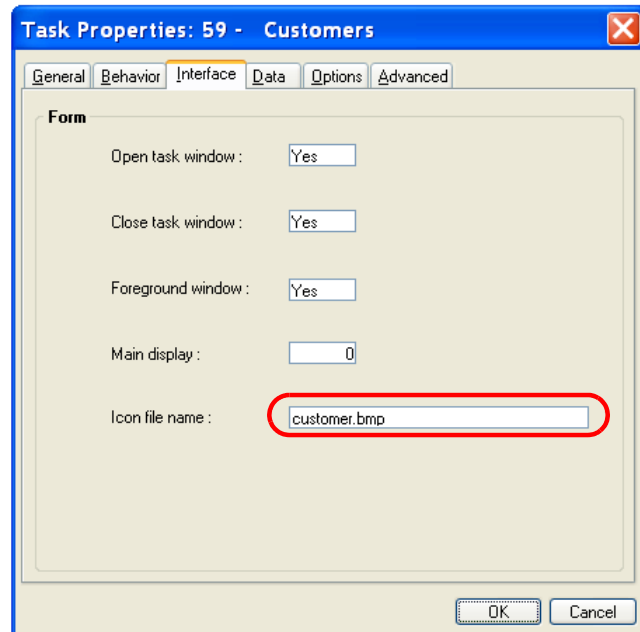
## How do I Set an Icon for a Form?

If you want, you can change the icon that shows up in the corner of any particular form. This icon will show on the upper left corner of the form, and also when the form is minimized. This overrides whatever icon you chose at the project level.

### Choosing a form icon

1. Go to Task properties (*Ctrl+P*).
2. Select the *Interface* tab.
3. Zoom from the *Icon file name* field to find the icon file you want to display.

**Hint:** You don't want to have a hard-coded path name here, because when your application is installed, your users won't have the same setup you do. If you code no path name, eDeveloper will look for the icon file in the directory where the .edp file is located (**%WorkingDir%**). But you can also use your own logical name to indicate where the icon file will be at runtime, or use a relative path.



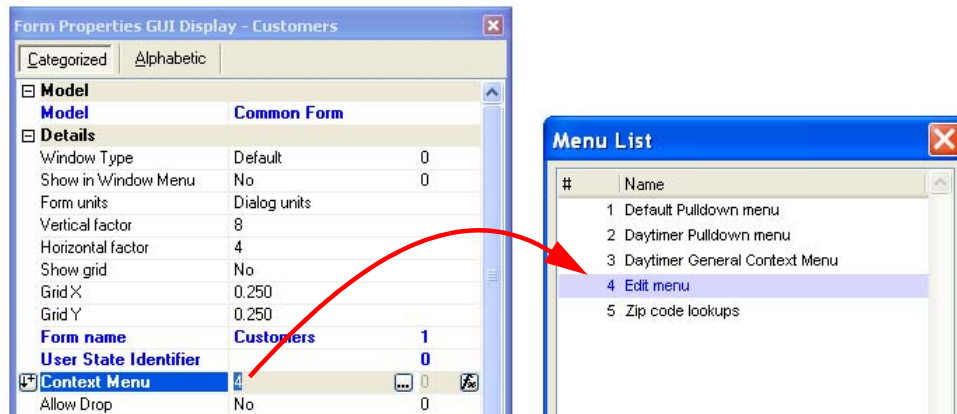
## How do I Set a Default Context Menu For All Controls of a Form?

You can create a context menu for the entire application in the menu repository. However, you can override that and create a default menu for each form also.

### Setting the default context menu for a form

**Prerequisite:** The menu you want to select must already exist.

1. Go to the form properties (*Alt+Enter* when no control is selected).

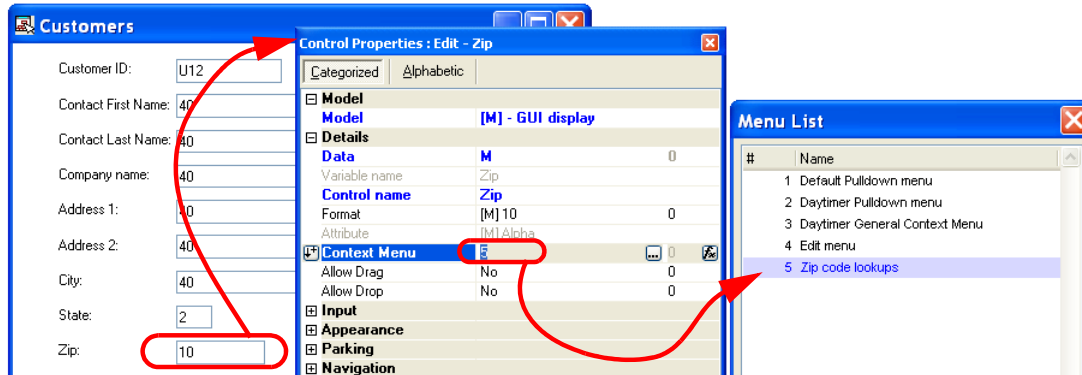


2. Zoom from the *Context Menu* field and select the menu you want to use.
3. Alternatively, you can enter an expression that will evaluate to the context menu number at runtime, by zooming from the field at the right (or clicking on the *fx* button).

Now, when the user presses the right mouse button, the context menu will appear no matter where they are on that form.

## How do I Set a Context Menu for an Individual Control?

You can create a context menu for the entire application in the menu repository, and you can override that and create a default menu for each form. You can also create a context menu at the field level. In this example, we create a special context menu to look up zip codes.



### Creating a field level context menu

**Prerequisite:** The menu you want to select must already exist.

1. Go to the control properties (*Alt+Enter* when the control is selected).
2. Zoom from the *Context Menu* field and select the menu you want to use.
3. Alternatively, you can enter an expression that will evaluate to the context menu number at runtime, by zooming from the field at the right (or clicking on the *fx* button).

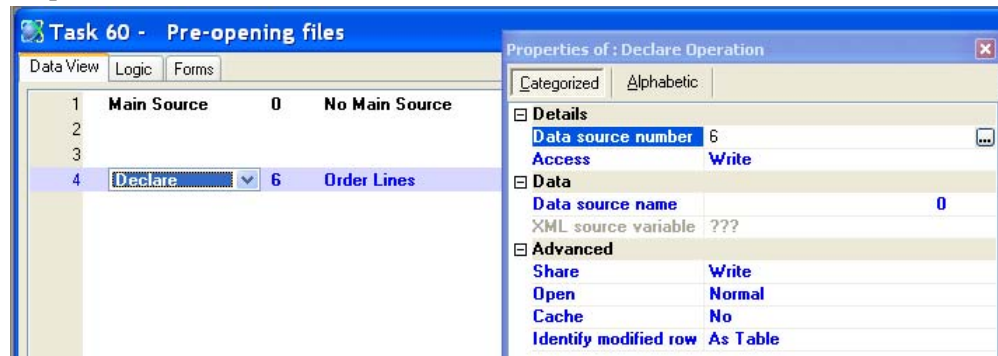
Now, when the user presses the right mouse button, the context menu will appear when the cursor is positioned on that field.

## How do I Improve Performance When a Task is Being Called Repeatedly by a Batch Task?

Sometimes, when one batch task is called repeatedly by another batch task, performance can be very slow. This often turns out to be due to the overhead required to open and close the files used by the subtask. There are a few simple steps you can take to improve performance in these cases.

### Improving performance of batch subtasks

1. The first and most important step is to make sure that the files used by the subtask are pre-opened in the parent task. Different file systems have different amounts of overhead for a file open, but it often takes longer to open a file than to read the records.

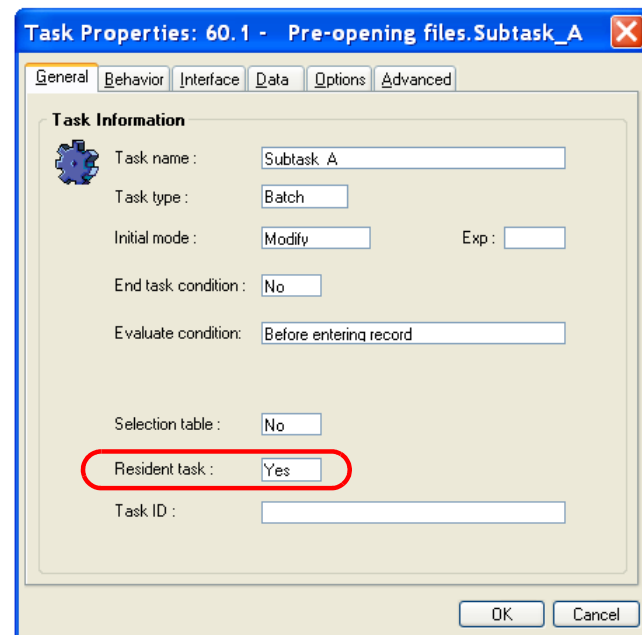


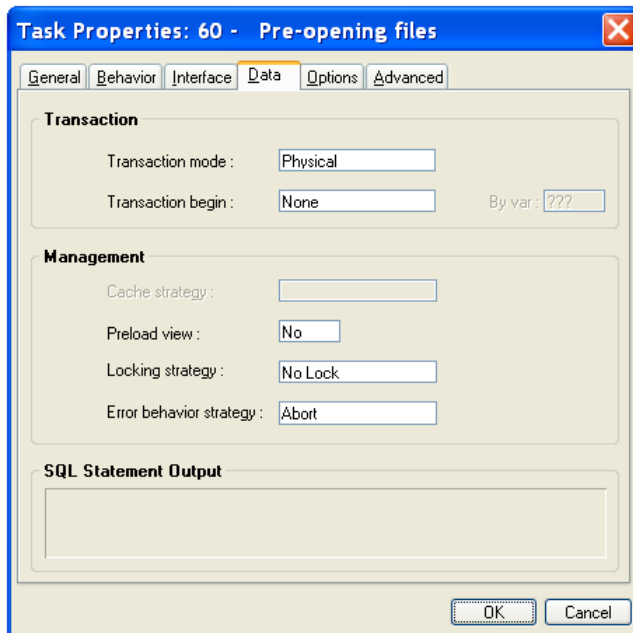
If the file is not being used in the parent task, you can pre-open it by inserting a declare line in the data view. The *Access* used in this line must match the *Access* used when the subtask opens the file.

2. Secondly, make sure the batch task is loaded as resident. This keeps the batch task from being reloaded every time it is called.

To make the subtask load as resident:

- Go to *Task Properties (Ctrl+P)*.
- Go to the *General* tab.
- Set *Resident Task* to *Yes*.





3. Last, make sure that the transaction mode in the parent task is not deferred. When the transaction mode is deferred, all the subtask transactions “stack up” and can seriously degrade performance. To check the transaction mode:

- Go to the parent task.
- Go to *Task Properties* (**Ctrl+P**).
- Go to the *Data* tab.
- Set the transaction mode to what you need for the task. In this case we turned off transactions totally. Since many large batch tasks are for reports, which do not update data we care about, there is no need for transaction processing.





---

## Chapter 6: Extended Logic

---

### How do I Send an eMail?

eDeveloper has some built in email functions to send and receive eMail.

To send an eMail, there are two steps:

1. Connecting to the server, and
2. Sending the mail

We'll cover these individually.

#### Connecting to the server

You connect to the server using the **MailConnect()** function. In order to do this, you need to know the name of your mail server. If you don't know what that is, you can probably find it in your eMail software's setup options. The **MailConnect()** syntax is:

```
MailConnect ( type, server, user, password )
```

where:

- **Type** stands for the server type. 1=SMTP, 2=POP3, 3=IMAP
- **Server** is the address of the server
- **Userid** is the user's id
- **Password** is the user's password

For an SMTP server named juno.olympus.com, the connection string would be:

```
MailConnect (1, 'juno.olympus.com', '', '')
```

The return code depends on what kind of server you are connecting to. You can get the details in the Mail-Connect help entry.

## Sending the eMail

Once you are connected to the server, you can send the mail using the **MailSendQ** function. The syntax of that function is:

```
MailSend (From, To, Cc, Bcc, Subject, Message, Attachment)
```

Where:

*From* is the From email address. This is not checked, so it can basically be anything.

*To* is the To email address. If this is incorrect, the mail will not arrive.

*Cc* is a list of email address to CC

*Bcc* is a list of email address to Bcc

*Subject* is the text that shows on the subject line

*Message* is the text of the message itself

*Attachment* is a string representing a file to send as an attachment

So one example might be:

```
MailSend ('jenny@frigates.com', 'fred@aaawidgets.com',  
'', '', 'May Invoice', 'Attached is your  
May invoice', 'F:\Invoices\aaa_11_2007.pdf')
```

Of course, you will probably use variables, not hard-coded text. This is a very straightforward syntax. You can string multiple addresses together, separated by commas. You can also string together multiple attachments.

The return code is numeric, where zero means success. If the return code is not zero, you can use the return code as a parameter to the **MailErrorQ** function to show a user-friendly error message. If the return code is in the variable BP, then

```
'Send return code: '&Str (BP,'5NC')&' '&MailError (BP)
```

will give a structured message with the error code number and message.

## How do I Add an Attachment to an eMail I Send?

When you send an eMail using eDeveloper functions, you will use the **MailSend()** function. The syntax of that function is:

```
MailSend (From, To, Cc, Bcc, Subject, Message, Attachment)
```

To send an attachment, you only need to specify the *Attachment* parameter of the MailSend function.

You can send multiple attachments by stringing the file names together, separated by commas.

So one example might be:

```
MailSend ('jenny@frigates.com', 'fred@aaaawidgets.com',  
'', '', 'May Invoice', 'Attached is your  
May invoice', 'F:\Invoices\aaa_11_2007.pdf')
```

In this instance, the file 'F:\Invoices\aaa\_11\_2007.pdf' is the attachment.

**See also:** Chapter 6, “How do I Send an eMail?” on page 157.

## How do I Receive An Email?

When you receive email, there are several steps.

1. To receive mail, you need to use a POP3 or IMAP connection (rather than the SMTP connection that is used for sending). This connection will probably require a userid and password, so your login might be something like:

```
MailConnect (3, 'juno.olympus.com', 'FredZ', 'rabbit16')
```

Here the '3' is the connection type, to connect to an IMAP server, and the other three parameters are the server name, user name, and password.

2. The return code, if positive, will indicate the number of emails in the queue. Save that number!
3. Now, you need to cycle through those messages. You will cycle until the index is equal to the number of messages. For each message, you can use the various functions to fetch the different pieces of the email. For instance, suppose Y is the index. Then you could fetch:

```
MailMsgDate(Y)  
MailMsgFrom(Y)  
MailMsgSubj(Y)  
MailMsgText(Y)  
MailMsgFiles(Y)
```

You can save all these in your own email system, if you want.

When you have successfully read a message, you can use

```
MailMsgDel(Y)
```

If you don't delete the messages, they will stay in your mailbox. (Keeping them in the mailbox is a great idea when you are testing).

4. And when you are finished, you can use

```
MailDisconnect(2, 'TRUE' LOG)
```

to disconnect your session and delete mail from the server. The '2' is the disconnect type, to disconnect from the mail "receive" server. The 'TRUE' LOG boolean is a parameter that forces the deletion of mail on the server.

### A note on server types

There are two types of servers that are used to download mail, and they work slightly differently. When you connect to a POP3 mail server, only the new messages are received. But when you connect to an IMAP server, you will receive all existing messages, both the new ones and the ones that were previously downloaded.

**See also:** Detailed information on each of these functions is found in the eDeveloper Help system.

## How do I Handle eMail Attachments?

When you receive your emails, one of the functions is

```
MailMsgFile(Y)
```

Where Y is the mail index. This returns the number of attachments on this particular email.

Then, you need to cycle through each of the attachments using

```
MailMsgFile(Y,Z)
```

where Y is the email index, and Z is the attachment index. This will give you the actual file system name of the attachment, as it is stored on your computer.

## How do I Retrieve a Web Page or other URL Content?

You can retrieve a copy of a web page, or any other URL content, by its URL using the **HTTPGet()** function.



### HTTPGet()

1. Use **HTTPGet()** to update a Blob. The syntax is:

```
HTTPGet(URL, Arg1, Arg2, ...)
```

Where URL is the web site's URL, and *Argx* are additional request header information.

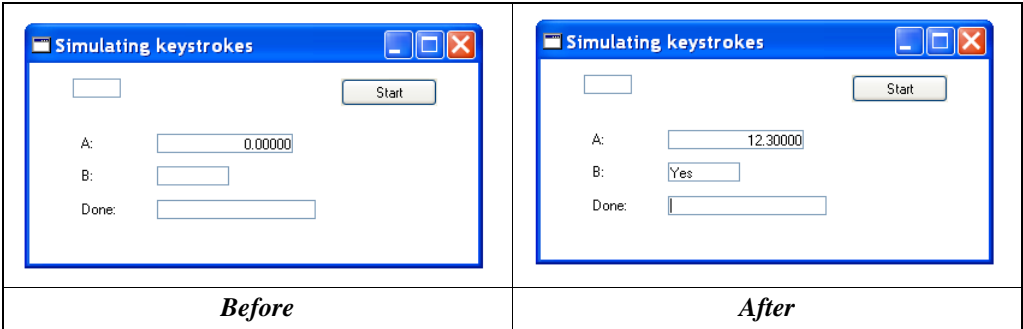
2. You can save the BLOB as a file using **Blb2file()**.
3. If the BLOB is a file ending in **.htm**, you can pass the filename as an OS Cmd and a browser will be called to view the HTML as a web page.

**See also:** **HTTPGet()** function in Magic Help

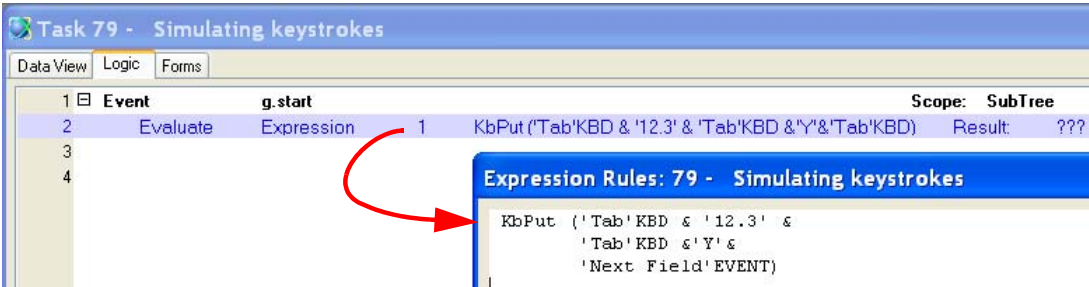
# How do I Simulate a Keystroke?

There are two ways to simulate a keystroke in eDeveloper, using a **KBPut()** function, or using a *Raise Event* operation. **KBPut()** is a more generic function, and it allows you to insert characters and create long macro streams. Raise Event, on the other hand, allows you more flexibility, since you can not only simulate keyboard events, you can also raise other kinds of events.

Here we will cover the **KBPut()** function.



## KbPut()



The syntax of the **KbPut()** function is:

`KbPut (String)`

Where String is a string of formatted characters as shown below. The concatenation operator (&) is used to attach them all together.

**TABLE 0.2.** Kbput Characters

What	Syntax	Example	Quick way to enter them
letters, numbers	Enter text in quotes	'123.2'	



TABLE 0.2. Kbput Characters

What	Syntax	Example	Quick way to enter them
Keystrokes	name followed by KBD	‘Tab’KBD	Select them from the <b>Shortcut</b> right-click menu
Events	name followed by EVENT	‘Next Field’EVENT	Select them from the Internal <b>Events</b> right-click menu

You can see that ‘Tab’KBD and ‘Next Field’EVENT are basically equivalent in this case. It is usually safer to use the EVENT coding, because the keyboard may be remapped.

**Note:** **KBPut()** only works in window that is pure eDeveloper. It does not affect ActiveX controls or called Windows dialogs.

**See also:** Chapter 4, “How do I Work with the eDeveloper Engine as an Event-Driven Engine?” on page 55.

## How do I Let The End-user Mark Several Records In a Table And To Handle The Marked Records Collection?



Sometimes it is helpful to allow a user to select several items from a list. eDeveloper has a built-in facility to do this, in the multi mark functions.

### Preparing the table for multi marking

The properties of the table have to be set up before multi-marking is available.

1. In the table properties, set *Multi marking* to Yes.
2. Select at least one column to be the marking column. Usually this is the column furthest to the right, but you can, if you wish, allow marking on all columns. For the marking column, set the *Marking Column* property to Yes.

If the table Style is *Windows 3D* or *Windows*, then the marking column will show up as 3D raised.

Now, the user can select multiple records from this table.

## Handling marked records



Once the records are selected, you need to process them. Typically this would mean writing the records out to another table for further processing, but in our example we just give the user a message.

1. Create an event that will be raised when it is time to handle the marked records. In our example, we use a push button that raises the event “ge.Start”.
2. In the handler for the event, process the record as if it were just the current record. In our example, the user will get three messages, one for each record selected. This is true even though there is no block loop involved.

## Other processing using MM functions

1. Within the event, you can use **MMStop()** to exit the processing before all the marked records are handled. When **MMStop()** is executed, the engine will park on the record that was being processed when the **MMStop()** was executed.
2. You can execute code after all the records are processed by using a block with the expression `MMCurr()=MMCount(0)`

**MMCurr()** will be the number of the record currently being processed, while **MMCount(0)** is the total number of records.

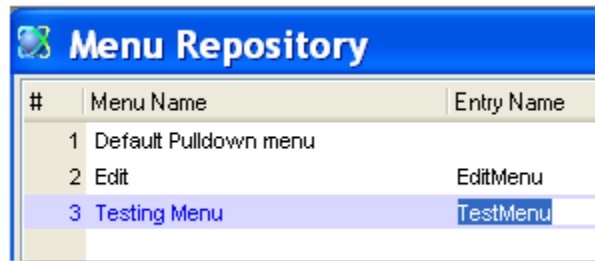
In our example we use this test to clear the multi marking after the last record is processed.

3. **MMClear()** unmarks all marked records.

## How do I Manipulate the Menu Entries to Become Invisible, Disabled or Checked?

In eDeveloper, you have complete control over the end-user menus. You can create your own overhead and context menus, and you can have the context menus be context sensitive to the form or field. But you can also enable and disable entries at runtime. In this section we show you how.

### The menu entry name



#	Menu Name	Entry Name
1	Default Pulldown menu	
2	Edit	EditMenu
3	Testing Menu	TestMenu

In order to enable and disable individual menu entries, you have to give each entry a unique name. This is done in the menu repository. For each menu item you want to work with, type in a unique text name. This name does not show to the user, so it can be any text you like. In this example, we have two Entry Names, “EditMenu” and “TestMenu”.

Once you have that taken care of, you can work with the **MnuCheck()** and **MnuEnable()** functions to check and uncheck, enable and disable, your menu entries at runtime.

### Menu Paths

The submenus of the main menus also have Entry Names, and these can be strung together to give a menu path. For instance, under the default pulldown menu, we have a menu named UtilityMenu, and under that, one named SetupMenu. The path is:

UtilityMenu\SetupMenu

Which we will use in the examples for **MnuAdd()**.

### The Menu number

In addition to the menu name, the menus are also referred to by their menu number literal. In our example above, the Edit menu is **'2'MENU**, and the Testing Menu is **'3'MENU**. We will use those literals in our **MnuAdd()** example.

## MnuCheck()

Task 8 - Menu Functions							
Data View		Logic		Forms			
9	Event	MnuCheck				Scope: SubTree	
10	Evaluate	Expression	4	MnuCheck('SetupMenu',v.Check?)			
11	Update	Variable	F	v.Check?		With: 5	NOT v.Check?
12							

**MnuCheck()** checks and unchecks a menu entry. The syntax is:

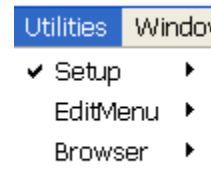
MnuCheck ( *EntryName* , *boolean* )

Where:

**EntryName** is the text in the Entry Name column of the menu. This can be any text you like. It does not show to the end user.

**Boolean**: if true, this checks the menu. If false, this unchecks the menu.

In our example, after the expression executes, the “SetupMenu” menu will be checked.



## MnuEnabl()

Task 8 - Menu Functions							
Data View		Logic		Forms			
22	Event	MnuEnabl					Scope: SubTree
23	Evaluate	Expression	10	MnuEnabl('SetupMenu',v.Enable?)			
24	Update	Variable	H	v.Enable?	With:	7	NOT v.Enable?
25							

**MnuEnabl()** enables and disables a menu entry.

The syntax is:

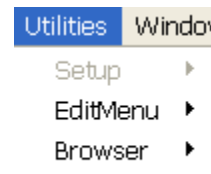
MnuEnabl ( *EntryName* , *boolean* )

Where:

**EntryName** is the text in the Entry Name column of the menu. This can be any text you like. It does not show to the end user.

**Boolean**: if true, this enables the menu. If false, this disables the menu.

In our example, after the expression executes, the “SetupMenu” menu will be disabled.



## MnuShow()

Task 8 - Menu Functions							
Data View		Logic		Forms			
14	Event	MnuShow				Scope: SubTree	
15	Evaluate	Expression	8	MnuShow('SetupMenu',v.Show?)			
16	Update	Variable	G	v.Show? With: 6 NOT v.Show?			

**MnuShow()** allows you to make an entire menu appear or disappear. The syntax is:

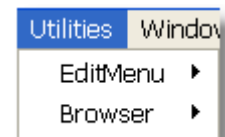
MnuShow(*EntryName*, *boolean*)

Where:

**EntryName** is the text in the Entry Name column of the menu. This can be any text you like. It does not show to the end user.

**Boolean**: if true, the menu shows If false, this menu disappears.

In our example, after the expression executes, the “SetupMenu” menu will not show to the user, though it still exists.



## MnuAdd()

Task 8 - Menu Functions							
Data View		Logic		Forms			
1	Event	MnuAdd				Scope: SubTree	
2	Evaluate	Expression	2	MnuAdd('2'MENU,'UtilityMenu\SetupMenu')			
3							

**MnuAdd()** allows you to add entire menu (including the sub-menus) to a menu. You do this by creating your menu in the Menu repository, then referring to it with a menu number literal.

In addition, we use a Menu Path to specify where the new menu will be located.

The syntax is:

MnuAdd(*MenuEntry*, *MenuPath*)

Where:

**MenuEntry** is the menu literal: in this case '2'MENU to refer to the second menu.

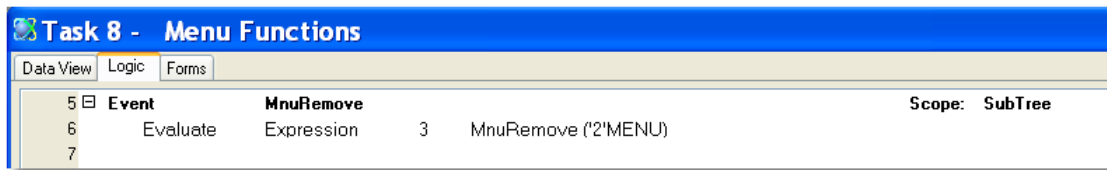
**MenuPath**: refers to where the new menu will end up. In our example:

'UtilityMenu\SetupMenu'



will locate the new menu right under the SetupMenu.

## MnuRemove



**MnuRemove()** is the opposite of **MnuAdd()**. It deletes a menu entry. It is not required that you specify the exact path: if you do not, the entry will simply disappear, wherever it was entered.

The syntax is:

```
MnuRemove (MenuEntry, MenuPath)
```

Where:

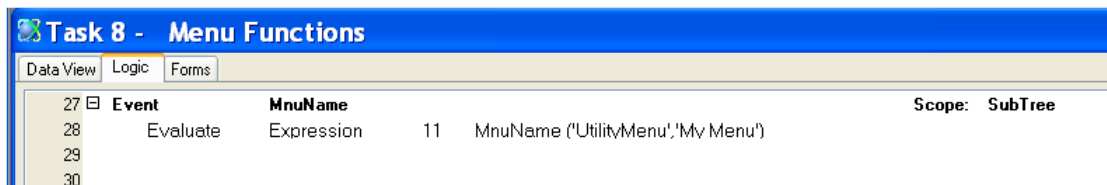
**MenuEntry** is the menu literal: in this case '2'MENU to refer to the second menu.

**MenuPath**: refers to where the new menu will end up. In our example we did not use a menu path, because we only added the MenuEntry in one place (which is the usual case).

## MnuReset()

**MnuReset()** allows you to reset the menu to the default. There are no parameters; it just clears every menu alteration you may have made.

## MnuName()



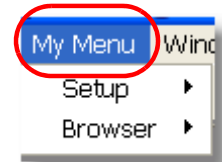
**MnuName()** allows you to rename a menu entry. This does not effect how it works; it just effects how it looks to the user. The syntax is:

```
MnuName (EntryName, EntryText)
```

Where:

**EntryName** is the text in the Entry Name column of the menu. This can be any text you like. It does not show to the end user.

**EntryText**: refers to the text that shows up to the user. In our example it was 'My Menu', and as you can see, the menu entry was renamed.



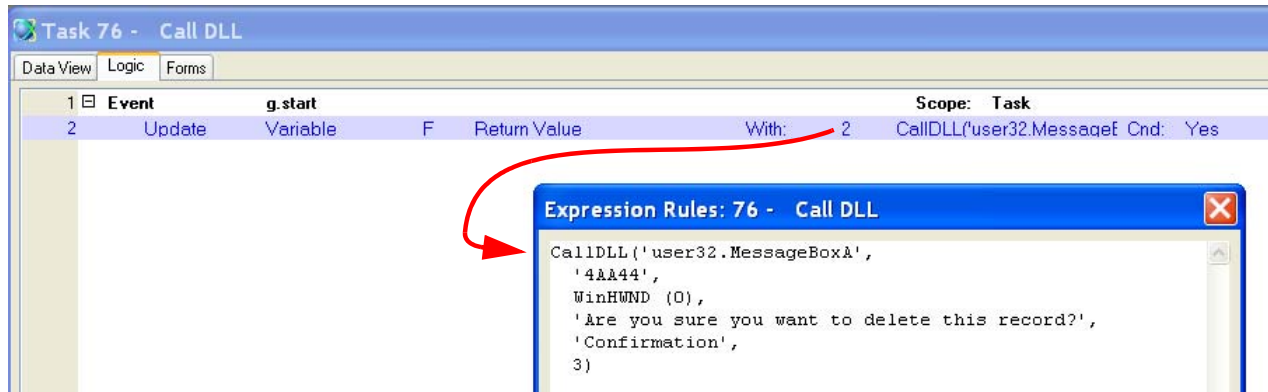


## How do I Call a DLL Function?

There are a lot of DLLs available in the Windows environment, in packages you may purchase, and you may even want to write some of your own. Every DLL call is different, because most of the work is matching your parameter string to what the DLL expects, and to figure that out you need the documentation for the DLL. We'll go through a simple example here though, for calling the windows API to show a message box.

In eDeveloper, you can call a DLL using the **CallDLL()** function, or by using the *Invoke UDP* operation. We will see how to use both here.

### Using CallDLL



1. Press *F4* to open up a line where you want to make the call.
2. Press *U* to create an *Update* operation, or select the operation from the pull-down list.
3. Select the variable you want to use for the return value. In this case, our DLL returns a numeric value, so we are using a numeric for the return code. The number happens to represent the user's response.
4. Zoom from the With: field to enter your DLL call. Your DLL call will use the **CallDLL()** function.

The syntax is:

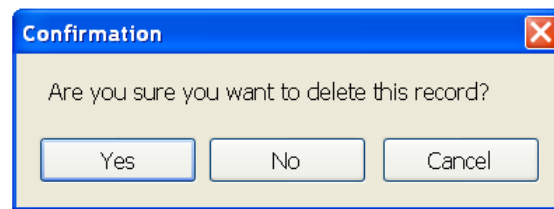
```
CALLDLL(DLLName, ArgString, Arg1, Arg2 . . . )
```

Where

- **DLLName** is the name of the DLL. In this case, it is user32.MessageBoxA.
- **ArgString** is a string of characters, where each character stands for the data type of the argument. There is one character for each argument, plus the last one, which is for the return value.
- **Argx** are the arguments being passed. In this case there are four: the Window handle, two text strings, and a number representing which style of box to use.

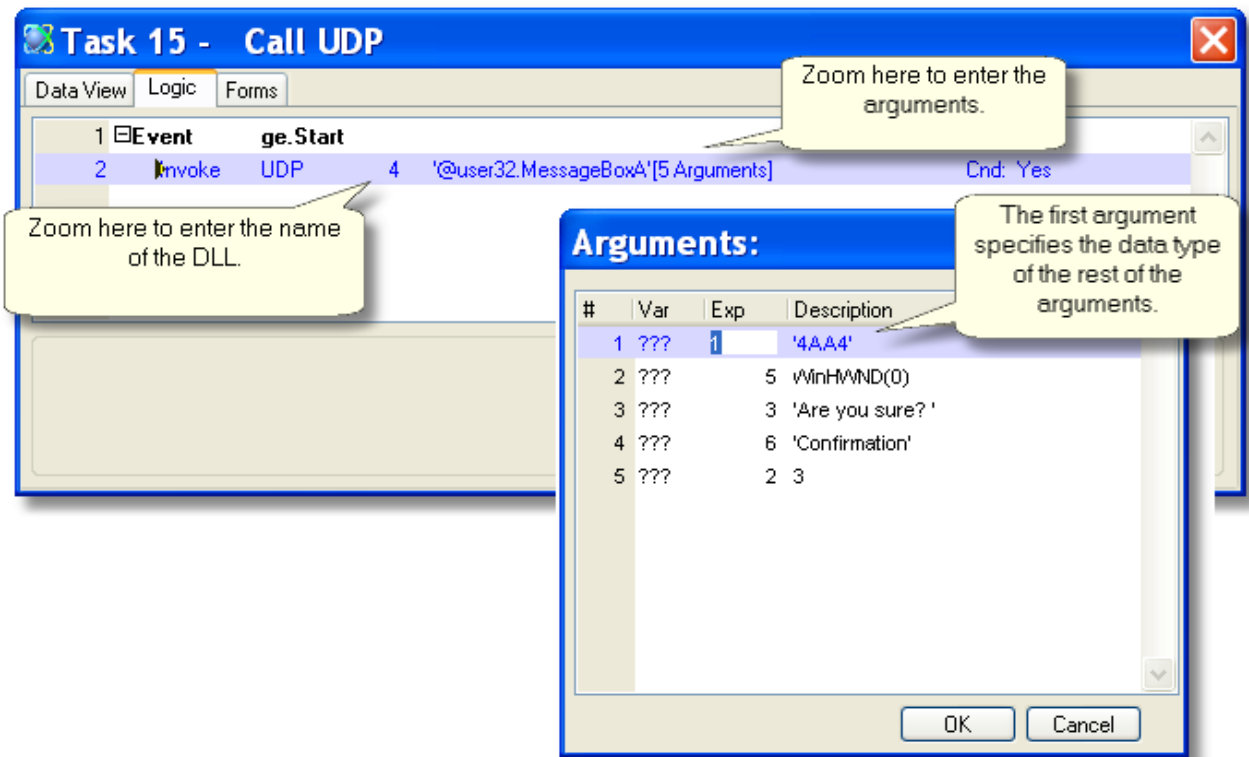
Letter	Data Type
1	Char
2	Short
4	Long
F	Float
8	Double
D	Double pointer
E	Float pointer
L	Long pointer
A	Null terminated string pointer
V	Void pointer
0	Void

When we are done entering the CallDLL and we run it, we get the Windows message box. When the user presses a button, that returns a number to our calling program (6=Yes, 7=No, 2=Cancel).



**Hint:** When using operating-system dependent functions such as this one, it's a good idea to encapsulate them inside an eDeveloper task or function. That way, if you decide to use a different call, or something changes about your installation, it's easy to change.

## Using Invoke UDP



Calling a DLL with an Invoke UDP operation is very similar to using the **CallDLL()** function.

1. Go to the appropriate Logic unit, and press F4 to open up a line.
2. Type I to select the *Invoke* operation. Tab to the next field.
3. Type U to select *UDP*. Tab to the next field.
4. Zoom to the Expressions list, and type in the name of the DLL you want to call, preceded by an '@'. In our example, we are calling the Windows user32.MessageBoxA, so we enter '@user32.MessageBoxA'

Press Enter to select the expression, then tab to the next field.

5. Zoom to add your arguments. These use the same format as those in the CallDLL() function, but they are easier to format.
  - The first argument contains one character for each argument required by the DLL. MessageBoxA takes 4 arguments, so we have 4 characters. Each character represents the data type of the argument: a '4' is a Long integer, and an 'A' is a null-delimited string.
  - For the other arguments, zoom from the Var or Exp columns as needed.

## How do I Call a Program By Its Name?

If you have a program with a public name, you can call the program directly via that name. This is not usually necessary, as you can use the Call Prog operations for most calls.

Program Repository: Utilities			
#	Name	Folder	Public Name
1	Main Program		
60	Pause	Utilities	
61	ActiveX calendar	Utilities	Calendar
62	<T> Test calendar	Utilities	

However, if you have a situation where you want to store the name in a table, this is useful. It's also useful if you won't have access to the program while you are coding, because it is in a cabinet file that is only available at runtime.

### Calling a program by name

**Task 75 - Calling a program by name**

Logic View | Logic | Forms

#	Event	g.start	Variable	F	Program to call	With:	1	Scope: Task	'Calendar'
2	Update	Variable	F	Program to call	With:	1	'Calendar'		
3	Call	By Name	Program to call	[0 Arguments]	Result:	???	Cnd:	Yes	

**Call By Name Details**

Public Program Details

Set an expression for the public name of the program you are calling. If the program is in a different cabinet file, set an expression for the cabinet file name.

Public program name:  Program to call

Cabinet file name:

**Expression Rules: 75 - Calling a program by name**

#	Expression
1	'Calendar'
2	F

Expanded View

Program to call

1. Press *F4* to create an operation line.
2. Press 'C' to create a *Call* line.
3. Type 'N' to select *By Name* for the call type, or use the pull-down list. Tab.
4. Zoom to access the *Call by Name Details* dialog box. Zoom from the *Public program name* field to the expressions list, and type in the name, or select a variable that will contain the name at runtime.
5. If the program you wish to call is in another cabinet file, do the same for the *Cabinet file name*.

Now, the program named in the variable will be called at runtime.

## How do I Call a Program Dynamically By Its Index?

You can call a program based on its index number. You can do this two ways, by using the *Call by Expression* operation, or by using the function **CallProg()**.

**CallProg()** is useful where you want the program to execute within an expression, and you want to use the returned value as a parameter to another function.



### Using Call by Expression

1. Use *F4* to create an operation.
2. Press 'C' for the *Call* operation. Tab to the next field.
3. Select *By Exp* for the call type. Tab to the next field.
4. *Zoom* to create an expression. The expression should be the program's index number in quotes, followed by the literal PROG. We are calling program 61, so we entered '61'PROG.

### Using CallProg()

1. In the Expression Editor, type in the function

`CallProg('n' PROG)`

(where *n* is the program index) wherever you want the return value of the program. This could be as an init value, as an expression displayed on a form, or as part of another expression. Or, if the program has no returned value, you can use an *Evaluate* operation.

You can also use **CallProg()** in conjunction with the **ProgIdx()** function, to call the program from within an expression using the program's public name.

These methods are mainly useful when you want to store the name or number of the program to call. For instance, you could create a table to allow the user to create their own private menu system.

**Note:** The PROG literal is what allows eDeveloper to keep the program number synchronized if the program happens to get moved within the Program repository. You can use the format `CallProg(61)` and it will function, but if program 61 gets moved to become program 65, the function will still be calling program 61.

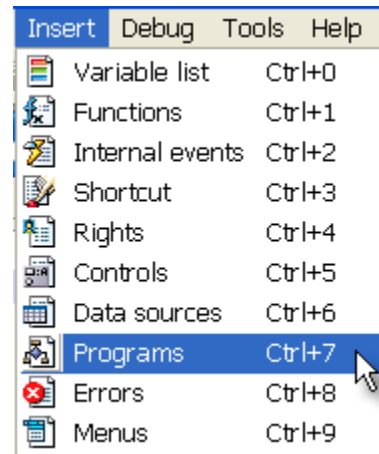
**Hint:** This method is useful in combination with the **ProgIdx()** function, which returns the program index based on its public name.

### Selecting the program

When you are selecting a program number, you do not have to memorize the number. For this, and other items you need to add to expressions, you can use the Insert menu (also available from the right-click menu). For the examples above, to select the program number, you would:

1. Press **Ctrl+7**, or select **Insert->Programs**, or select **right-click->Programs**. A list of programs will appear.
2. Find the program you want. You can use Locate (Ctrl+L), or type the first letters of the program name, or just scroll, to position on the program you want.
3. Press the Select button.

The program literal will be brought into your expression.



## How do I Retrieve the Name of the Control the User Parked On?

The screenshot shows a window titled "Control Park" with a blue title bar. Inside, there are several text input fields: "Customer ID:" with the value "XST", "Contact First Name:" with the value "Frank", "Contact Last Name:" with the value "Smith", and "Company name:" with the value "Xavier Systems Tracking". A "View Details" button is located to the right of the first three fields. Below these fields are two more fields: "Last Clicked:" (empty) and "Last Parked:" (containing the text "Customer ID"). A red curved arrow originates from the left side of the "Last Parked:" field and points towards the "Customer ID" text within it.

Sometimes it is useful to know where the user last parked. You may want to do processing. You do this by using the **LastPark()** function. Note that this is different from where the user last clicked. In the example above, we used tab to get from one field to another, so the last field we were parked on was Contact ID. However, we didn't click on any fields at all, so **LastClicked()** returns blank.

**LastPark()** allows you to retrieve the control name from the current task that contains the control, and from that task's child tasks. However, it does not work:

- To retrieve the control name of the *currently* parked field
- From within logic units that happen to be triggered within parent tasks to this task.

You can fetch the currently parked control from this or a parent task by using **HandledCtrl()**. You can use the **This()** function to retrieve other information about the current field.

### LastPark()

The syntax for **LastPark()** is:

```
LastPark(Generation)
```

where **Generation** is a number representing the tasks hierarchic position in the task tree. 0 is the current task, 1 is this task's parent, and so on.

**Note:** If you are trying to do specific processing for one field in your current task, you can do this using the Control events for that control.

## How do I Retrieve the Name of the Control the User Clicked On?

It is often useful to know where the user last clicked. This is different from where the user last parked, because some controls, such as push buttons, may not be parkable.

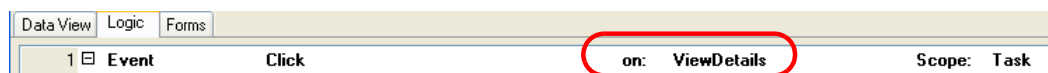


### LastClicked()

**Prerequisite:** The control must have a control name, or nothing will be returned.

You retrieve the value of the last control the user clicked on with the function **LastClicked()**. There are no parameters. It returns a string which represents the control name. That string can then be passed to other functions and expressions that test for the control name.

**Note:** If you are trying to execute some code based on which control the user clicked on, you can also do this by adding the “On” field to the Click event. For instance, in the example above, we could code the Event as:

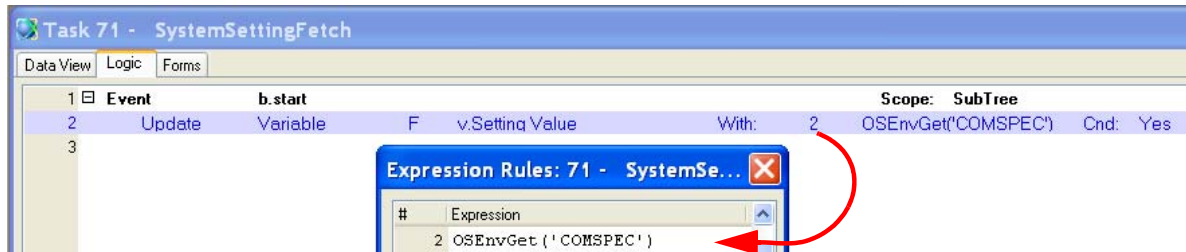


Then that logic unit would only execute if the View Details control was clicked.



## How do I Retrieve a Value from the System Environment Settings?

The system environment has a number of useful pieces of information available for you, such as the system user id, the operating system type and version, the computer name, and the path to the command shell. You can use these within your eDeveloper programs by fetching the environment variable with the function **OSEnvGet**.



### OSEnvGet

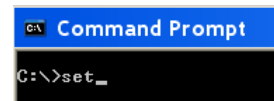
The function **OSEnvGet** has the syntax:

```
OSEnvGet (Variable)
```

Where *Variable* is the system name of the system environment variable.

In this example we used it with an update operation to store the path to the command shell. Some of the more useful command shell variables are COMSPEC, OS, USERNAME, USERDOMAIN, and LOGONSERVER. However, you can also create your own, and there are likely some useful settings that are local to your environment.

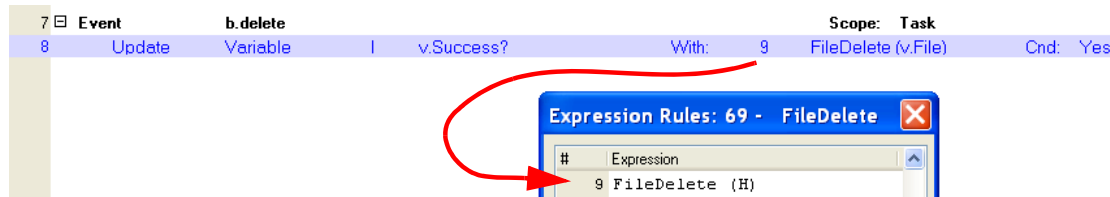
**Hint:** You can view all your current environment settings by opening a command shell and typing **SET** at the command prompt.



## How do I Delete a Disk File?

Sometimes you will want to delete disk files. After you create temporary files, for instance, you will want to clean them up out of the temporary directory. Or, if you are recreating a file, you may want to be certain that the old one is gone before starting the new process.

It is always better to use the eDeveloper functions rather than using an exit to the operating system. Using an exit to the operating system is very dependent on the version of the operating system and how the user is set up; the eDeveloper functions are far more reliable. Also, the eDeveloper functions allow you to check return codes more easily.



### FileDelete()

You can delete a file on disk using the **FileDelete()** function. The syntax is:

```
FileDelete(FileSpec)
```

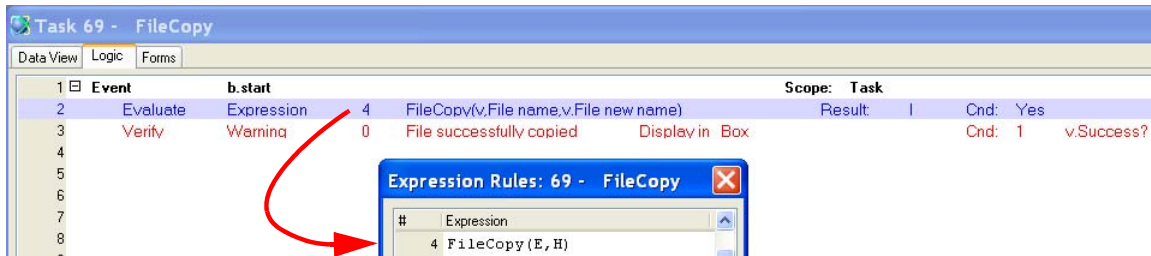
Where *FileSpec* is the file you want to delete. It gives a return code of TRUE if the file was successfully deleted.

**Note:** The return code will be *false* if the file wasn't deleted. It will also be *false* if there was no file to delete in the first place. Therefore, it's a good idea to use this in conjunction with **FileExist()**, so you know the file has been deleted.

**See also:** Chapter 6, "FileExist()" on page 184.

## How do I Copy a Disk File?

Sometimes you may want to make a copy of a disk file. This is useful for making backup or archival copies, for instance.



It is always better to use the eDeveloper functions rather than using an exit to the operating system. Using an exit to the operating system is very dependent on the version of the operating system and how the user is set up; the eDeveloper functions are far more reliable. Also, the eDeveloper functions allow you to check return codes more easily.

### FileCopy()

You can copy a file on disk using the **FileCopy()** function. The syntax is:

```
FileCopy(origin, target)
```

Where *origin* is the original file and *target* is what you want to copy it to. It gives a return code of *true* if the file was successfully copied.

**Note:** **FileCopy()** will overwrite any existing file of the same name. If you want to give the user a warning message, use the **FileExist()** function first.

**See also:** Chapter 6, “FileExist()” on page 184.

## How do I Check if a File Exists on Disk?

One common function you need is to check if an IO file exists on disk. For instance, if a user was dumping records to be read into Excel, you might want to give a message if no records were found or the process was unsuccessful for some other reason.



It is always better to use the eDeveloper functions rather than using an exit to the operating system. Using an exit to the operating system is very dependent on the version of the operating system and how the user is set up; the eDeveloper functions are far more reliable. Also, the eDeveloper functions allow you to check return codes more easily.

### FileExist()

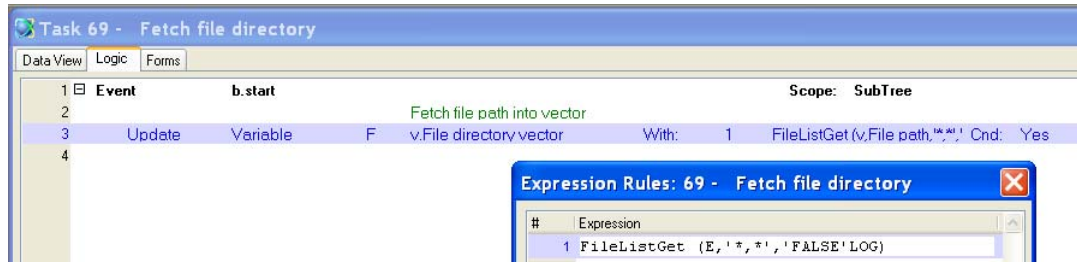
You can determine whether or not a file exists using the **FileExist()** function. The syntax is:

```
FileExist(FileSpec)
```

Where *FileSpec* is the name of the file you are looking for. It returns *true* if the file is found.

**See also:** The eDeveloper Help for IO functions.

## How do I Retrieve the Content of a File Directory?



Sometimes you will need to read a file directory inside a program. This is often done when you are checking to see if another program has dropped a file for your program to handle, such as when files are sent by email, faxing programs, or from the web.

### FileListGet()

You fetch the directory contents using the **FileListGet()** function. The syntax is:

```
FileListGet(DirectoryName, Filter, SubDirSearch)
```

Where *DirectoryName* is the name of the directory to search, *Filter* is a filter mask, and *SubDirSearch* controls whether or not to search subdirectories.

The function returns a vector, where each cell is an alpha string.

**See also:** Chapter 15, “Passing an array to a COM object” on page 395  
The eDeveloper Help for IO functions.

## How do I Rename a File on Disk?

Sometimes you need to be able to rename a disk file. For instance, you might want to rename the previous copy of a backup file before creating a new one.

The screenshot shows a logic editor with a table of events and a dialog box titled "Expression Rules: 68 - Fetch file information".

#	Event	b.rename	Expression	Scope	SubTree
5					
6					
7	Evaluate	Expression	3 FileRename(v.File name,v.File new name)	Result	I Cnd: Yes
8					
9					
10	Update	Variable	1 v.Success?	With: 3	FileRename(v.File name,v
11					
12	Verify	Warning	0 File successfully renamed	Display in Box	Cnd: 1 v.Success?

The dialog box "Expression Rules: 68 - Fetch file information" has a list of expressions:

- 1 I
- 2 FileSize (E)
- 3 FileRename (E,H)

The "Expanded View" section shows: FileRename (v,File name,v.File new name)

It is always better to use the eDeveloper functions rather than using an exit to the operating system. Using an exit to the operating system is very dependent on the version of the operating system and how the user is set up; the eDeveloper functions are far more reliable. Also, the eDeveloper functions allow you to check return codes more easily.

### FileRename()

You can rename a file on disk using the **FileRename()** function. The syntax is:

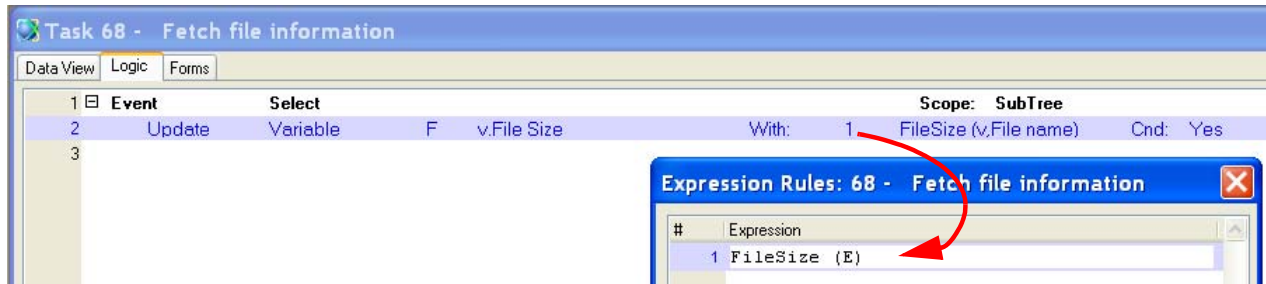
```
FileRename (origin, target)
```

Where **origin** is the original file name and **target** is what you want to rename it to. It gives a return code of true if the file was successfully renamed.

We show two examples here. The first, on line 7, uses an *Evaluate* operation and fetches the return code in the *Result* field. The second, on line 10, uses an *Update* operation to accomplish the same thing. Either will work.

**See also:** The eDeveloper Help for IO functions.

## How do I Get the Size of a File on Disk?



Extended Logic

Sometimes you need to know how big a file is on disk. This might be the case if you were trying to see if any data were created in it (the size is not zero) or if it might be too large to send as an email attachment.

### FileSize()

You find size of a file on disk using the **FileSize()** function. The syntax is:

```
FileSize (FileSpec)
```

Where *FileSpec* is the file name.

**See also:** The eDeveloper Help for IO functions.





# Chapter 7: Deployment

## How do I deploy my project?

Once you have your project complete, you will want to give some thought to how the application is to be deployed. This will vary depending on who the end-users are. It’s a good idea for any application to have a unique icon and application name. If your application is a saleable product, however, then you will probably want it to have a splash screen, and to be self-installing.

Below are the basic steps involved in deploying a project, and where to find information detailing how to do each.

#	Description	How to do it
1.	<b>Set up your application icon.</b> This allows you to have an easily identifiable symbol for your application.	Chapter 7, “Setting the Logo File” on page 196.
2.	<b>Create your cabinet file.</b> The cabinet file is a non-modifiable, packaged version of your project.	Chapter 7, “How do I Create a Cabinet File?” on page 191
3.	<b>Install eDeveloper Runtime at the client site.</b> The eDeveloper Runtime product is what will execute your application.	The installation directions come with the eDeveloper product.
4.	<b>Create a shortcut or menu option for the end user.</b> Make it easy for the end user to start the application. There are several ways the end user can start your application; which you use is up to you.	Chapter 7, “How do I Create a Shortcut for my Application?” on page 192

#	Description	How to do it
5.	<b>Set up a splash image, if you want.</b> Like the application icon, this gives you a way to brand your application.	Chapter 7, “How do I Display a Splash Image on Loading an Application?” on page 196
6.	<b>Create a self-installing file, if you want.</b> You can have a professional-looking self-install file for you application.	Chapter 7, “The Default Application is saved in the Magic.ini file, which is by default stored where eDeveloper is installed. For some applications, this is all you need, if there is only one default application per computer. However, if you have in installation where different projects are started at different times, you will want to start the project directly, as shown in Chapter 7, “Creating a shortcut to the project file” on page 192.” on page 194

## How do I Create a Cabinet File?

When you are working in the eDeveloper Studio, you are accessing an *eDeveloper Project*, or **.edp**. This **.edp** refers to a collection of source files, which are typically kept in the `\source` subdirectory in XML format.

However, once your project is complete, you will want to package it up for installation. The packaged file is known as the *eDeveloper Cabinet File*, or **.ecf**. From an installation point of view, the **.edp** and **.ecf** work similarly in that you can click on either one to get started. There are some important differences though, as shown below.

<b>.edp</b>	<b>.ecf</b>
eDeveloper Project	eDeveloper Cabinet File
Opens the eDeveloper Studio	Runs your application
For programmers only	For users mainly, or for use as components while developing
Requires XML source files exist.	Contains the packaged version of the XML source

When you are deploying your project, you want to deploy only the **.ecf** file, and some other supporting files (fonts, colors, components).

Also, when you are deploying, the user will be installing and running only the Runtime version of eDeveloper.

It is very simple to create the cabinet **.ecf** file. Below we give you the steps.

### Creating a cabinet file

1. Open the project.
2. From the overhead menu, select **File->Create Cabinet**.
3. You will be prompted for the cabinet file name. Type in the name or select it, then press Save.
4. If the **.ecf** file already exists, you will receive a warning box about overlaying it. Select **Yes** if you want to overlay it.

After the dialog boxes disappear, your **.ecf** file will appear, and be ready for use.

**See also:** Chapter 7, “How do I Create a Shortcut for my Application?” on page 192.

## How do I Create a Shortcut for my Application?

Probably the most common way to run an application is to run it from a Windows shortcut. These shortcuts are very easy to create. The shortcut can either use the .edp file as a target, or use the eDeveloper engine as a target, with the setup such that it automatically loads the desired project. Both methods are shown below.

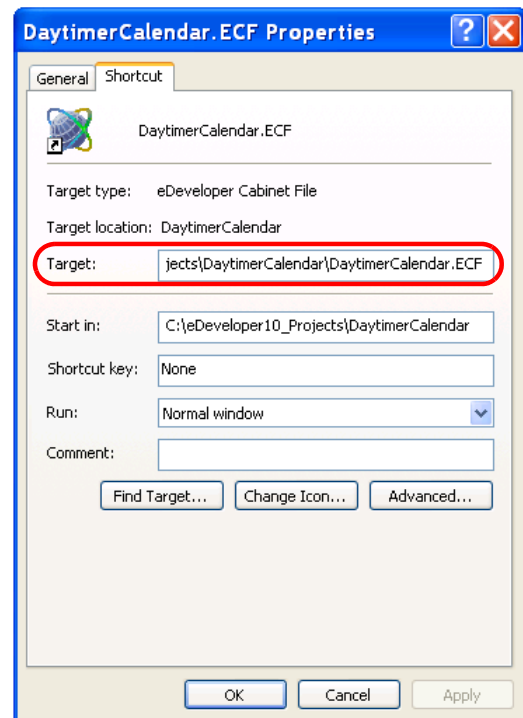
**Note:** You can also keep in mind that you are not required to create a shortcut. If the user clicks on the .ecf file in Windows Explorer, Windows will automatically invoke eDeveloper to run it.

### Creating a shortcut to the project file

1. In the folder where you want to create the shortcut, select **New->Shortcut** from the right-click menu. The *Windows Create Shortcut* wizard will appear.
2. You will be prompted to type the location of the item. Enter the path name to your cabinet file, or use the Browse button to select it. Click **Next** to continue.
3. You will then be prompted to fill in a name for your shortcut. Give it any name you like.
4. Press Finish to end the dialog

Now, you will have a shortcut in your folder that will start your application. By default, the icon will be the eDeveloper icon, but you can customize this as shown in Chapter 7, “Using your own icon” on page 193.

If you look at the Properties for this shortcut, they will look something like the example on the right. The eDeveloper Cabinet File, or .ecf, will be the Target.



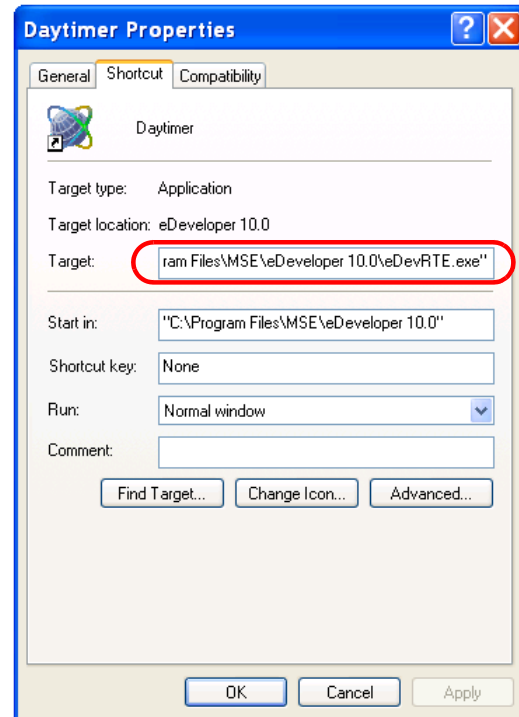
## Creating a shortcut to the eDeveloper engine

1. In the folder where you want to create the shortcut, select **New->Shortcut** from the right-click menu. The *Windows Create Shortcut* wizard will appear.
2. You will be prompted to type the location of the item. Enter the path name to the eDeveloper runtime engine, which is by default:

"C:\Program Files\MSE\eDeveloper 10.1\DevRTE.exe"

3. You will then be prompted to fill in a name for your shortcut. Give it any name you like.
4. Press Finish to end the dialog

This method will only start your particular application if the application has been set to start automatically, as explained in Chapter 7, "How do I Make the Runtime Engine Automatically Load a Cabinet File?" on page 194.



## Using your own icon

No matter which type of shortcut you create, the default icon will be the eDeveloper icon. You can, however, use your own custom icon.



**Prerequisite:** You should first create your custom icon file.

1. Select Properties from the Right-click menu of your shortcut.
2. Click on the Change Icon button.
3. You will see the Change Icon dialog box. The top line says "Look for icons in this file". Use the Browse button to select your icon file, then select the icon you want to use.

Now your icon will show instead of the default eDeveloper icon.

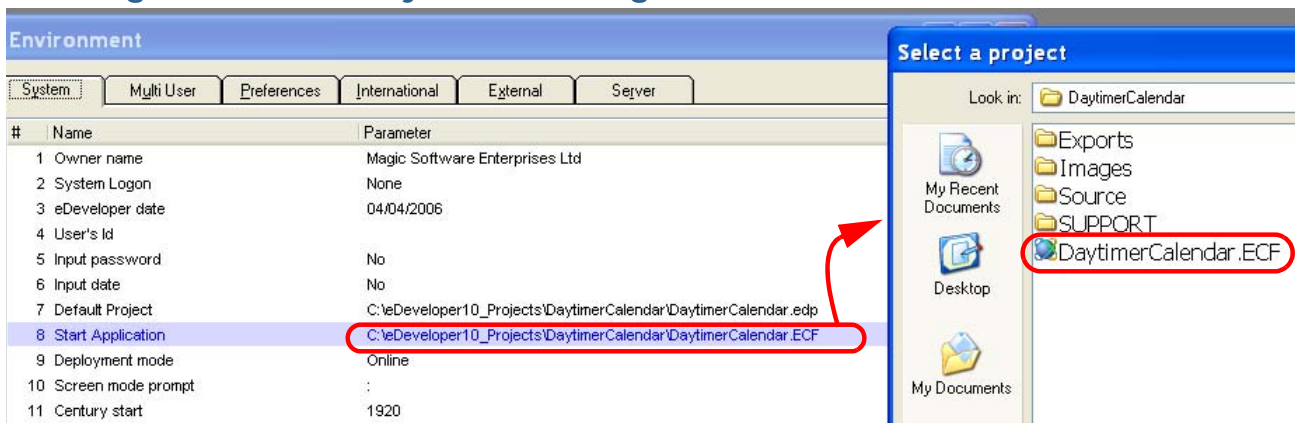
If you do not have your own customized icon file, the system will default to the Windows icon file and you can select one of those. You can also purchase generic icon files.

## How do I Make the Runtime Engine Automatically Load a Cabinet File?

You can make the runtime engine automatically load a cabinet file by specifying the cabinet file to run in the Options menu. If you specify the cabinet file there, then any time eDeveloper is run on that computer, that specific project will run by default.

Note that there are two defaults specified. One is for the “Default Project”, which is what will open automatically in the eDeveloper Studio ( the **.edp** file). The other is “Start Application” which is what will open when the runtime engine is invoked (the **.ecf** file).

### Setting the Default Project in the Magic.Ini

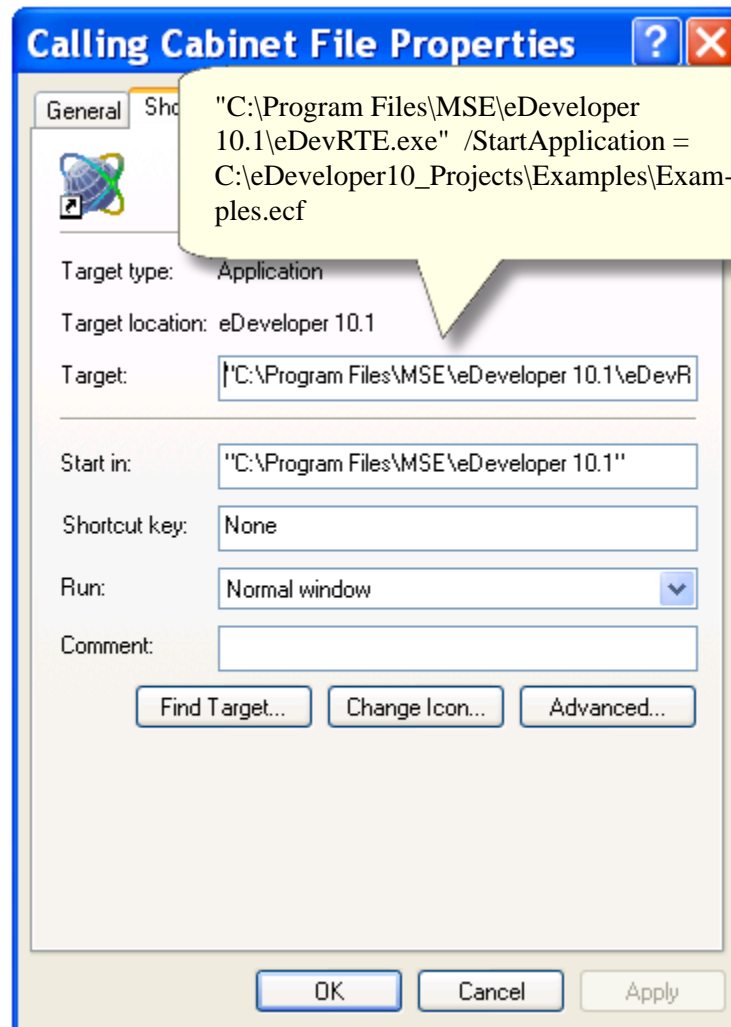


1. Open your project.
2. Go to **Options->Settings->Environment->System->Start Application**
3. Press **zoom (F5, or Edit->Zoom)** to get a dialog to select the .ecf file you want. Alternatively, you can just type in the file path.

Now, when you start eDeveloper in Runtime mode, this cabinet file will open.

**Note:** The *Default Application* is saved in the Magic.ini file, which is by default stored where eDeveloper is installed. For some applications, this is all you need, if there is only one default application per computer. However, if you have in installation where different projects are started at different times, you will want to start the project directly, as shown in Chapter 7, “Creating a shortcut to the project file” on page 192.

## Specifying the cabinet file in the Shortcut



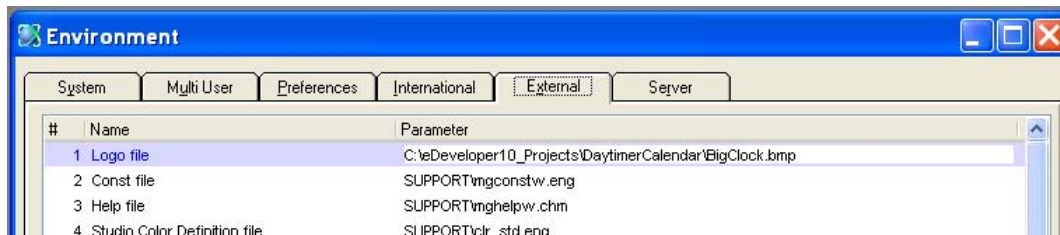
Alternatively, you can specify the cabinet file in the Target field of the Windows shortcut.

1. In the Target field, enter the path and file name of the eDeveloper runtime engine, enclosed in double quotes.
2. Follow that with a space and a forward slash.
3. Enter the path and filename of the cabinet file.

## How do I Display a Splash Image on Loading an Application?

You can make eDeveloper display your custom splash image at runtime if you like, by specifying a logo file. This is stored in the Magic.ini file, so will pop up whenever any eDeveloper application is started. One of the advantages of a splash screen is that it gives the user an interim visual representation of the application until it is fully loaded.

### Setting the Logo File



1. Create your logo file. This can be any image file, but you have to create it to be the exact size you want, as it is not resized when it displays.
2. Open your project file, and go to **Settings->Environment->External->Logo File**.
3. Type in the name of your logo file, or **zoom (F5, Edit->Zoom)** to select your file.

The next time you run your application, the splash screen will appear.



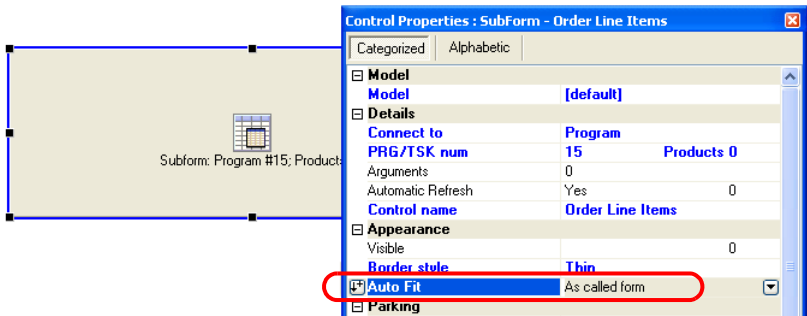
# Chapter 8: Subforms

## How do I Make the Subform Control Fit the Dimensions of the Called Program Form?

When you are using a subform, you may not know the size of the called program’s form. Rather than guess, you can use the Autofit option to make the control resize itself automatically.

### Setting Autofit

- 1. Select the Subform control.
- 2. Zoom to go to the Control Properties.
- 3. For the *Autofit* property, select *As called form*.



**Note:** When you use *As called form*, the size of the Subform control has no effect on the actual size at runtime. Therefore, the placement properties also have no effect.

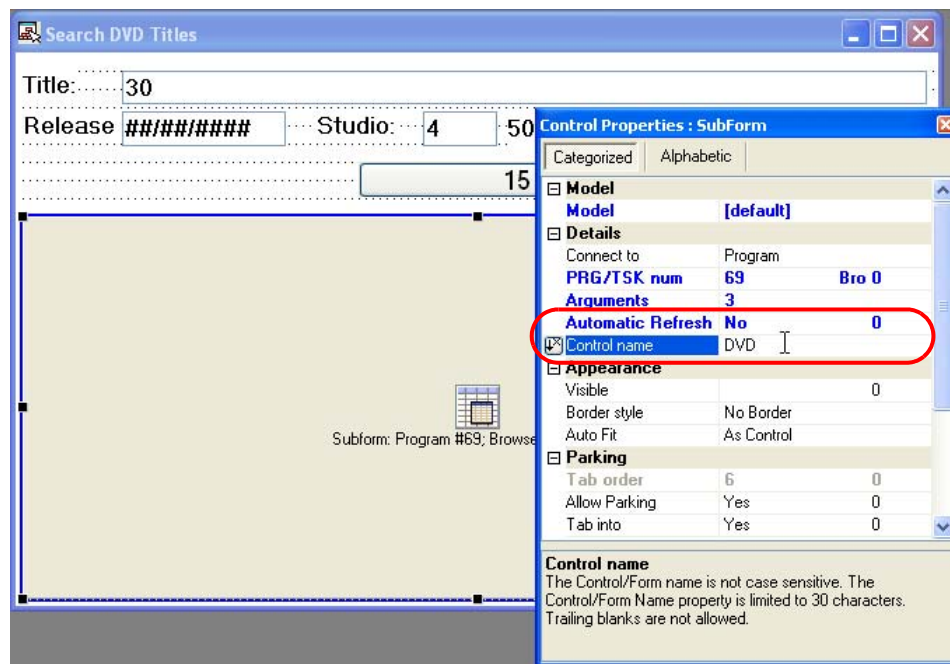
**See also:** Chapter 8, “How do I know which Autofit Option to use?” on page 208.

## How do I Manually Refresh the View of the Subform?

Normally, the Subform control automatically refreshes itself whenever the parameters that are sent to it change. So if, for instance, you have a list of DVD titles in the parent form and are showing details of each DVD in the subform, the details will change automatically as you scroll through the list. This is usually exactly what you want, and you don't have to do any work to make that happen.

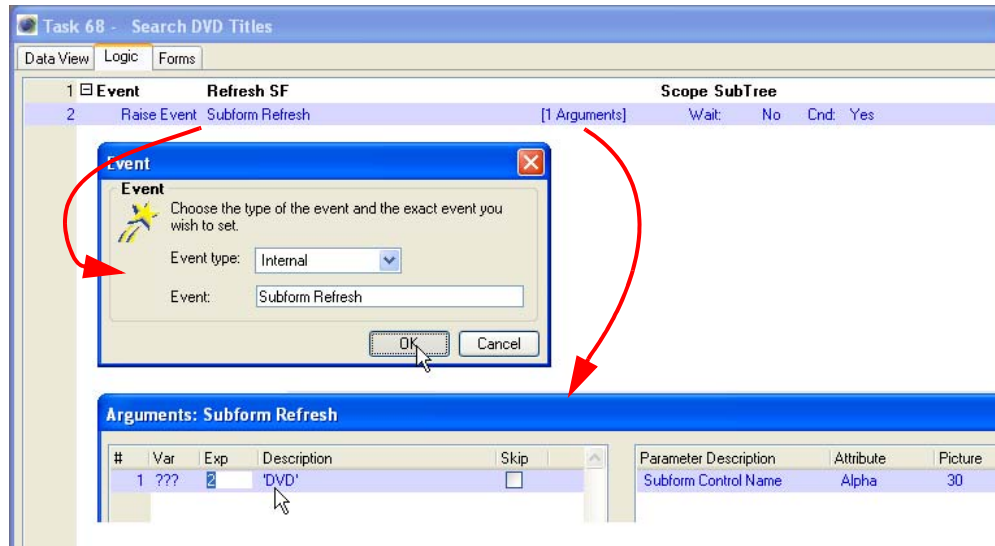
However, if the record set is large, or the search complex, the refresh might take enough time that it will be bothersome to the user. In this case, you might want to disable the automatic refresh and have it done only when the user presses a "search" or "refresh" button. Here is how to do it.

### Manually refreshing a subform



1. On your Subform control, set *Automatic Refresh* to *No*.

2. Make sure your Subform control has a control name. Here it is “DVD”.



3. Create a user event that gets triggered when you want to refresh the subform. In this example we have a push button that raises the user event “Refresh SF”.
4. In the Logic Editor, create a handler for your event.
5. In this logic unit, raise the event **Subform Refresh**.
6. As an argument to **Subform Refresh** event, pass the name of the subform you want to refresh.

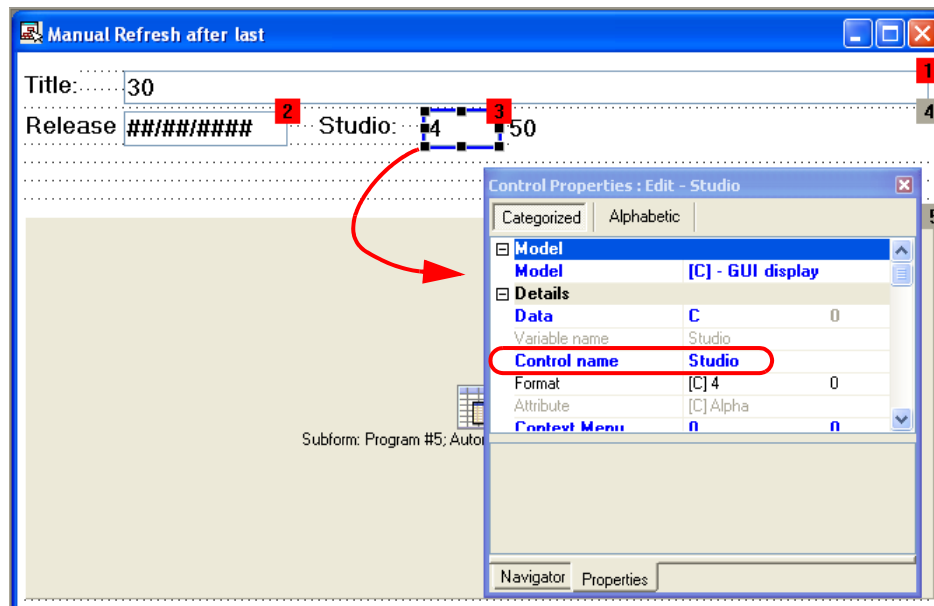
Now, your subform will only refresh when the user presses your refresh button.


**See also:** The eDeveloper V10 online demonstration, *GUI Subforms - Rapid implementation of Parent-Child relationships*  
The eDeveloper V10 documentation

## How do I Refresh the Subform View only on Modifying the Last Argument When Passing Several Arguments to the Subform?

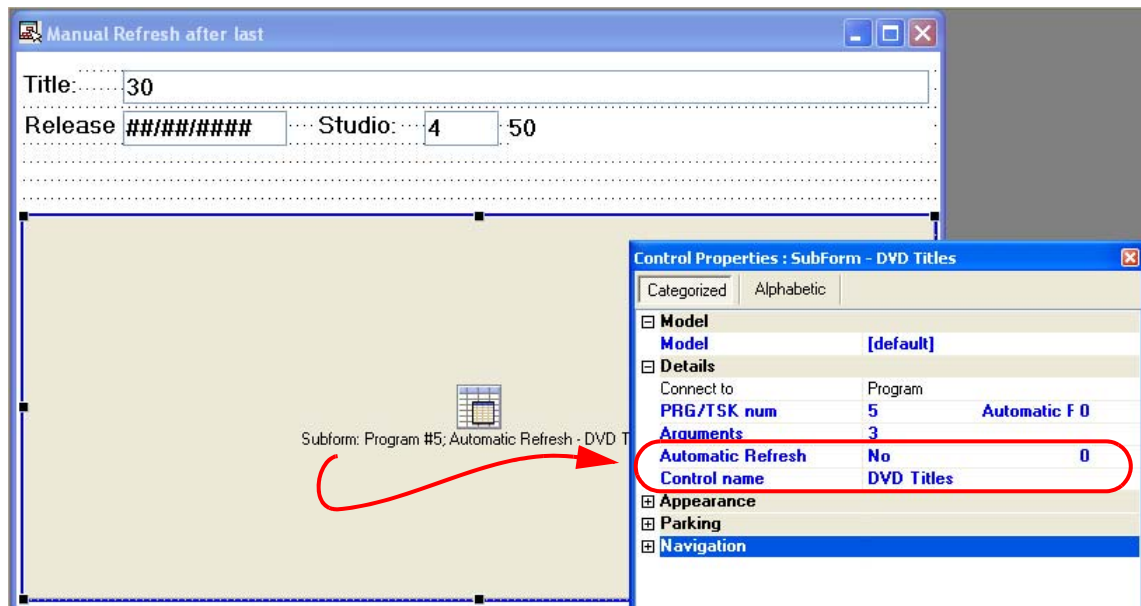
Normally, the subform will refresh itself whenever any of the parameters that are being passed to it change. However, if there are several parameters -- such as search criteria for a list -- then the subform will refresh every time the user changes any one of the criteria. This might not be what the user expects, and it can also slow down processing if it's a complex search.

### Using an expression on a subform automatic refresh

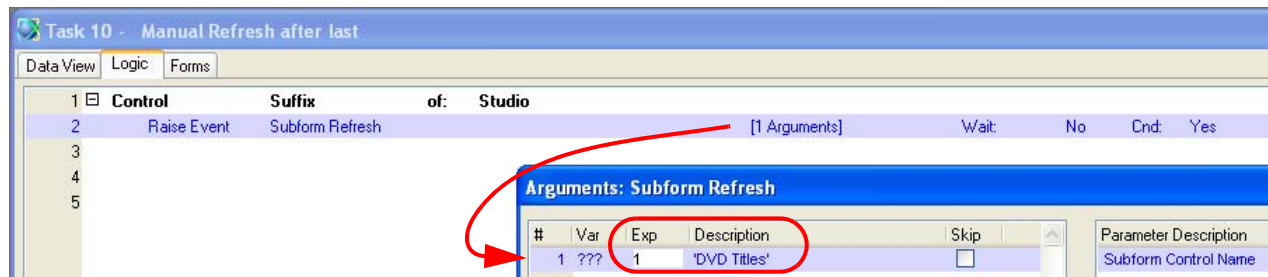


1. On your parent task, make sure that your tab order is correct, by pressing  on the Commands palette.

2. Make sure the last field has a *Control Name* (“Studio”, in this example).



3. In the Subform control, set *Automatic Refresh* to *No*.
4. Make sure the subform has a control name (“DVD Titles” in this example).



5. In the *Logic Editor*, create a *Control Suffix* event for the last search criteria control.
6. Add a *Raise Event* operation that raises the internal event *Subform Refresh*.
7. As an argument to this event, pass the name of your subform (“DVD Titles” in this example).

Now, the subtask will refresh when the user passes the last control.

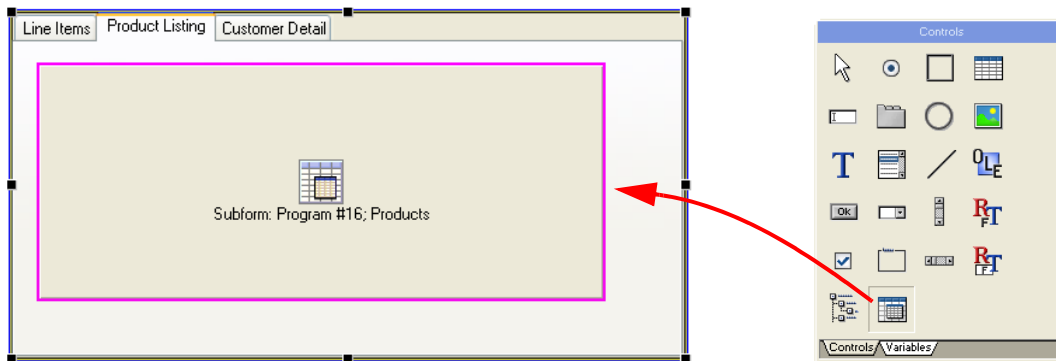
**Hint:** If this is combined with the *Tab Into* property on the subform, the user will enter the criteria and drop into the subform nicely. Also, in your *Control Suffix* control, you can add an *Evaluate* operation with **CtrlGoToQ** to set exactly which control in the subform the user lands on.

**See also:** The eDeveloper V10 online demonstration, *GUI Subforms - Rapid implementation of Parent-Child relationships*  
The eDeveloper V10 documentation

## How do I Control the Visibility of a Subform When it is Placed on a Tab Control?

When you place a subform on a Tab control, it will only be visible on that tab. There is nothing else you need to do: just place it as you would any other control on a tab.

### Adding a subform to a tab



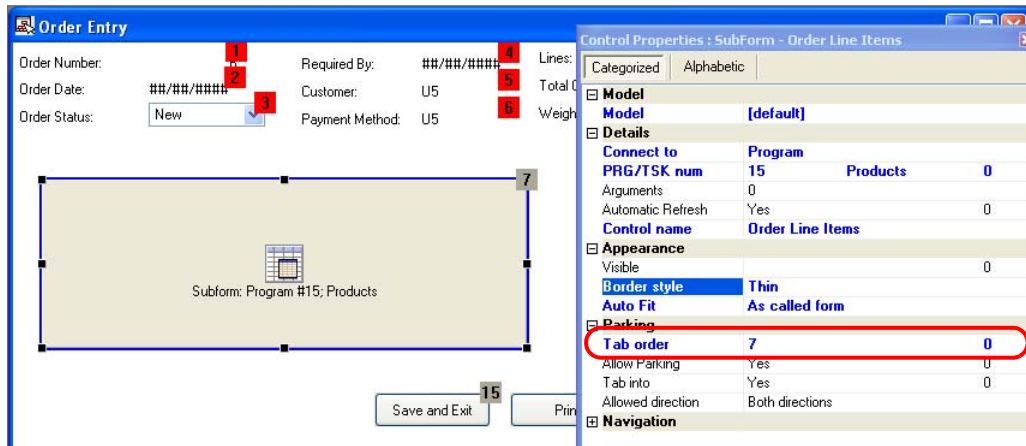
1. Select the Tab control by pressing *Ctrl+Click*, on a part of the tab that doesn't contain any other control. This selects *only* the Tab control, not the items attached to it.
2. Press the **Enter** key repeatedly. You will see that you are cycling through the tabs.
3. When you reach the tab that you want the subform on, click on the Subform control, and drop it on the tab. You will see it turn pink, as shown above. Then proceed to set up the subform as you ordinarily would.


**See also:** Chapter 3, "How do I Automatically Drop Form Controls Using a Specific Control Model?" on page 48.  
Chapter 5, "Attaching the control" on page 137.

## How do I Set the Subform to Be Tabbed Into from a Specific Control of the Parent Form?

Usually, eDeveloper will set the tab order to be as you would expect it, tabbing from the upper left corner to the lower right. However, you can set the tab order manually if you need to.

### Setting the Tab Order of a Subform Control

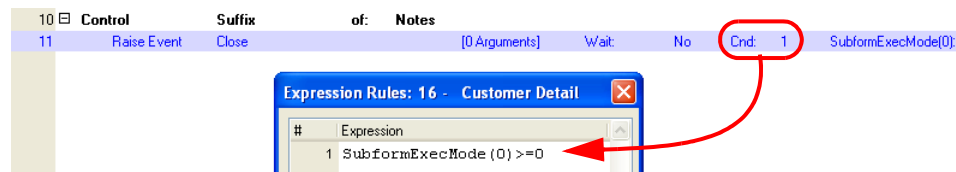


1. Turn off automatic tabbing, if it isn't already, by pressing the  icon on the commands palette (**Drawing->Order->Automatic Tab Order**).
2. Set the tab order of the other controls as desired.
3. Set the tab order of the Subform control to one greater than the control that it is tabbing from. So in this case, the tab order goes from "Payment Method" (6) to the "Products" subform (7).
4. Make sure the Subform control has the defaults still set for *Allow Parking* and *Tab into* (both should be Yes).

## How do I Automatically Return Back to the Parent Form by Tabbing Out of the Last Control of the Subform Display?

When the user tabs into a subform, they might expect to be able to tab out of it also. You can do this easily by using a control event. However, a subform program might also be called as a stand-alone program, in which case you might not want the program to exit by tabbing. So, you use the **SubformExecMode()** function to control when you want the exit to happen.

### Setting an event to exit a subform



1. Decide which control is the last one on your form, and make sure it has a control name. In this case, or last control is named “Notes”.
2. Create a *Control Suffix* event, selecting that control’s name.
3. Create a *Raise Event* operation, with the *Cnd* of

`SubformExecMode(0) >= 0`

**SubformExecMode()** returns -1 if the task is not called as a subform. The other return codes give more information about how the subform task is being called. See the eDeveloper help for the function for more information.



## How do I Execute Task Prefix/Suffix Logic of a Subform only on Opening the Subform Task for the First Time by its Parent?

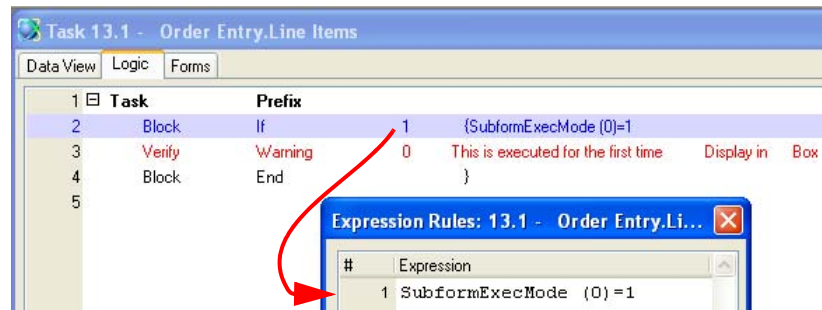
Task prefix executes whenever a task starts, and task suffix executes whenever the task ends. When you are using a subform, the subform task executes once, to display its contents, when the parent task first executes. But it also executes when the parent task is refreshed, and then again when the user actually enters the subtasks. So, if you want to have logic that executes only during this initial execution of the subform task, you need to use the **SubformExecMode()** function.

If the subform task is opened for the first time, SubformExecMode(0) will return 1. So using

```
SubformExecMode ( 0 ) = 1
```

as the condition on the block will cause the block to *only* execute when the subform is called the first time.

### Coding a block that only executes when the subform executes the first time



1. Open up a line in the logic unit you are modifying (**F4** or **Edit->Create Line**).
2. Select the *Block* operation by typing B or using the pulldown list.
3. Two lines will open up, a *Block If* and a *Block End*. Tab to or click on the field after the *If*.
4. *Zoom* (**F5**, double click) to the expression rules. Enter the expression:

```
SubformExecMode ( 0 ) =1
```

5. Press *Enter* to select the expression and bring back the expression number

Now the block will only execute when the subform is executes for the first time.

## How do I Execute the Task Prefix/Suffix Logic of a Subform Whenever the User Enters the Subform?

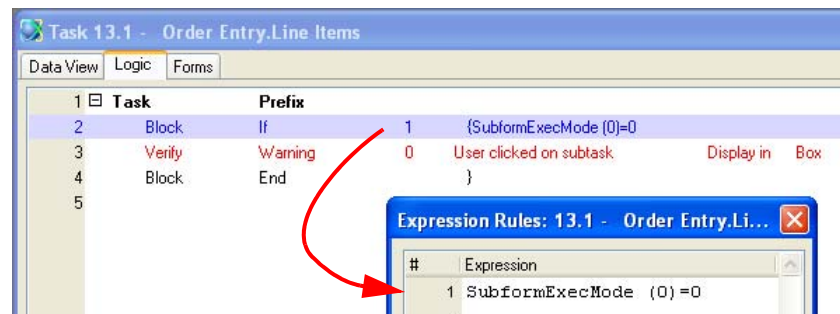
Task prefix executes whenever a task starts, and task suffix executes whenever the task ends. However, in a subform, you can have a *Block* operation within the task logic unit which executes depending on how the subform was entered. This is done using the **SubformExecMode()** function.

If the subform task was entered by a user action, such as tabbing into the subform or clicking on it, then **SubformExecMode(0)** will return 0. So using

```
SubformExecMode ( 0 ) = 0
```

as the condition on the block will cause the block to only execute when the user enters the subform.

### Coding a block that only executes when the user enters the subform



1. Open up a line in the logic unit you are modifying (**F4** or **Edit->Create Line**).
2. Select the *Block* operation by typing B or using the pulldown list.
3. Two lines will open up, a *Block If* and a *Block End*. Tab to or click on the field after the *If*.
4. *Zoom* (**F5**, double click) to the expression rules. Enter the expression:

```
SubformExecMode ( 0 ) =0
```

5. Press *Enter* to select the expression and bring back the expression number

Now the block will only execute if the user enters the subform.

## How do I Execute Task Prefix/suffix Logic of a Subform Whenever the Subform is Refreshed?

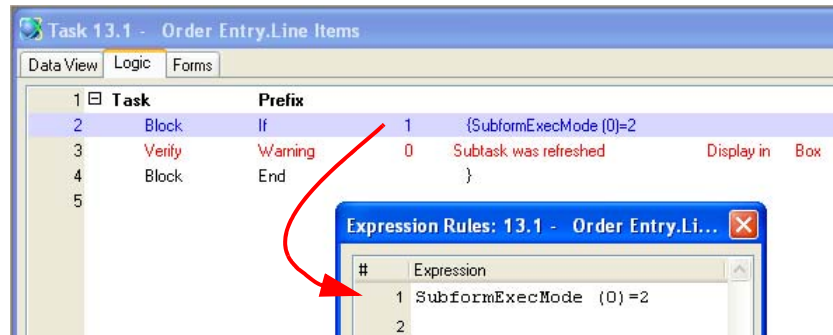
Task prefix executes whenever a task starts, and task suffix executes whenever the task ends. However, in a subform, you can have a *Block* operation within the task logic unit which executes depending on how the subform was entered. This is done using the **SubformExecMode()** function.

If the subform task was executed during a refresh operation, such as an automatic refresh or a manual **SubformRefresh** action, then **SubformExecMode()** will return 2. So using

```
SubformExecMode ( 0 ) = 2
```

as the condition on the block will cause the block to only execute when the subform is refreshed.

### Coding a block that only executes when the subform is refreshed



1. Open up a line in the logic unit you are modifying (**F4** or **Edit->Create Line**).
2. Select the *Block* operation by typing B or using the pulldown list.
3. Two lines will open up, a *Block If* and a *Block End*. Tab to or click on the field after the *If*.
4. Zoom (**F5**, double click) to the expression rules. Enter the expression:

```
SubformExecMode ( 0 ) = 2
```

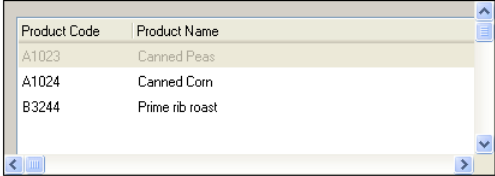
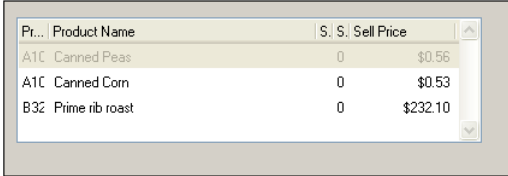
5. Press **Enter** to select the expression and bring back the expression number

Now the block will only execute when the subform is refreshed.

## How do I know which Autofit Option to use?

Each of the Autofit options works a bit differently, and it might be confusing which one you need. Below is an example of each of the three options, and how they work at runtime. The Subform control in all three cases is exactly the same size.

### Autofit Option Results

None	
	<p><i>None:</i> The subform task form fits just within the boundaries of the Subform control. If the called form is too big, it is cut off and scrollbars appear. For instance, if your Subform control is 80 units wide, and the called form is 150 units wide, you will only see 80 units of the called form.</p> <p>If you use placement on the Subform control so it resizes, you will see more of the subform task, but the controls within the subform task won't change.</p>
As Control	
	<p><i>As Control:</i> The subform task is fitted to the control size. This has the same effect as if the user resized the subform task. So, if you are using placement in the subform task, the controls will stretch or shrink accordingly.</p>

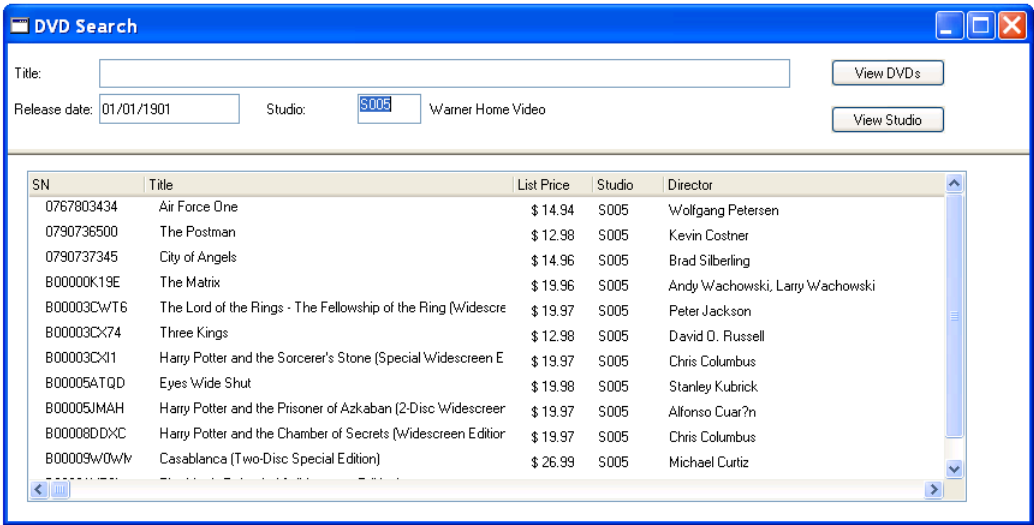
As Called Form																									
<div><table><tr><th>Product Code</th><th>Product Name</th><th>Supplier</th><th>Stock Quantity</th><th>Sell Price</th></tr><tr><td>A1023</td><td>Canned Peas</td><td></td><td>0</td><td>\$0.56</td></tr><tr><td>A1024</td><td>Canned Corn</td><td></td><td>0</td><td>\$0.53</td></tr><tr><td>B3244</td><td>Prime rib roast</td><td></td><td>0</td><td>\$232.10</td></tr></table></div>					Product Code	Product Name	Supplier	Stock Quantity	Sell Price	A1023	Canned Peas		0	\$0.56	A1024	Canned Corn		0	\$0.53	B3244	Prime rib roast		0	\$232.10	<p><i>As Called Form:</i> The Subform control sizes itself according to the subform task. So in this example, the Subform control has grown to contain the entire subtask form, even though the Subform control is the same size as the two examples above. In this instance, placement properties on the Subform control have no effect.</p>
Product Code	Product Name	Supplier	Stock Quantity	Sell Price																					
A1023	Canned Peas		0	\$0.56																					
A1024	Canned Corn		0	\$0.53																					
B3244	Prime rib roast		0	\$232.10																					



# Chapter 9: Splitter

## How do I Display Two Forms Using a Splitter?

When you are designing a screen, it is difficult sometimes to know how much screen real estate to give to each part. Often, in fact, the user will want to move one part of the form “out of the way” to show more of the other part. An easy way to give the user that kind of flexibility is to use the eDeveloper splitter.



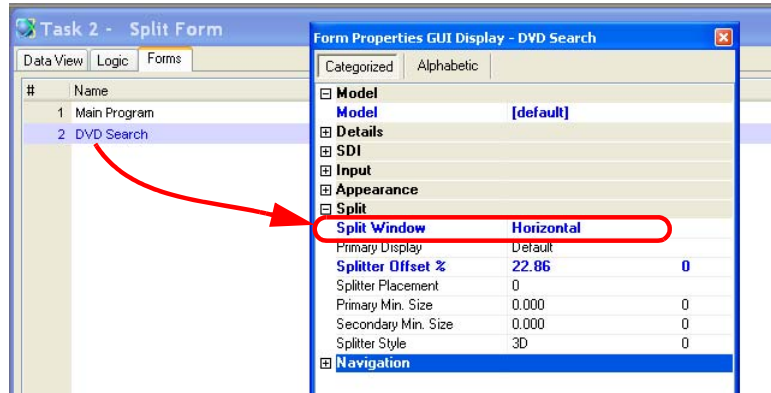
Split forms are two separate forms which act as one. In the example above, moving the bar in the middle of the form changes the ratio between the top and bottom areas.

Also, the task running at the bottom part of the form can change. Any number of tasks can be called. In this example, if the user presses the “View Studio” button, then instead of seeing the DVD list at the bottom of the screen, the Studio will be displayed.

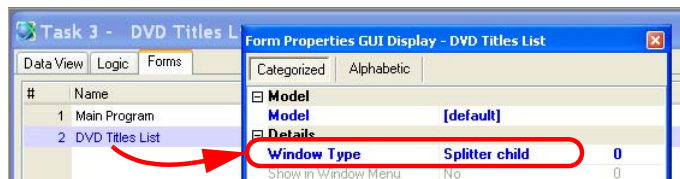
**Note:** The splitter and subforms overlap a bit in functionality, as both of them will display a separate form as “part of” the main form. However, they differ in that split forms involve dividing the main form in two with a straight line, while subforms are rectangles placed within the main form. Also, split forms can be resized by the end user, while subforms only resize in proportion to the main form being resized.

## Setting up a split form

1. In the parent task, set the **Form Properties->Split->Split Window** to *Vertical* or *Horizontal*. (Note that the form must be closed to set this property).

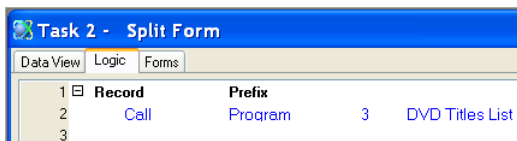


2. For the child task, set the *Form Properties -> Window Type* to *Splitter Child*. Note that the *Window Type* can also be set to an expression, so you can reuse this task for different purposes.



3. In the parent task, you also must explicitly call the child task. This could be as an event handler when a button is pressed, for instance. If you want the subform to initialize as soon as the parent task appears, you can call the task in Record Prefix. Below is a summary for how to do that.

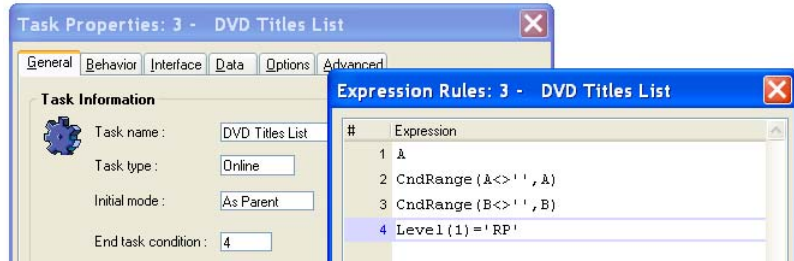
In the parent task, call the child task in Record Prefix





In the child task, set **Task->Properties->End Task Condition = Level(1)='RP'**.

This will display the task, then exit back to the parent task, when the parent task is first called.



Also in the child task, set **Task Properties ->Interface ->Close task window to No**.

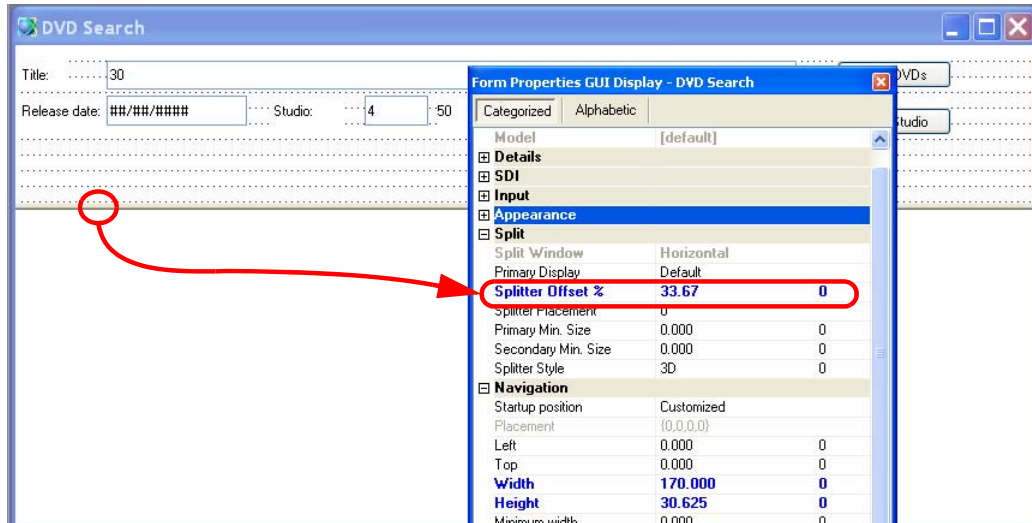


4. It is also useful, in a split form, to use the Placement properties so the tasks size according to how the splitter is moved.

**See also:** Chapter 8, “Subforms” on page 197.

Chapter 9, “How do I set the Initial Proportions of the Split Screens?” on page 214.

## How do I set the Initial Proportions of the Split Screens?



While the user can resize the split form as desired, you control the initial proportions of the split screen. There are several ways you can do this.

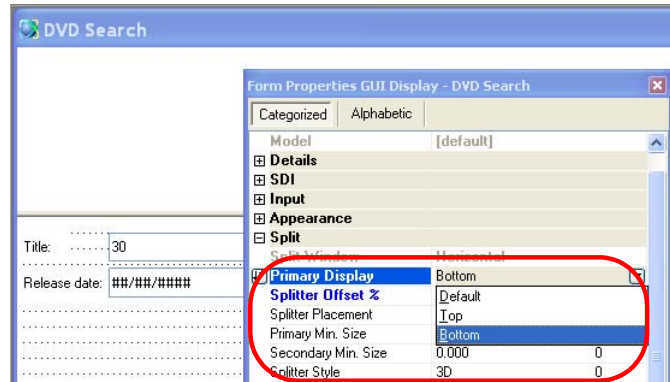
1. Manually type in the % you want in *Form Properties*-> *Splitter Offset%*.
2. Move the splitter bar to where you think it should be, by dragging it with the mouse. The *Splitter Offset%* will be updated to reflect the movement.
3. Use an Expression in the *Splitter Offset%* to set the offset at runtime.

## How do I set the Parent Task Display on the Opposite Size of the Split Screen?

When you set up a split screen, the primary display (i.e. the parent task) will have its placement depend on whether the project is “Right to Left” or “Left to Right”. However, you can force the display to be anything you like.

### Setting the parent display position

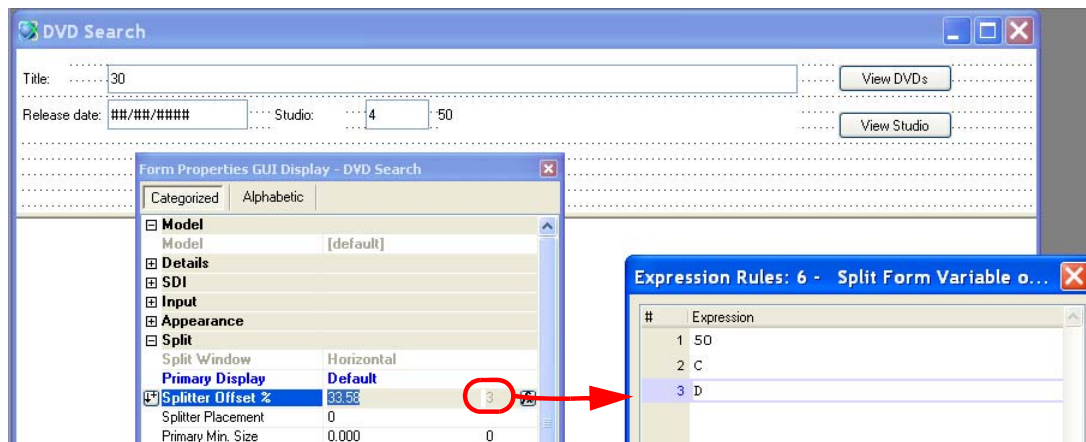
1. Go to **Form Properties --> Primary Display**.
2. Select the position you want for the parent task. In this example, we selected Bottom, so the parent task displays on the bottom, as shown.



## How do I Dynamically Set the Offset of the Splitter?

You will set the splitter offset manually, when you create the forms. However, you can also set it dynamically, using an Expression. In this example, we change the splitter offset depending on a variable.

### Setting a dynamic offset



1. Go to **Form Properties -> Splitter Offset**
2. Zoom from the **Expression** column (or press the **fx** button) to enter the expression you want.

**Note:** The splitter offset is re-evaluated upon re-entering the task, or when you update the variables that make up the expression, or when you switch between records.

---

## How do I Obtain the Current Offset of the Splitter?

The user can change the splitter offset at runtime, or it may be changed based on expressions. You can query the value of the current offset using the **SplitterOffset()** function.

`SplitterOffset(0)` returns the percentage offset. `SplitterOffset(1)` returns the absolute offset, which will be in whatever units the form uses (dialog units, centimeters, or inches).

## How do I Keep the Splitter Location as Set at Runtime by the End-User?

There are a lot of ways the user can customize a form in eDeveloper. It is very convenient if the user can keep that customization from session to session.

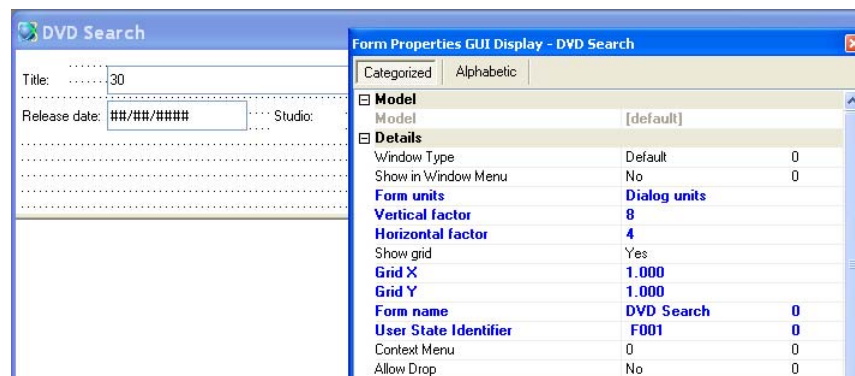
eDeveloper will in fact keep the user settings, such as the splitter location automatically. All you have to do is give the form a **User State Identifier**. The User State Identifier is just an alpha string, and you can either hard-code it or enter it as an expression to be evaluated at runtime. eDeveloper uses it to store the end-user form settings. It works as follows:

- If the User State Identifier is blank, eDeveloper does not store any of the user state information.
- If the User State Identifier is not blank, eDeveloper stores the data using the string as the identifier for the state information.
- So, if each form has one unique identifier, each form's state will be stored for each user. This is probably the simplest way to implement this feature.
- If two forms share the same identifier, they will share the user state information. Suppose form A and form B share the User State Identifier of "F001". Then if the user sets form A to a 75% split, then when the user opens form B, it will also be set to a 75% split.
- One form can also have two (or more) identifiers. That is, suppose form C is sometimes used for one purpose, and has an identifier of "F001A". But for a different purpose, it is "F001B". Then the end user could set two different splitter values, depending on how the form is being used currently.

In any event, here is how you set the User State Identifier.

**Note:** While we are talking about splitter location here, the User State Identifier is used to store other state settings also, such as form dimensions and column widths.

### How to set a User State Identifier



1. Open your parent form.
2. In **Form Properties->User State Identifier**, type in a unique string (here, it is "F001").

Alternatively, you can specify the string using an expression, by tabbing to the field to the right and pressing zoom to enter an expression.

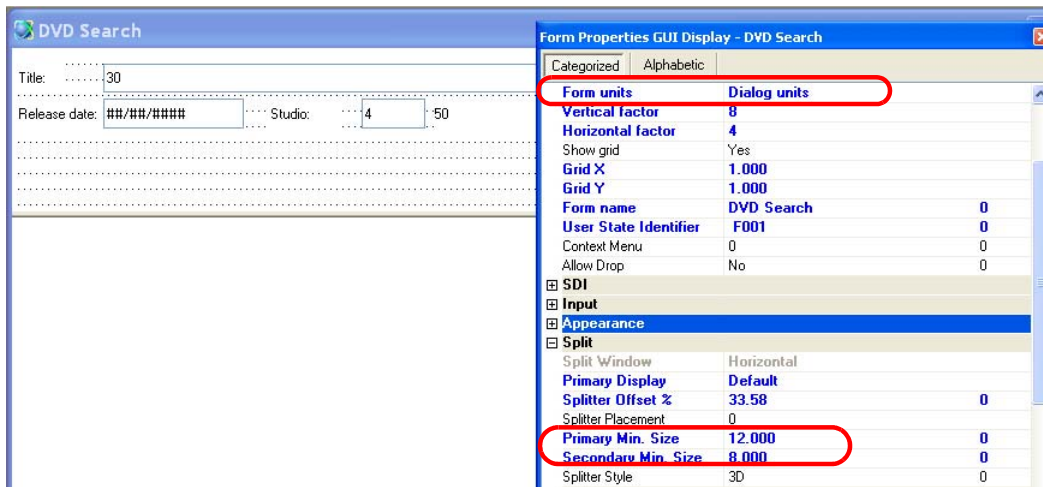
**Hint:** While this works in Runtime, the user state information is not kept in the Studio. So if you want to test the concept, create a Cabinet file and run your application in Runtime.

## How do I Set a Minimal Size of the Split Areas?

By default, the end user can resize the split areas until one area takes over the entire screen. If you want to stop the resizing at a certain point, you can do that by setting a minimal size for either the primary or secondary screen.

The minimal size value is absolute, not a percentage, and is determined by whatever **Form Units** the form is using. In this example, *Dialog Units* are used.

### Setting a minimal size



1. Open up your parent form.
2. Go to Form Properties.
3. Set the Primary Min. Size to the smallest size you want the primary to resize to.
4. Set the Secondary Min. Size to the smallest size you want the child form to resize to.

Now, in this example, the parent form (on top) cannot be resized smaller than 12 dialog units. The child form (on bottom) cannot be resized smaller than 8 dialog units.

**Note:** You can also use an Expression to set the minimum sizes.

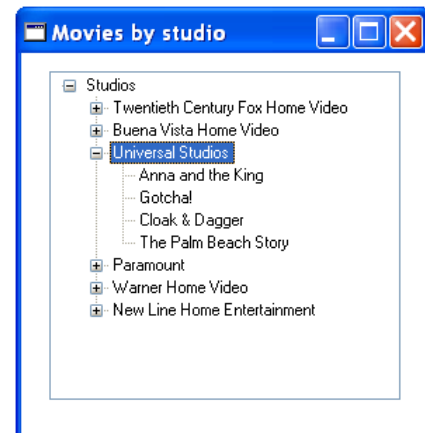


## Chapter 10: Tree Control

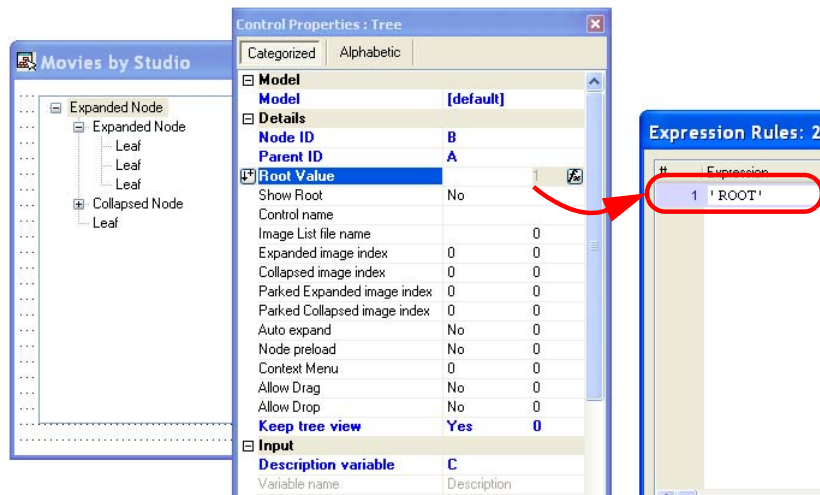
### How do I Display Data In A Tree-format?

The tree format is a convenient way to display lists of data. It has become popular in the Windows environment, and users are familiar with it.

eDeveloper has an easy-to-use tree control that you can use as needed. All you have to do is store your data in a hierarchically designed table (a memory table will do fine) and set the options you want. Here are the basic steps.



## Implementing a tree control



1. Store your data in the correct format for a tree. See Chapter 10, “How do I Properly Define an Hierarchical Data Source that Will Fit a Tree Control Display?” on page 223.
2. Create the Data View of the task, using the Tree data source as the Main Source, and selecting the columns that you need for the task (Node, Parent, etc.)
3. Select the tree control from the Controls palette and drop it on the form. Size it as desired.
4. In the *Control Properties* for the tree, set the following options
  - **Node ID:** The key for this particular record.
  - **Parent ID:** The key for the parent of this record.
  - **Root Value:** The very top record (here named ‘ROOT’).
  - **Description variable:** The value that displays on the tree.

That is all you need to do. If the data is correctly formatted, it will display as a table. There are other options you can set too, to control the look of the table and how it works in runtime; these are covered in other sections of this How-To.

**See also:** Chapter 10, “How do I Properly Define an Hierarchical Data Source that Will Fit a Tree Control Display?” on page 223.  
 Chapter 10, “How do I Set Icons for the Tree Nodes?” on page 224.  
 Chapter 10, “How do I Automatically Open the Tree with All Its Nodes or only Several Nodes Expanded?” on page 230.

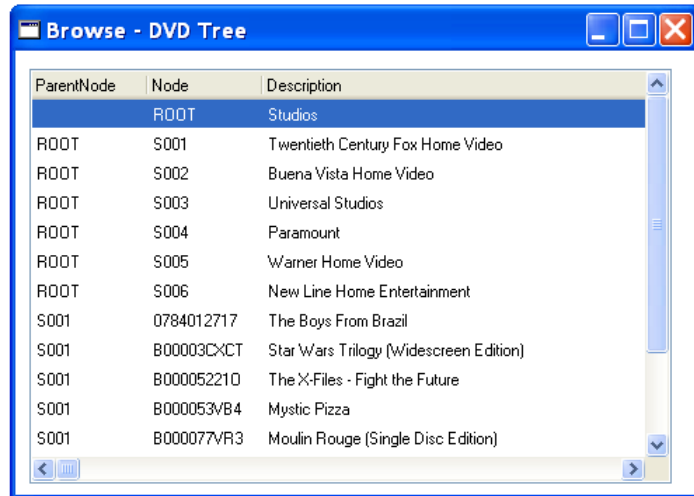
## How do I Properly Define an Hierarchical Data Source that Will Fit a Tree Control Display?

To get a tree to display, you need to have the data structured correctly. In order to do this, you may need to create a memory table to temporarily hold the data.

The key to the structure are two fields: the Parent ID and the Node ID. The Node ID is what identifies this particular record, and it will probably be the unique identifier of a record. The Parent ID identifies the parent of this record.

### Recursion

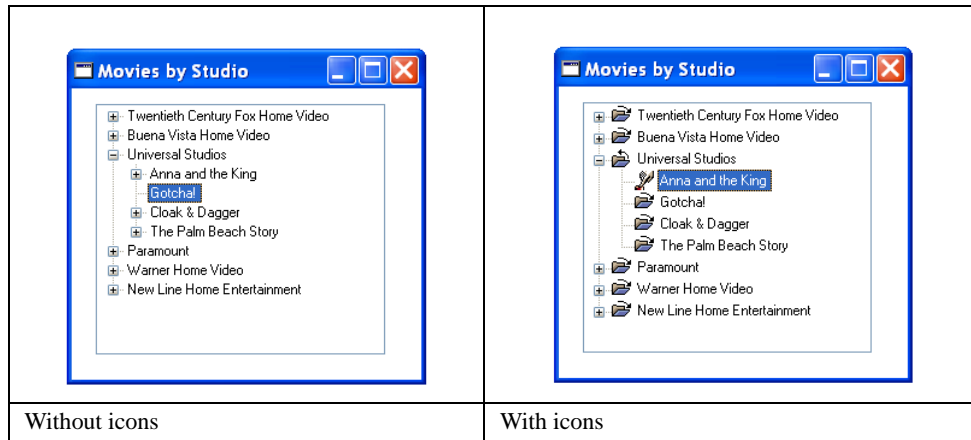
You must be sure that the Parent ID and Node ID of each record are different. Otherwise you will get a recursive tree, where one branch can open an infinite number of times. In other words, the node “S002” cannot also have a Parent Node of “S002”. This is usually only a problem with the Root node, because there is a tendency to leave both the Parent Node and Node fields blank



ParentNode	Node	Description
	ROOT	Studios
ROOT	S001	Twentieth Century Fox Home Video
ROOT	S002	Buena Vista Home Video
ROOT	S003	Universal Studios
ROOT	S004	Paramount
ROOT	S005	Warner Home Video
ROOT	S006	New Line Home Entertainment
S001	0784012717	The Boys From Brazil
S001	B00003CXCT	Star Wars Trilogy (Widescreen Edition)
S001	B000052210	The X-Files - Fight the Future
S001	B000053VB4	Mystic Pizza
S001	B000077VR3	Moulin Rouge (Single Disc Edition)

**Hint:** It is likely that a tree might contain data from several different tables. In our example, we used data from the “studio” table and from the “DVD” table. Since each of these tables has very different data, we just collected the minimum information to display onscreen. If we wanted to display more information, we would link to the original record rather than trying to move it into this temporary table.

## How do I Set Icons for the Tree Nodes?



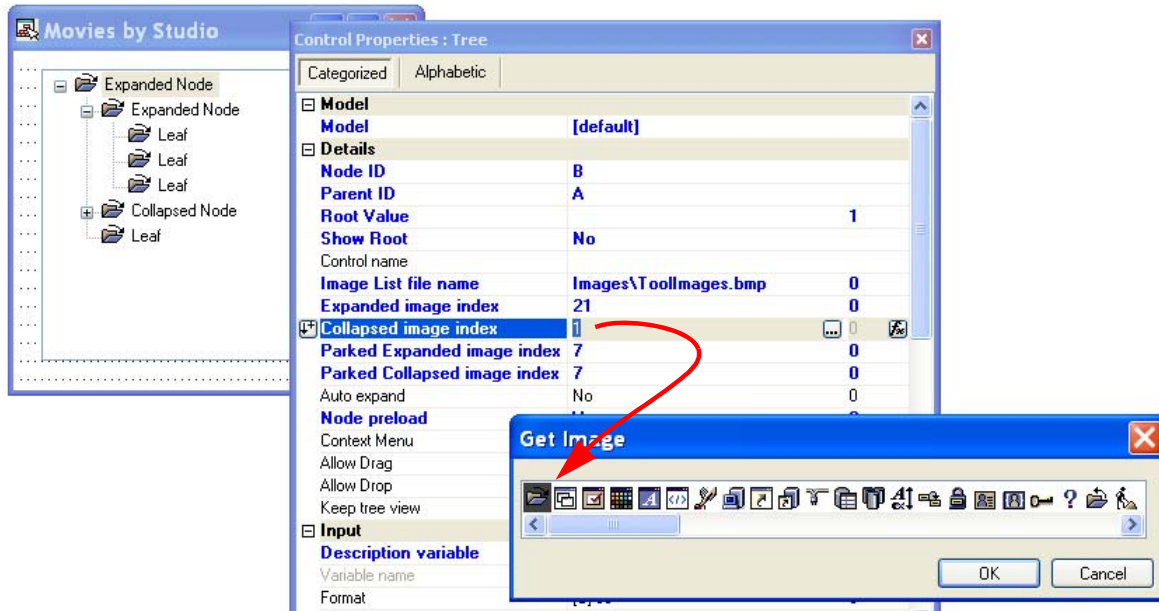
You can choose whether or not to use icons on your tree control.

The tree has four states:

- **Expanded image:** When a node is fully expanded, or has no children.
- **Collapsed image:** When a node is collapsed.
- **Parked expanded image:** When the cursor is parked on an expanded node.
- **Parked collapsed image:** When the cursor is parked on a collapsed node.

Each of these icons is set by specifying a number, which is an index into the icon file. If the index is set to zero, then no image appears on a node in that state.

## Using icons

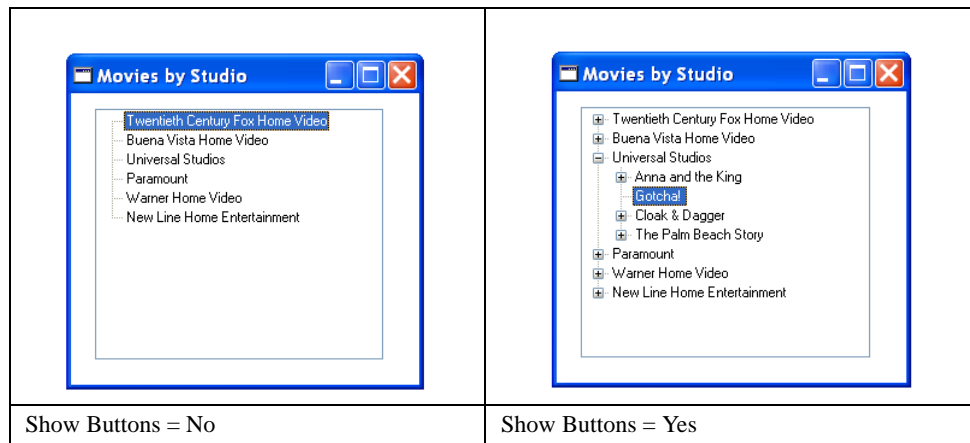


**Prerequisite:** You need to have an icon file. An icon file is a bitmap file with a series of icons in it. You will select which icons you want by number. There are icon files available for purchase, or you can design your own.

1. Select the tree control
2. Go to **Control Properties -> Image List file name.**
3. Zoom to select the image file, or type in the name.
4. Now, for each of the four states, *Zoom* to select the icon you want from the image file.

**See also:** You can find an example icon file at %EngineDir%\MGresrc.Toolbar\_Images

## How do I Show\hide the Expand\collapse Buttons?



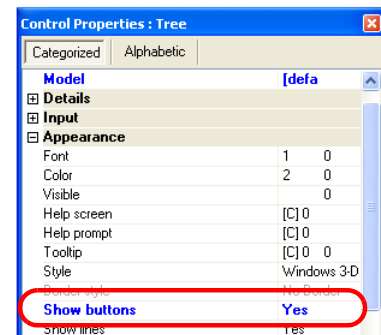
By default, each node of a tree has a small '+' or '-' button next to it. Clicking on a '+' will expand a node, and clicking on a '-' will collapse the node.

You can turn these buttons on and off using the *Show Buttons* property of the tree control.

### Turning on/off the expand/collapse button

1. Select the tree control
2. Go to **Control Properties -> Show Buttons**.
3. Set the value to *Yes* if you want the buttons, *No* otherwise.

**Note:** If *Show Buttons* = *Yes*, then by default leaves of the tree -- that is, the items with no sub-items -- will show up with a "+" sign when the tree first appears. If you want to avoid this, set **Node Preload** to *Yes*. See Chapter 10, "How do I Set the Tree Control to Display the Expand Button only in the Relevant Nodes that Actually Have Child Nodes?" on page 233.



## How do I Add a New Child Node at Runtime?

When the user is working with a tree, much of the functionality is the same as working with a table. However, in a tree the data is not flat. So when the user wants to “add a line”, that line can either be a sibling node or a child. In either case, it is up to you as the programmer to be sure the parent node and current node fields are initialized correctly when a new node is created.

### Creating a Child Node

Task 43 - Tree, Add node						
Data View Logic Forms						
1	Event	Create Child				Scope: Task
2	Update	Variable	C	Parent for Init	With: 2	TreeValue(0)
3	Update	Variable	D	Node for init	With: 4	GetNextRecNum ('DVD')
4						

1. Set up a button that raises the event *Create Child*, and label it “Create child” or whatever is appropriate to the application. “Create child” does not exist by default on the overhead menu, so if you want to make it a runtime option, you’ll need to add it.

Task 43 - Tree, Add node						
Data View Logic Forms						
1	Main Source	7	DVD Tree	Index:	1	
2						
3	Virtual	1	Parent for Init	Alpha	10	
4	Virtual	2	Node for init	Alpha	10	
5						
6	Column	1	ParentNode	Alpha	10	Init: 5 Parent for Init
7	Column	2	Node	Alpha	10	Init: 6 Node for init
8	Column	3	Description	Alpha	60	
9						

2. Create two variables somewhere above the Parent and Node variables, in the *Data View*, which will be used in an Init to initialize those two fields if a new record is created.
3. Create a handler for the *Create Child* event, with *Propagate* set to Yes.
4. Within the handler, update the initialization fields appropriately so the new record will be a child. That means the Parent field should have the value of the current field, which can be automatically obtained by using the function **TreeValue(1)**.

You also need to initialize the Node field with a unique value. In our example, we used a function we wrote that will automatically fetch a unique key for us.

Now, the new node will be initialized appropriately, and the user can fill in the rest of the data as desired.

## How do I Add a New Sibling Node at Runtime?

When the user is working with a tree, much of the functionality is the same as working with a table. However, in a tree the data is not flat. So when the user wants to “add a line”, that line can either be a sibling node or a child. In either case, it is up to you as the programmer to be sure the parent node and current node fields are initialized correctly when a new node is created.

### Creating a Sibling Node

Task 43 - Tree, Add node							
Data View		Logic		Forms			
1	Event	Create Line				Scope:	Task
2	Update	Variable	C	Parent for Init	With:	3	TreeValue(1)
3	Update	Variable	D	Node for init	With:	4	GetNextRecNum ('DVD')
4							

1. Set up a button that raises the event *Create Line*, and label it “Create sibling” or whatever is appropriate to the application. The user can also press **F4** or select **Edit->Create Line**, which will add a sibling node, but that might not be obvious to the user.

Task 43 - Tree, Add node							
Data View		Logic		Forms			
1	Main Source	7	DVD Tree	Index:	1		
2							
3	Virtual	1	Parent for Init	Alpha	10		
4	Virtual	2	Node for init	Alpha	10		
5							
6	Column	1	ParentNode	Alpha	10	Init:	5
7	Column	2	Node	Alpha	10	Init:	6
8	Column	3	Description	Alpha	60		Parent for Init
9							Node for init

2. Create two variables somewhere above the Parent and Node variables, in the *Data View*, which will be used in an Init to initialize those two fields if a new record is created.
3. Create a handler for the *Create Line* event, with *Propagate* set to *Yes*.
4. Within the handler, update the initialization fields appropriately so the new record will be a sibling. That means the Parent field should have the value of the parent, which can be automatically obtained by using the function **TreeValue(1)**.

You also need to initialize the Node field with a unique value. In our example, we used a function we wrote that will automatically fetch a unique key for us.

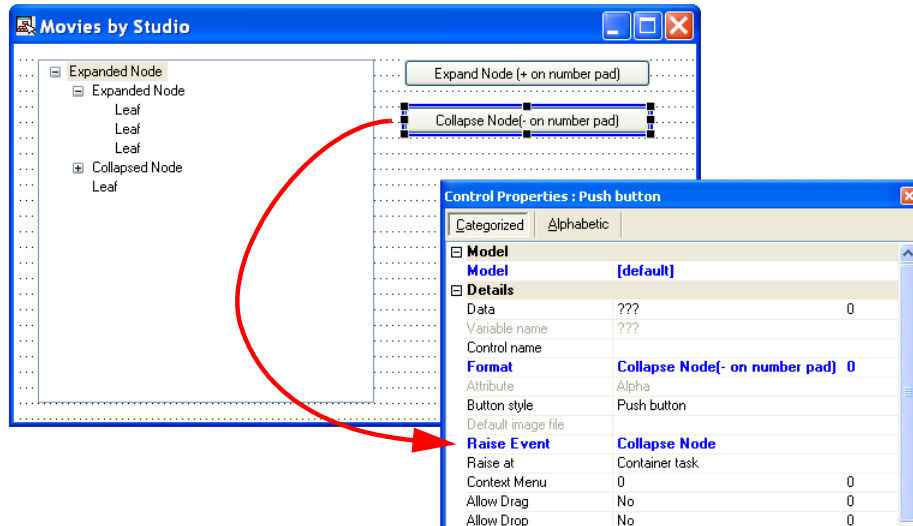
Now, the new node will be initialized appropriately, and the user can fill in the rest of the data as desired.



## How do I Explicitly Expand\Collapse Tree Nodes at Runtime?

When the user wants to expand or collapse a node on a tree, they can simply click on the + or - sign next to the node. However, if you want to expand or collapse a node explicitly, by raising an event, you can do that by using the **Expand Node** and **Collapse Node** events.

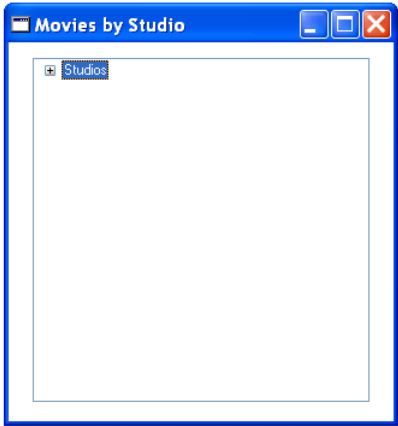
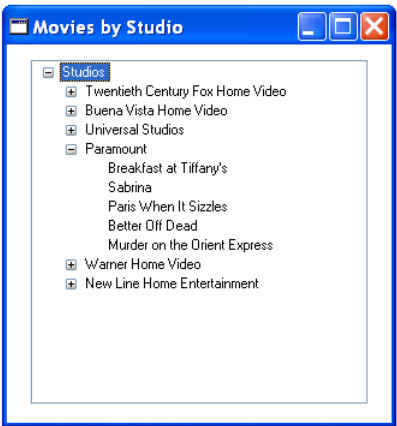
### Raising an Expand or Collapse Node event



You can raise these event like you would raise any others, using a Raise Event operation in a logic unit, or the Raise Event property on a push button.

Either event works on the node the user is currently parked on.

## How do I Automatically Open the Tree with All Its Nodes or only Several Nodes Expanded?

	
<i>Auto Expand</i> = No	<i>Auto Expand</i> set to an expression which opens the root and one node

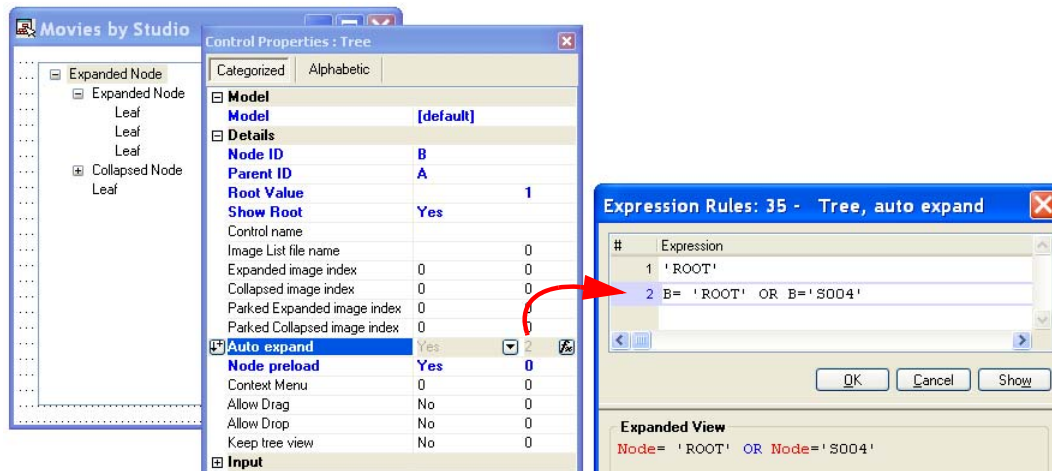
By default, when a tree opens none of the nodes are expanded, as shown in the example on the left. The user can double click on a node, or press the '+' button, to expand the tree as needed.

However, you can selectively expand the tree when it opens. This is done using the *Auto expand* control property.

If *Auto expand* is set to Yes, all the nodes will expand. However, it can also be set to an expression to selectively expand nodes. The example on the right expands two nodes: 'ROOT' and 'S004'.

This feature can be used with **TreeNodeGoto()** to open the tree and position the cursor for the user.

## Setting Auto expand to a specific node



1. Select the tree control
2. Set *Control Properties* -> *Auto expand* to Yes. This will expand *all* the nodes when the tree opens.

Alternatively, you can use an expression to selectively expand nodes. In this example, we expand the node when the node ID is 'ROOT' or 'S004'. This causes the root and one node to expand, as shown in the example at the top.

## Setting Auto expand to a tree level

Alternatively, you can use the **TreeLevel()** function to expand only certain tree levels.

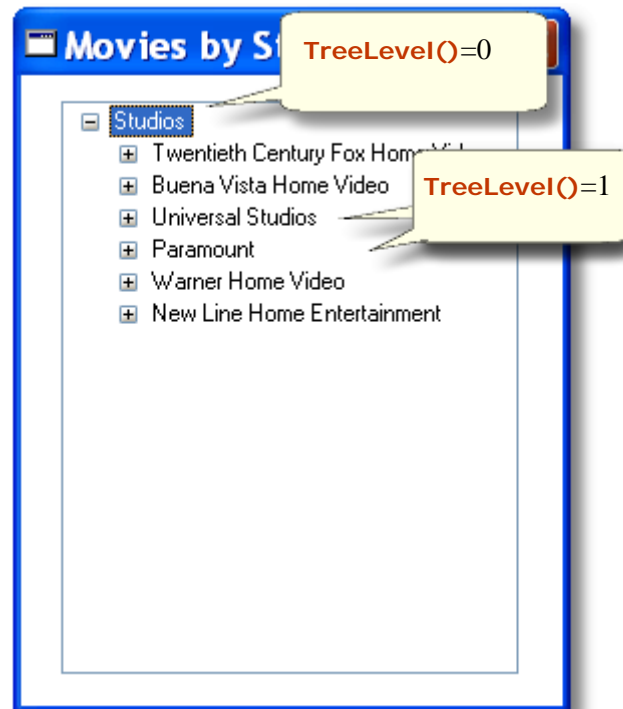
In this example, we only expand the top level of the tree, by using the expression

```
TreeLevel()=0
```

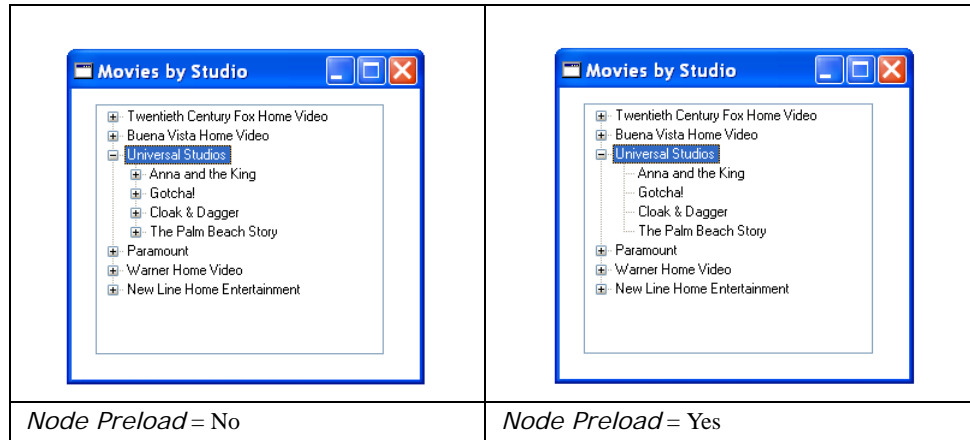
for the *Control Properties* -> *Auto expand* property.

We could have expanded the top two levels instead, by using the expression

```
TreeLevel()=0 or TreeLevel()=1
```



## How do I Set the Tree Control to Display the Expand Button only in the Relevant Nodes that Actually Have Child Nodes?



By default, each node of a tree has a small '+' or '-' button next to it. Clicking on a '+' will expand a node, and clicking on a '-' will collapse the node. However, initially all the nodes will have a '+' in front of them, because the engine has not fetched any of the subrecords and so does not "know" if any exist. Once the user clicks on the node, the '+' will disappear if there are no subrecords.

You can change this behavior though, by instructing the engine to preload all the records in the tree. Then the leaf nodes will be correctly marked, with no expand button.

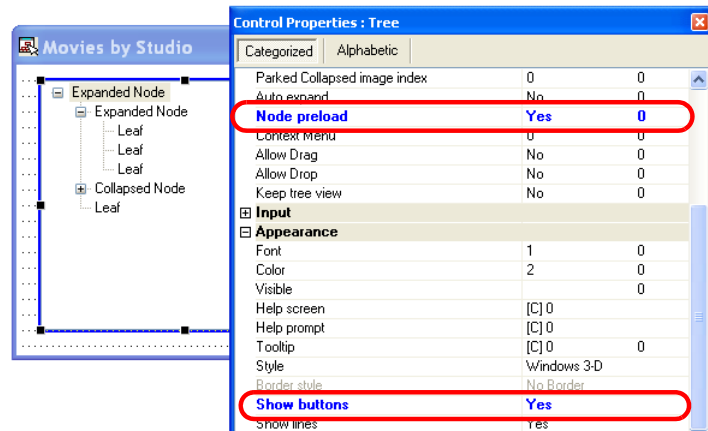
### Turning on preload

1. Select the tree control
2. Set *Control Properties* -> *Node Preload* to Yes

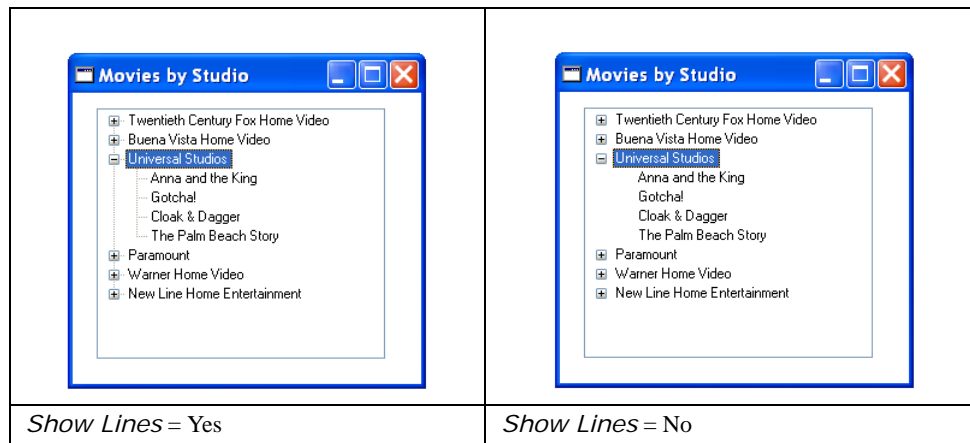
Now the expand and collapse buttons will only show when they are relevant.

**Note:** Setting Node Preload to Yes might make the tree take perceptibly longer to load if there are a lot of records.

**See also:** Chapter 10, "How do I Show/hide the Expand/collapse Buttons?" on page 226.

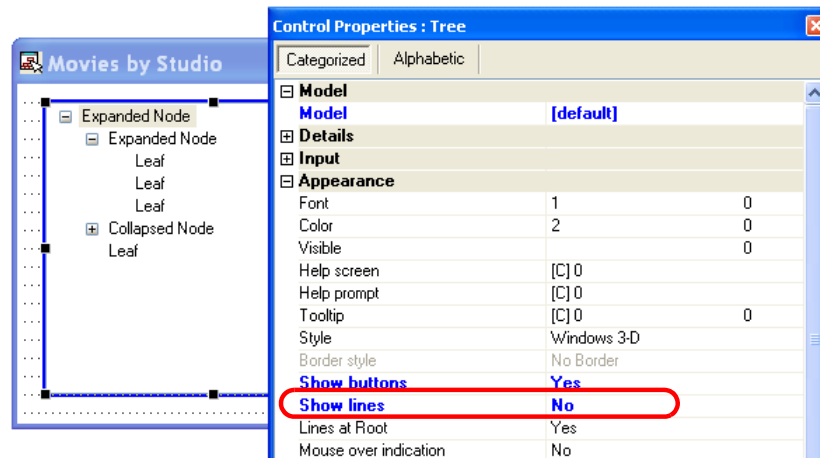


## How do I Show\Hide the Connecting Lines of the Tree Control?



By default, each item in the tree is connected with a dotted line. You can turn these lines on and off using the *Show Lines* property.

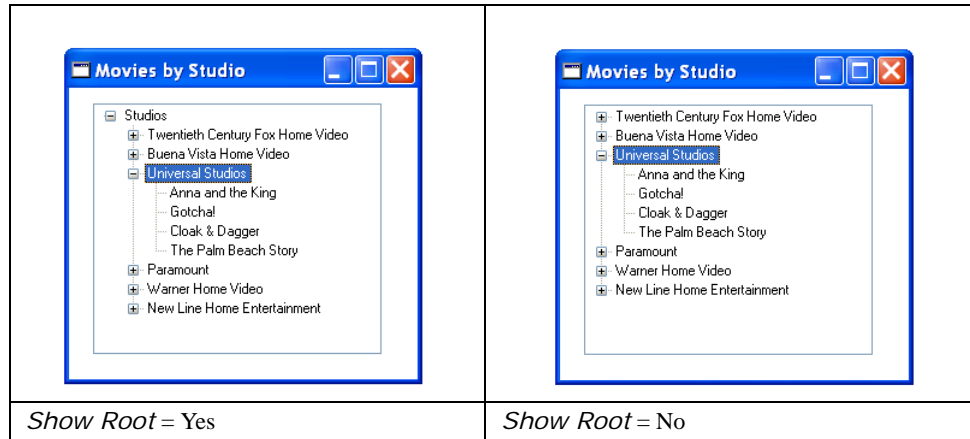
### Turning connecting lines on and off



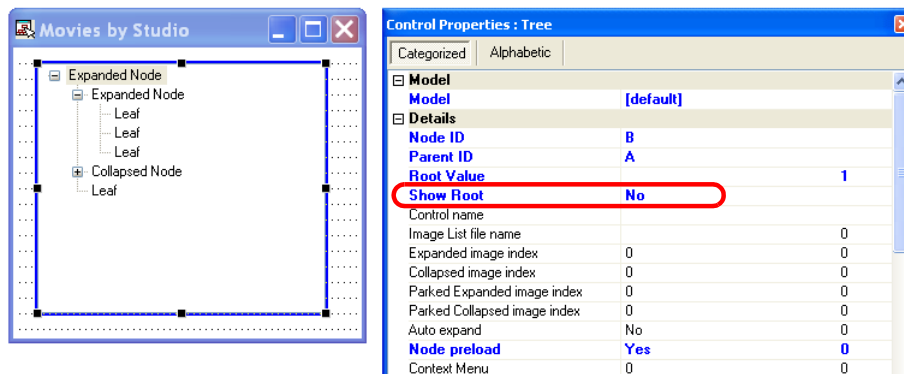
1. Select the tree control.
2. Set *Control Properties* -> *Show Lines* to *No* if you don't want connecting lines, or *Yes* if you do want connection lines.

You will see the lines turn on and off in the tree control as soon as you change the property.

## How do I Show\Hide the Root Node as Part of the Tree Content?



### Turning show root on and off

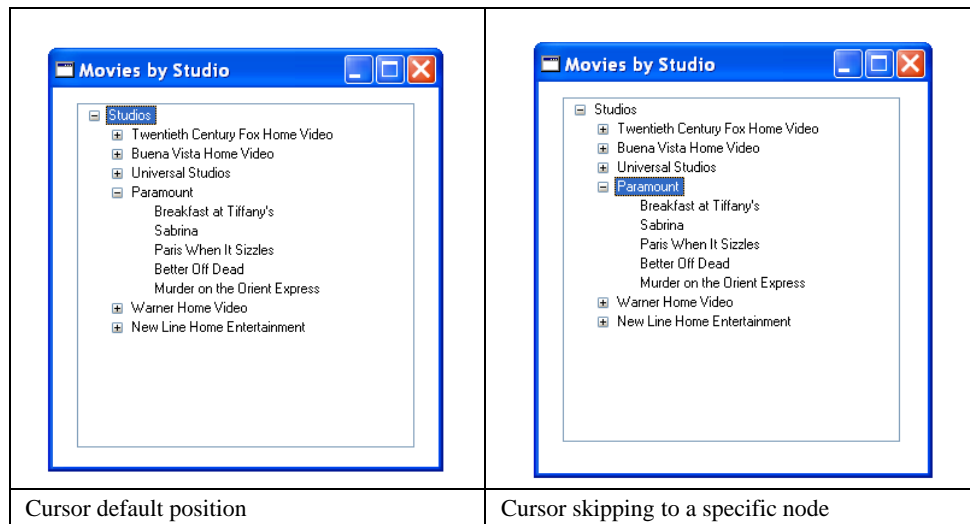


1. Select the tree control.
2. Set *Control Properties* -> *Show Root* to *No* if you don't want the root level to appear, or *Yes* if you do want the root to appear.

You will see the results the next time you view the tree in runtime.

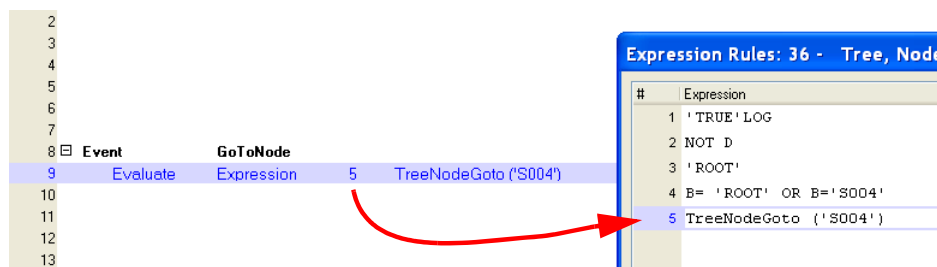
**Note:** The root record must actually exist, whether or not you are actually going to show it onscreen, or you will get an error message.

## How do I Skip to a Specific Tree Node?



By default, when a tree opens the cursor is parked on the top node, as shown on the left. However, you can force the cursor to move to any given node. This is done using the **TreeNodeGoto()** function.

### Using TreeNodeGoto()



1. Enter an event where you want the tree to expand.
2. Create an Evaluate Expression operation for the expression:

`TreeNodeGoto (Node)`

Where **Node** is the value of the node you want to go to. In our example, the studio “Paramount” has an ID of “S004”, so that is the node we want to go to. Note that we don’t need to know what level that particular node is at.

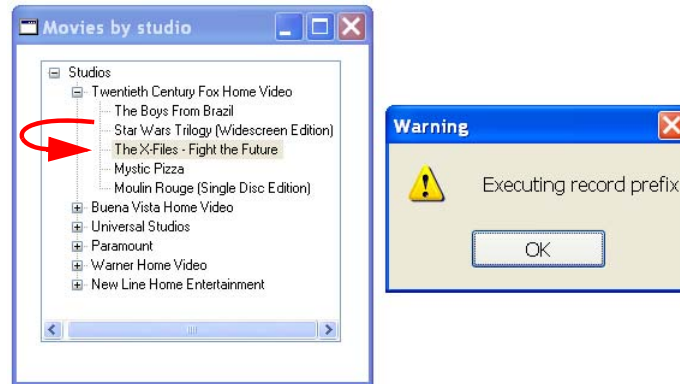
**Hint:** You can use this function in Task Prefix to park the focus on a certain node. You can also use it in Record Prefix, but you should use it with a condition of *IsFirstRecordCycle* (0) or *Wait=No*, or it may cause a loop.



**See also:** Chapter 10, “How do I Explicitly Expand\Collapse Tree Nodes at Runtime?” on page 229.



## How do I Respond to the End-user Movement from one Node to another?



Sometimes you will want to perform some actions when the user moves from one node to another. This is very easy when you remember that each node in the tree is actually one record in a table. The usual record events ... *Record Prefix* and *Record Suffix* ... work just as they do with any other record. The records are simply displayed in a different format.

### Capturing movement entering a node

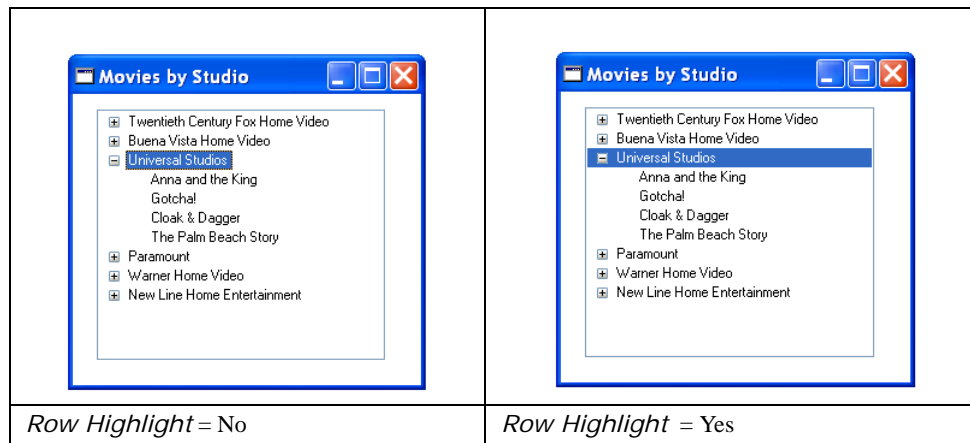
1	Record	Prefix				
2	Verify	Warning	0	Executing record prefix	Display in	Box
3						

- To capture the movement before the cursor hits a node, put your operations in *Record Prefix*.

### Capturing movement upon leaving a node

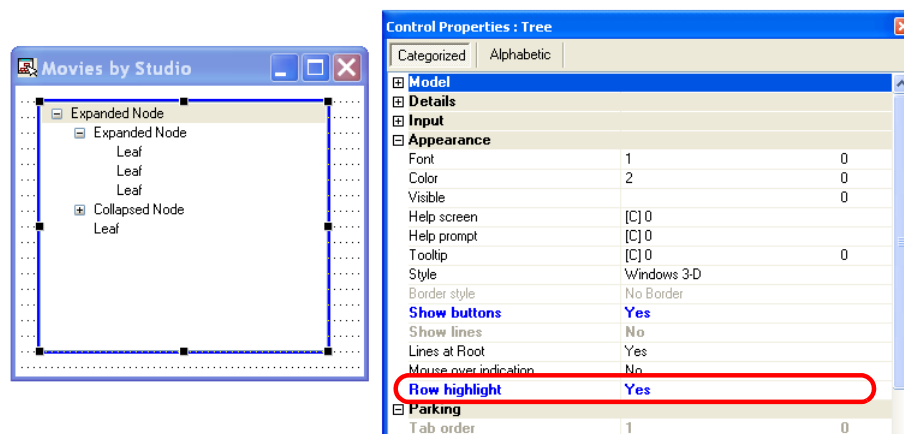
- Record Suffix* is only executed when the data view is changed.
- If you need to capture the movement in leaving a node when the record is not changed, you need to also set the *Task Properties* -> *Behavior* -> *Force Record Suffix* flag to *Yes*.

## How do I Highlight the Entire Line of the Current Tree Node?



When you select a node in the tree, it is highlighted. By default, the highlight only extends to the edge of the node text. However, you can also choose to extend the highlighting across the entire tree. This is controlled with the *Row Highlight* property.

### Turning row highlight on and off



1. Select the tree control.
2. Set *Control Properties* -> *Row highlight* to *Yes* if you want the highlight to extend across the entire tree, *No* otherwise.

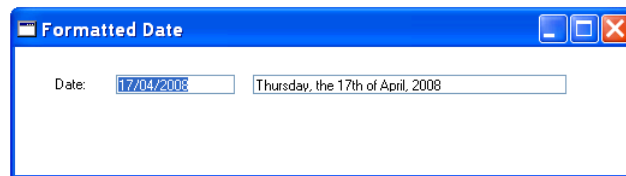
You will see the row highlight change in the tree control as soon as you change the property.

**Note:** This feature is only available when *Show lines* is *No*.

## Chapter 11: More on Logic

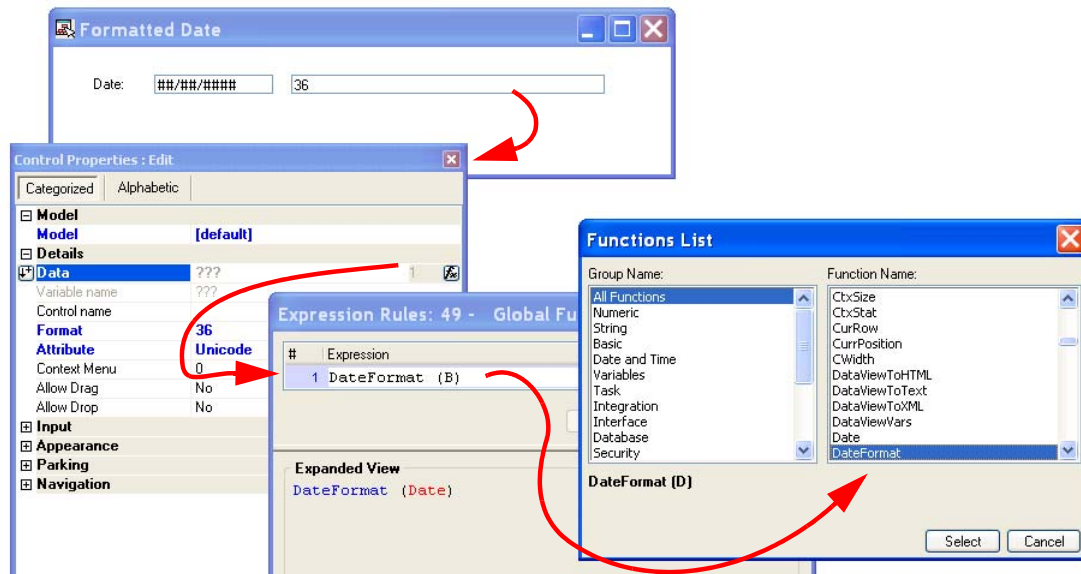
---

### How do I Create a Function?



In eDeveloper, a function is just a special type of logic unit. Functions use the same input parameters, local variables, and the same operations that are available for the other logic units. One main difference is that

they have a return value, so that when a function is used in an expression or on a form, the value can be used directly in the expression without using a variable to hold a parameter.



For instance, in this example, we have a function that formats a date into a complex formatted string for reports. The formatted string can be used directly on the form, by using the function in an expression.

The other difference is that your functions show up on the Functions List right along with all the built-in eDeveloper functions, complete with a list of the expected parameters. This makes them easy to find and to use. This also means that if you name your function the same as a built-in function, you can override the built-in function. So if you want to extend or change a built-in function, you can.

Below is a summary of how to create a function.

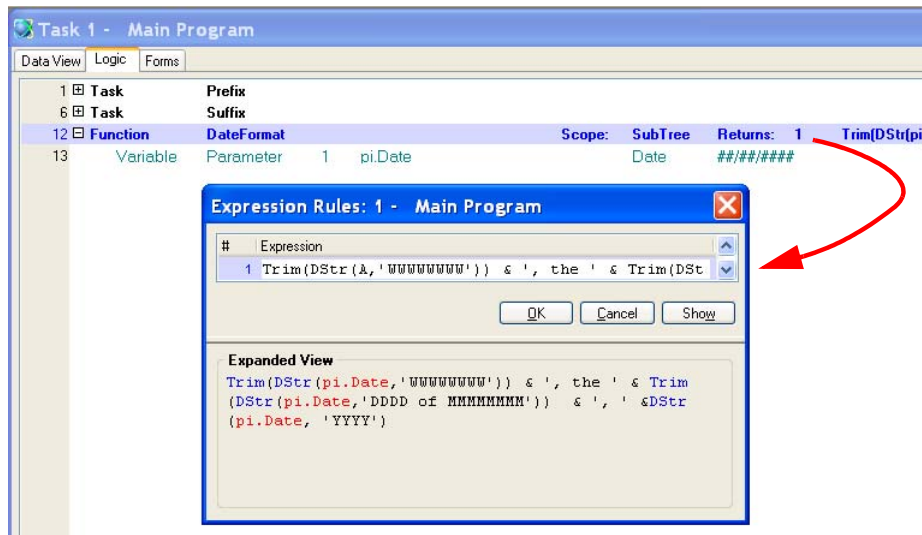
## Function Scope

The scope of a function depends on where it is located, and on the value of the *Scope* property of the function.

- If the function is in a task and the *Scope* is set to *Task*, then the function will only be visible for that task.
- If the function is in a parent task, and the *Scope* is set to *Subtask*, then the function will be visible for all subtasks. Thus, if the function is in the Main Program, then it will be visible for the entire application.
- If the function is part of a component, then it can be shared across many projects.

**See also:** Chapter 11, “How do I Create a Function That is Available for the Current Task Only?” on page 247  
Chapter 11, “How do I Create a Function That is Available For the Entire Project?” on page 248

## Creating a Function



1. Go to the *Logic* section of your task.
2. Press *Ctrl+H* to open up a header line.
3. Type **F** to make this a function. The cursor will move to the right, into the function name field.
4. Type in your name for this function. You can use any name, but if you use the name of an existing eDeveloper function, then you will override the function provided by eDeveloper.
5. Create your input and output parameters, if any (see Chapter 11, “How do I Set the Function’s Parameters?” on page 244 for details on how to do this).
6. Create your return values (see Chapter 11, “How do I Set the Return Value of a Function?” on page 246 for details on how to do this).

Now you have created a function! The function will show up on the function list with all the built-in eDeveloper functions, which makes it easy to use.

## How do I Set the Function's Parameters?

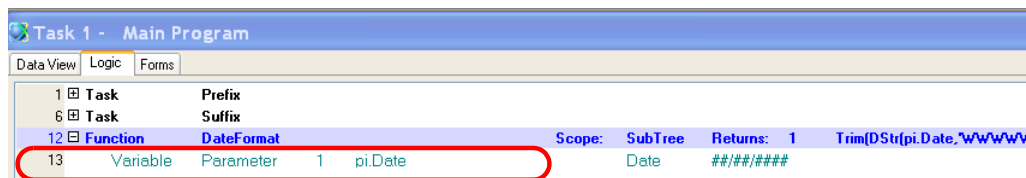
The parameters for a function work just like the parameters for a task or event handler. They are variables, with a type of *Parameter*.

To create parameters for a function, you just create however many Parameter-type variables you need, in the order you want them. Once you create them, they will show up on the Function List represented by letters, to make them easy to use.

Note that the parameters used by a function are strictly *input* parameters. Because functions are used in expressions, the data is always sent by reference and cannot be changed by the function. If you want to pass any information back, you need to use the *Returns:* field (see Chapter 11, “How do I Set the Return Value of a Function?” on page 246).

**Hint:** If you forget to set the variables to type *Parameter*, then they will still work, but will not show up on the Function List. If you have functions whose parameters don't show up, that is likely the reason.

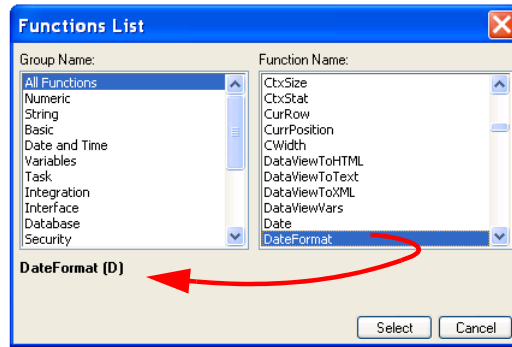
### Creating parameters for a function



1. Go to your Function logic unit.
2. Press **F4** (**Edit->Create line**). A blank line will appear below your cursor position, and your cursor will be located on a field at the left side of the new line.
3. Type **V**. The word “Variable” will appear and your cursor will move to the right.
4. Type **P**. The word “Parameter” will appear and your cursor will move to the right.
5. Type in the name of your parameter, then tab to the right.
6. Select the model for this field, or set the data type and other properties manually.



7. Repeat for however many parameters you need.



Now your function can accept parameters. When your function shows up on the Functions List, you will see letters representing the data type if each of the parameters, as shown above.

## How do I Set the Return Value of a Function?

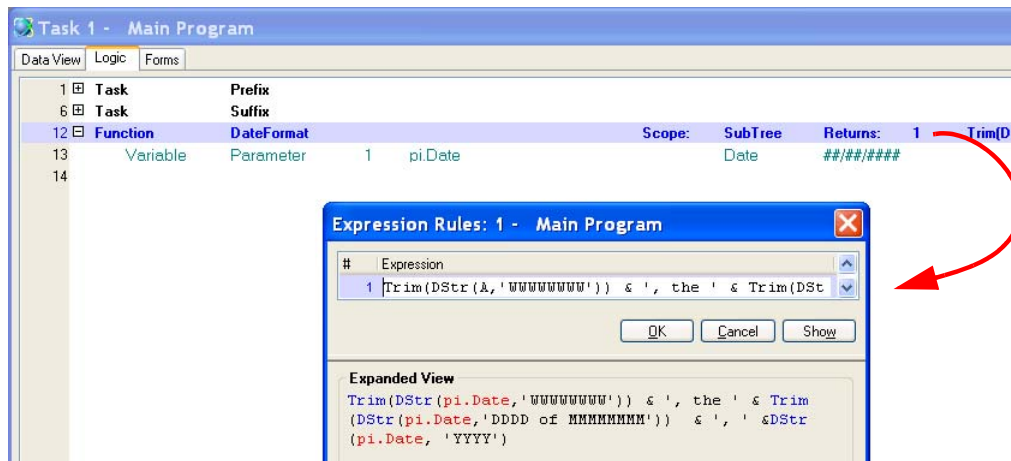
One of the main advantages of functions is that they can have a return value. This value can be used directly on a form or as a parameter, or be nested inside other functions, without the use of intermediate variables, which makes for very efficient coding.

Also, because functions are always called in an expression, the input parameters are purely *input*. Any changes you make to parameters in the function are not passed back to the caller, so if you want to pass back any data you need to do that in the return value.

Still, the use of a return value is purely optional. You are not required to send back any value.

Note that you can send back any data type for your function. It is up to you to use the function correctly. However, if you use, say, a function that returns a string in a numeric field, the syntax checker will report it as an error (Attribute mismatch).

### Setting a return value for a function



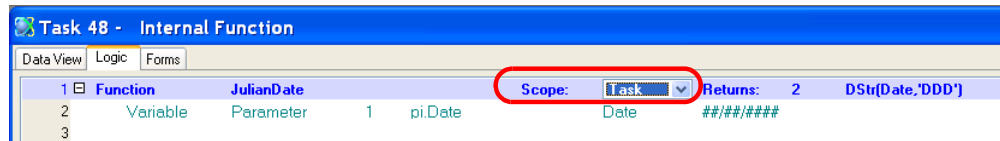
1. Go to your function.
2. Move to the Returns: field, by clicking on it or tabbing to it.
3. **Zoom** (F5 or double-click) on the Returns: field. This will bring you to the Expression Rules.
4. Enter an expression that will evaluate to the value you want to return. In this case, we used **Trim()** and **Dstr()** to format the date into a string.

Now, when the function is executed, it will return whatever you specified in the expression.

## How do I Create a Function That is Available for the Current Task Only?

The availability of a given function is determined by the *Scope* property. If the *Scope* is *Task*, then the function will only be visible to this task.

### Creating a local function



1. Go to your function
2. Click on the field after *Scope*: (or tab to it).
3. Choose *Task* from the selection list.

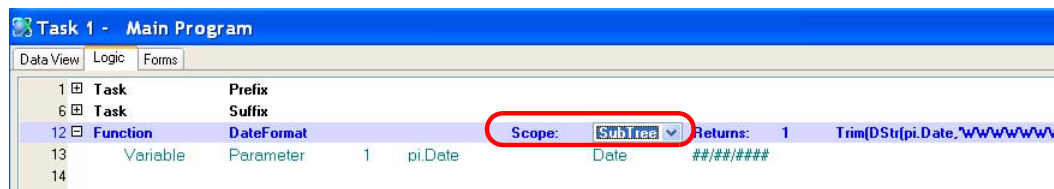
Now the function will only be visible within this task.

**See also:** Chapter 11, “How do I Create a Function?” on page 241 for details on the basics of creating a function.

# How do I Create a Function That is Available For the Entire Project?

The availability of a given function is determined by the *Scope* property. If the *Scope* is *Subtask*, then the function will be visible for this task and all it's children. This means that if the function is entered in the Main Program with a *Scope* of *Subtask*, then the function will be visible within the entire project.

## Creating a global function



- 1. Create your function in the *Main Program*.
- 2. Click on the field after *Scope*: (or tab to it).
- 3. Choose *Subtask* from the selection list.

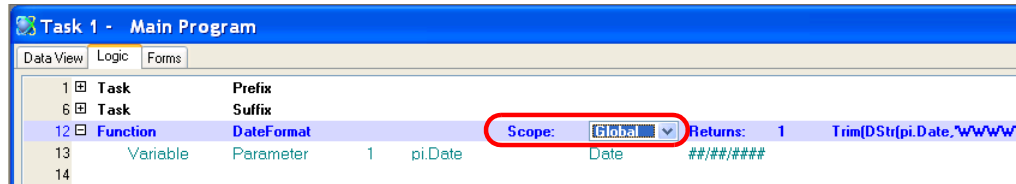
Now the function will be available for the entire project.

**See also:** Chapter 11, “How do I Create a Function?” on page 241 for details on the basics of creating a function.

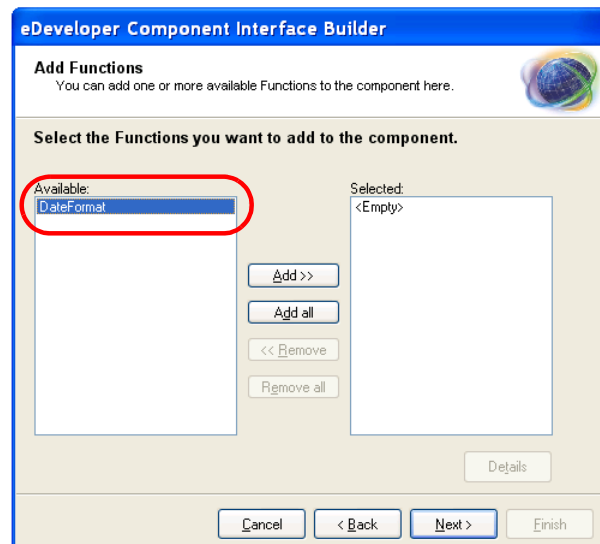
## How do I Share a Function Between Several Projects?

The availability of a given function is determined by the *Scope* property. If the *Scope* is *Subtask*, then the function will be visible for this task and all its children. This means that if the function is entered in the Main Program with a *Scope* of *Subtask*, then the function will be visible within the entire project.

### Creating a global function



1. Create your function in the *Main Program*.
2. Click on the field after *Scope*: (or tab to it).
3. Choose *Global* from the selection list.



Now the function will show up when you are generating a component, as shown above.

**See also:** Chapter 16, “How do I Reuse eDeveloper Objects Across Projects?” on page 411.

## How do I Iterate on a Series of Operations?

When you want to have a set of operations repeat themselves over and over, most programming languages have some kind of loop mechanism. In eDeveloper, this loop mechanism is the *Block While* operation. It can be entered in any logic unit.

As the name suggests, a *Block While* will execute while the condition that is controlling it is TRUE. In this example, we create a *Block While* operation to execute exactly 10 times.

### LoopCounter()

The *Block While* operation has it's own special function, called *LoopCounter()*. This function returns the number of times this loop has iterated, so you don't have to create a special counter for each loop.

### Creating a Block While operation



1. Go to the logic unit you want to use.
2. Press **F4** (**Edit->Create line**). A blank line will appear below your cursor position, and your cursor will be located on a field at the left side of the new line.
3. Type **B**. A Block operation will appear, with a *Block If* and *End Block* operations, and the cursor will be positioned on the "If".
4. Type W. Now you have a Block While.
5. Tab to the right, and **zoom** (**F5** or **double-click**). This will bring you into Expression Rules.
6. Enter the desired expression. You want an expression that will be FALSE when you want the loop to quit. In this example, we entered:

```
Loopcounter() <= 10
```

which will be FALSE on iteration number 11, so the loop will execute exactly 10 times.

## How do I Update a Variable?

There are several ways a variable can get updated in eDeveloper:

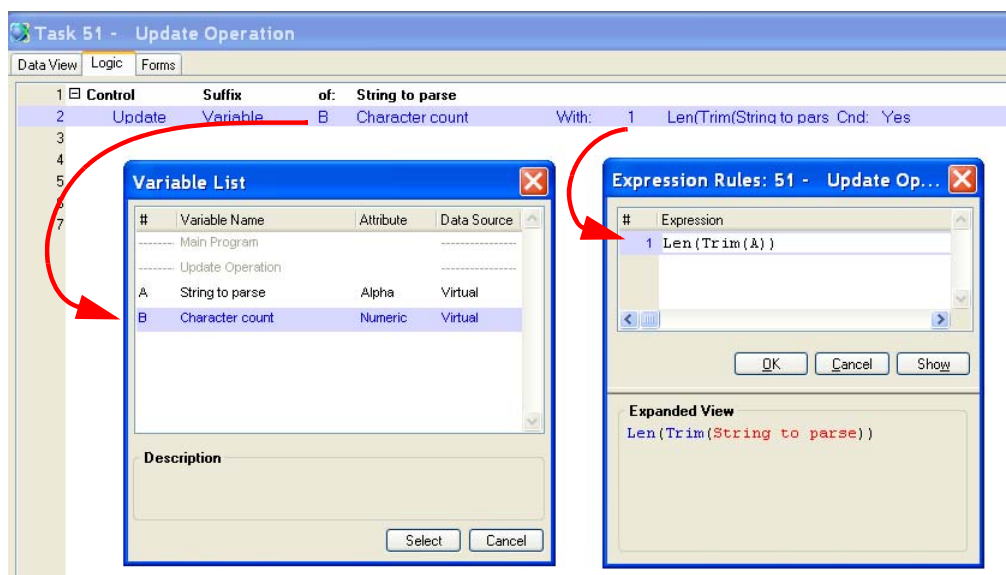
- A default value may have been assigned to the field at the *Model* or *Data Source* level, or in the current task.
- An *Init* may have been coded for that variable
- If the variable is on the form, the user may have typed in a new value
- An *Update* operation was executed
- One of the more advanced functions, such as the *VARSET* function, may have been used
- Data was read during an IO operation

Here we are going to cover the *Update operation*, which is the usual way to change a variable's value from within a logic unit. The Update operation is procedural: that is, you control exactly when it happens within a logic unit.

Much of the time, in online programs, the Init column is used instead of an Update. This is because an Init happens non-procedurally; the value is automatically updated whenever any of the values in the expression change. This saves you from having to figure out “when” you should do the update.

In this example, we are updating a “character count” field with the number of characters in a text field. The character count variable gets updated as soon as the user leaves the field, because we are coding it in the *Control Suffix* logic unit.

### Using the Update operation



1. Go to the logic unit you want to use.



2. Press **F4** (**Edit->Create line**). A blank line will appear below your cursor position, and your cursor will be located on a field at the left side of the new line.
3. Type **U**. The *Update Variable* operation will appear, and the cursor will move to the right.
4. Press **zoom** (**F5** or **double-click**) to bring up a list of variables. Select the variable you want to update, by moving the cursor to that line and pressing Enter or Select. You can also just type in the letter of the variable.

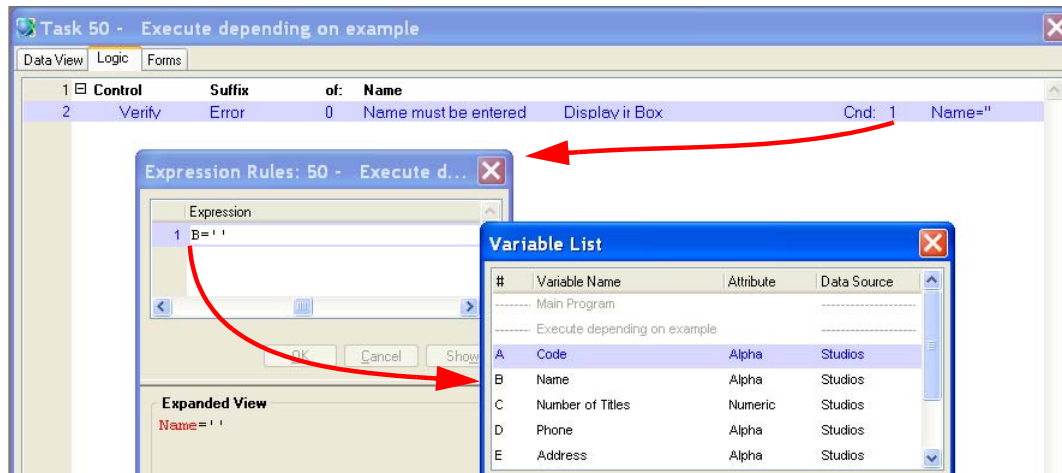
Press Tab to jump to the next field to the right, which is marked *With:*

5. **Zoom** (**F5** or **double-click**) from the *With:* column, which will bring you into Expression Rules.
6. Type in an expression you want to use to update the variable with. If you just want to update one variable so it matches another variable, then just enter the variable. Or, the expression could be a number, like zero, if you are resetting a counter. Or it could be an expression like we used here, which returns the number of characters in a string (minus whatever blanks might be at the end).
7. Press **Enter** or click the **OK** button to “bring back” the expression number into the *With:* field.

**See also:** Chapter 20, “How do I Set the Value of a Task Variable?” on page 521.

## How do I Condition the Execution of an Operation Based on a Variable's Value?

You have a lot of control over the execution of an operation in eDeveloper. Basically any boolean operation you can think of can be entered as an expression. In this example, we will enter an expression that makes an error message occur if a certain field is left blank when the user leaves the field.



### Entering an expression for an operation

1. Go to the operation you want to have a condition on. In this example, it's a *Verify Error* operation.
2. Tab to the *Cnd:* column at the far right of the operation.
3. **Zoom** (F5 or **double-click**). This will bring you to the *Expression Rules*.
4. Press F4 to open up a line. You will now be parked on a blank line.
5. **Zoom** (F5 or **double-click**) again, which will bring up a list of variables. You can select a variable by:
  - Moving to the variable and pressing Enter or clicking the Select button.
  - Typing in the letter that represents the variable (**B** in this example).
6. Type in the rest of the expression as needed.

The expression should evaluate to TRUE when you *want* the operation to execute. You can think of the *Cnd:* column as being an “IF”. If the condition is true, the operation executes. In this example, **B= ‘ ’** will be TRUE if **B** is blank, and then the user will get the Verify error message.

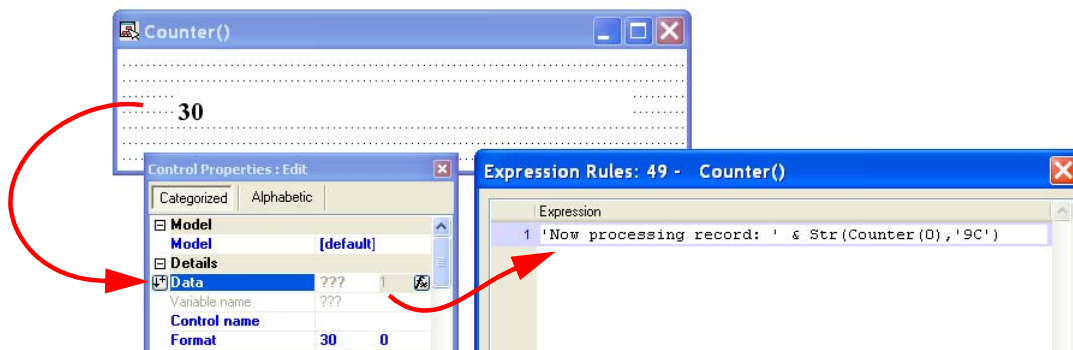
**See also:** Chapter 21, “Expressions” on page 537.

## How do I Retrieve the Sequential Number of a Record That is Handled by a Batch Program?

Often, in a batch program, you will want the number of the current record. You might use this, for example, to give the user an informational message, to create a sequential line number, or to cancel processing during debugging.

eDeveloper has a built-in function to handle this, called **Counter()**. It takes one parameter, which is the generation of the task. So Counter(0) returns the number of records processed by the current batch task, Counter(1) returns the records processed by the parent, etc.

### Using Counter()



1. Wherever you want to retrieve the current number of records processed, in the Expressions Rules, type

`Counter(0)`

For the current task. Use Counter(1) to refer to the number of records the parent has processed, Counter(2) for the grandparent, etc.

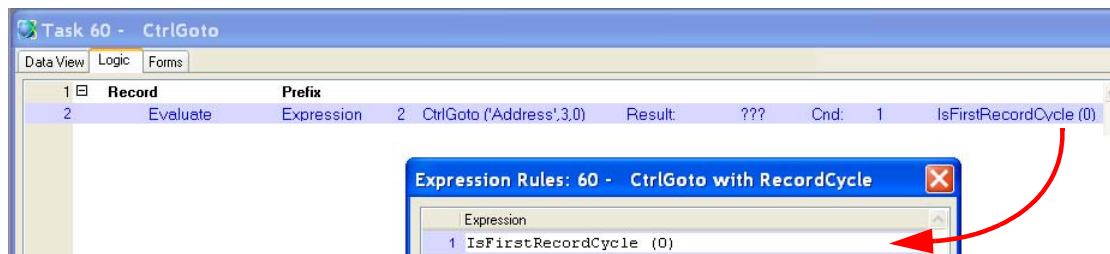
In this example, we use the value returned by Counter(0) to display a message to the user. Since the value returned is numeric, we use the Str() function to convert it to a string.

## How do I Condition the Logic to Be Executed Only for the First Record in an online Task?

For an online task, you will often want to execute logic when the user first enters the task. The proper place to do this is Record Prefix, because at that point the tables are open and the data initialized. However, if the user is viewing a tabular data, you might not want to execute the logic every time the user moves to a new line in the table.

For this instance, there is a special function, **IsFirstRecordCycle()**. This returns TRUE only the first time Record Prefix is executed.

### Using IsFirstRecordCycle()



1. Create an expression where you want to use *IsFirstRecordCycle()*.
2. Type:

```
IsFirstRecordCycle(0)
```

to test for the first record cycle in this task (IsFirstRecordCycle(1) will check the parent task).

In this example, we used IsFirstRecordCycle(0) so that a CtrlGoto would only execute the first time the task is opened.

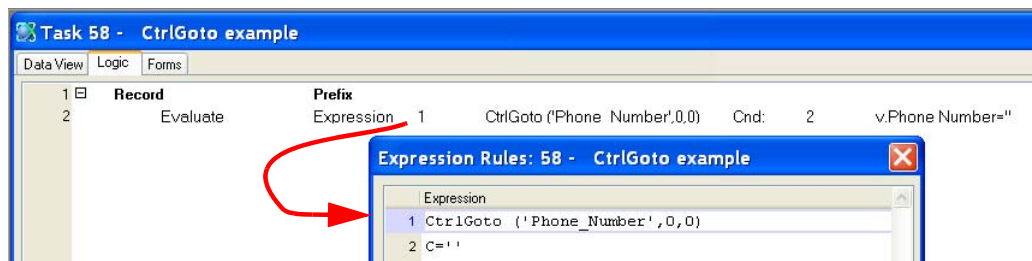
## How do I Make the cursor jump to a Specific Control?

Sometimes you may want to make the cursor jump to a particular control. For instance, when an error message is given, you may want to jump to the field in error, or you may want to skip fields that the user doesn't ordinarily use when the screen is first presented.

You can make the cursor jump to any control at runtime, by using the **CtrlGoto()** function. The syntax is as follows:

<b>CtrlGoto('Control name', row, generation)</b>	
<b>'Control name'</b>	The control name, as a string, in quotes.
<b>row</b>	In tabular data, this refers to the line number.
<b>generation</b>	Which task generation. The current task is task 0. Task 1 would switch focus to the parent task.

### Using CtrlGoto()



1. Go to the logic unit that will execute the cursor jump. In this case, we are using Record Prefix, because we want the cursor to jump to Phone Number when the user first enters the record.
2. Type **F4** to add an operation line.
3. Type **A** to select Evaluate Expression. The cursor will jump to the Expression field.
4. Press **F5** or double click to zoom to the Expression Rules.
5. Type **F4** to open up a line. Type

```
CtrlGoto(
```

or type **Ct** and press **Ctrl+Spacebar** to use the Auto complete feature.

6. From the right-click menu, select *Controls* to select the control you want, or type in the control name.
7. For most tasks, you will use 0,0 for the last two parameters, which means we don't have a table and we are jumping to a control in this task.

If, however, you are using a table, then the first number would be the line number to jump to. For instance, `CtrlGoto('Phone_Number', 3, 0)` would move the cursor to the phone number field on the 3rd line down in the table.

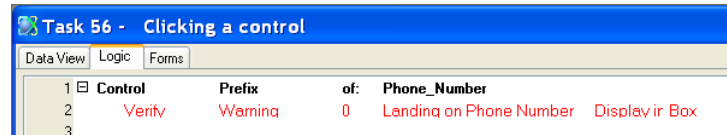
If you are jumping to a parent task, then you would use 1 for the second number, or 2 for the grandparent, etc. `CtrlGoto( 'Phone_Number' , 0 , 1 )` would move the cursor to the phone number field in the parent task.

## How do I Set Logic to Be Executed When the End-user Enters\Leaves a Control?

Much of your online programming logic will probably be executed when a user enters or leaves a field. For instance, when a user enters a field you may want to display some helpful text or automatically pop up a list of choices. When the user leaves a field, you may want to execute a function to do some computations, or validate what the user just entered.

This sort of logic is entered in the *Control Prefix* and *Control Suffix* logic units.

### Using Control Prefix



1. Press **Ctrl+H** to create your logic unit header line.
2. Type **C** to select the Control handler. You will jump to the next field.
3. Type **P** to select Prefix from the second drop down box. You will jump to the next field.
4. Zoom to select the desired control from a list of your current controls. If the control doesn't exist yet, or is out of scope, you can just type in the name.
5. Enter whatever operations you want to execute, in the newly created logic unit.

Now, the logic you entered will only be executed when the user lands on the specified control.

### Using Control Suffix



1. Press **Ctrl+H** to create your logic unit header line.
2. Type **C** to select the Control handler. You will jump to the next field.
3. Type **S** to select Suffix from the second drop down box. You will jump to the next field.
4. Zoom to select the desired control from a list of your current controls. If the control doesn't exist yet, or is out of scope, you can just type in the name.
5. Enter whatever operations you want to execute, in the newly created logic unit.

Now, the logic you entered will only be executed when the user leaves on the specified control.

**Note:** Control Suffix and Control Verification may seem similar, but they work quite differently. Control Suffix will always execute when the user leaves the control. But Control Verification will also execute when a user moves past the field, whether or not they landed on it. Use Control Verification when, for instance, you want to make sure a field was not left blank before a record is stored.

# How do I Condition an Operation to be Executed only When the End-user Tabs from one Field to Another in a Certain Direction?

Most of the time, the logic associated with a certain field should execute whether the cursor is moving up the screen or down it, as in a Windows environment users tend to click wherever they feel like. However, for more keyboard-centric applications, you can specify whether logic is executed according to the direction of the cursor flow. You do this using the **Flow()** function.

Flow(<string>) where <string> is:	Direction	Mode
N	Next	Step mode
F	Forward	Fast mode
P	Previous	Step Mode
R	Reverse	Fast Mode
S	Select exit	
C	Cancel	

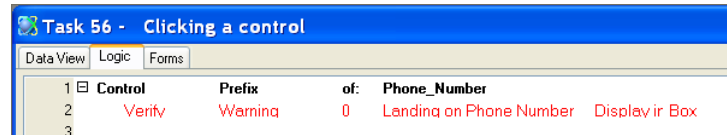
Flow( 'N' ) will return TRUE if the user is tabbing forward into a field. Flow( 'P' ) will return TRUE if the user is moving backward into it.



## How do I Condition an Operation to be Executed only if the End-user Sequentially Tabs from one Control to another, or When the End-user Skips to a Specific Control?

Sometimes you may have logic that needs to be executed as soon as a user lands on a specific control, either by tabbing to it or clicking on it. This is easily done by putting that logic in the Control Prefix logic unit of that control.

### Using Control Prefix



1. Press Ctrl+H to create your logic unit header line.
2. Type C to select the Control handler. You will jump to the next field.
3. Type P to select Prefix from the second drop down box. You will jump to the next field.
4. Zoom to select the desired control from a list of your current controls. If the control doesn't exist yet, or is out of scope, you can just type in the name.
5. Enter whatever operations you want to execute, in the newly created logic unit.

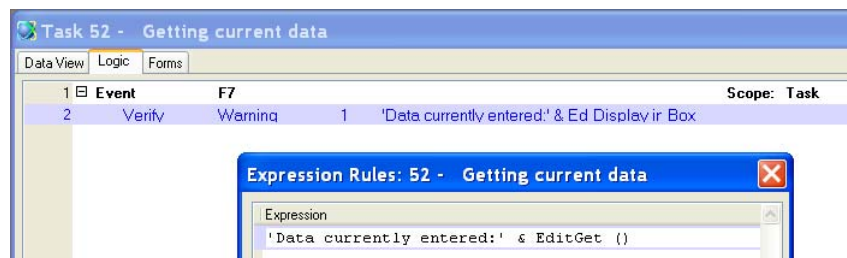
Now, the logic you entered will only be executed when the user first lands on the specified control.

## How do I Retrieve the Newly Entered Data of an Edit Control, Rich Edit Control, and Multi Choice List Box While Remaining on the Control?

When the user is typing data in a field, the changed field is not actually stored in the variable, or visible to handlers, until the user leaves the field. This is usually exactly what you want, because you do your validation etc. in control suffix, after the user leaves the field.

However, there are cases where you want the currently entered data before the user leaves the field. This would be the case when, for instance, the user is sitting on a field and hits a hot key to do something related to that field, such as a calculation.

### Using EditGet()



1. Wherever you want to retrieve the current value of the field in an expression, type

```
EditGet()
```

(or type **ed**, then **Ctrl+Spacebar**, and use the auto complete popup).

**EditGet()** will be replaced by the data the user typed in when the event handler was invoked.

Note that **EditGet()** will work for any edit field, so the Expression Editor doesn't know what data type will be returned. In the example above, if the user was parked on a numeric field and pressed F7, we'd get invalid results.

For this reason, you will probably want to use this in conjunction with the Control Name property or HandledCtrl(). See Chapter 11, "How do I Identify from Which Control an Event Was Triggered?" on page 263, and Chapter 11, "How do I Define an Event Handler to Be Executed Only When the User is Parked on a Specific Control?" on page 264.

## How do I Identify from Which Control an Event Was Triggered?

An event handler can exist at a higher level than the task it is handling. A good example of this are event handlers in the Main Program, which can be universally used in any task in the project. If you want to query which control the user is parked on, you need to use the **HandledCtrl()** function.

### Using HandledCtrl()



1. Go to Expression Rules.

2. Type:

`HandledCtrl ( )`

or, type **Ha** and press **Ctrl+Spacebar**, and the Auto Complete feature will fill in the rest of the function name for you.

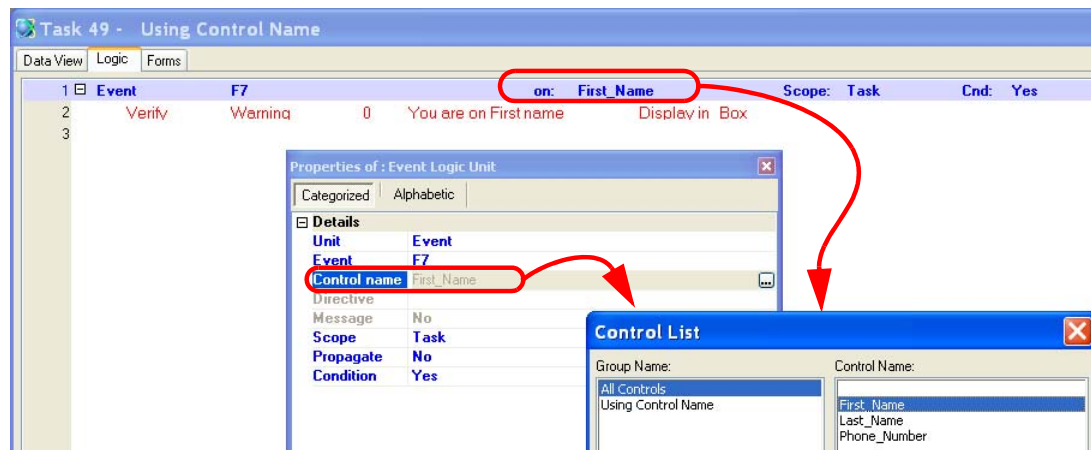
Now you can use the control name to make decisions within the event handler. For instance, you could have a generic help system that responds to characters in the control name to call specific help files.

**See also:** Chapter 11, “How do I Define an Event Handler to Be Executed Only When the User is Parked on a Specific Control?” on page 264.

## How do I Define an Event Handler to Be Executed Only When the User is Parked on a Specific Control?

It is easy to capture events using the Event logic unit. But what if you want an event to be triggered only when the user is located on a specific control? You can do that using the Control name property of the event.

### Using the Control name property of an event



1. From either the **on:** column of the event, or the *Control name* property, zoom to select the control name where the event will be active.

In this example, we selected the “First\_Name” control for the system event **F7**. That means, when the user presses the **F7** key, the handler will only execute if the user is parked on the control named “First\_Name”.

**Hint:** If you need to have one event handler handle a variety of controls, use the *HandledCtrl()* function within the event handler, as discussed in Chapter 11, “How do I Identify from Which Control an Event Was Triggered?” on page 263.

## Chapter 12: Date & Time

---

### How do I Retrieve the Current Date?

Getting the current date in eDeveloper is done using the **Date()** function. **Date()** returns the system date, as it exists on the machine eDeveloper is running on.

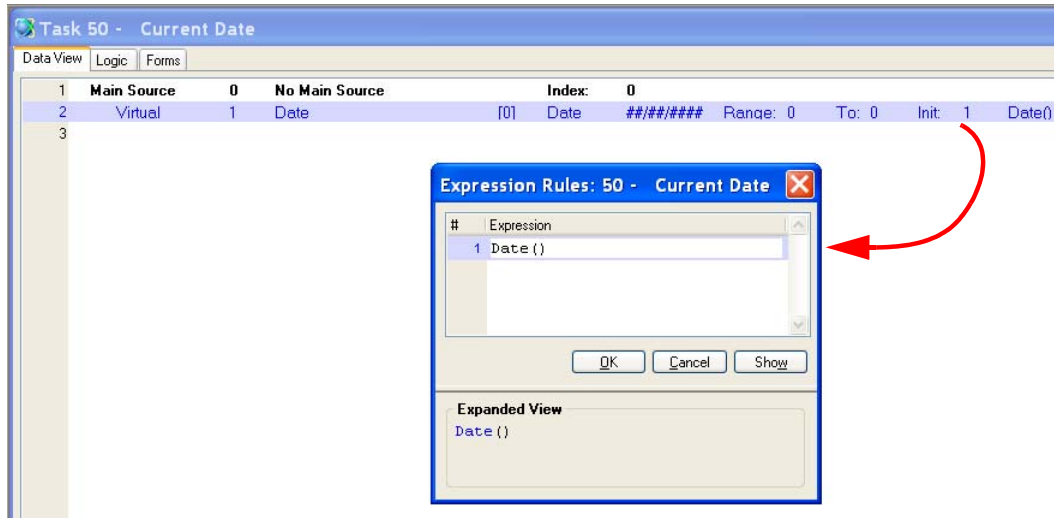
**Note:** You might notice a similar function, **MDate()**. This function does not return the system date, rather, it returns the login date that the user can set to some other date when they log in. This might be used in, say, an accounting system to allow the user to “pretend” it is a different date for bookkeeping purposes.

### Date Storage

Dates in eDeveloper are stored internally as an integer, that represents the number of days since the year zero. This isn't something you need to deal with, since all the conversions are done automatically. But it means that in terms of arithmetic, a date is considered just a number. So if you add 5 to a date, you get a date 5 days from the original. You don't need to worry about issues like “is the date at the end of the month” or “is it a leap year”.

When dates are stored in your data source, the storage is determined by the properties set up in the data source definition. If you are sharing the data with any other tools, such as report writers, it is important that the definition of the date field be something the other tool can handle.

## Using Date()



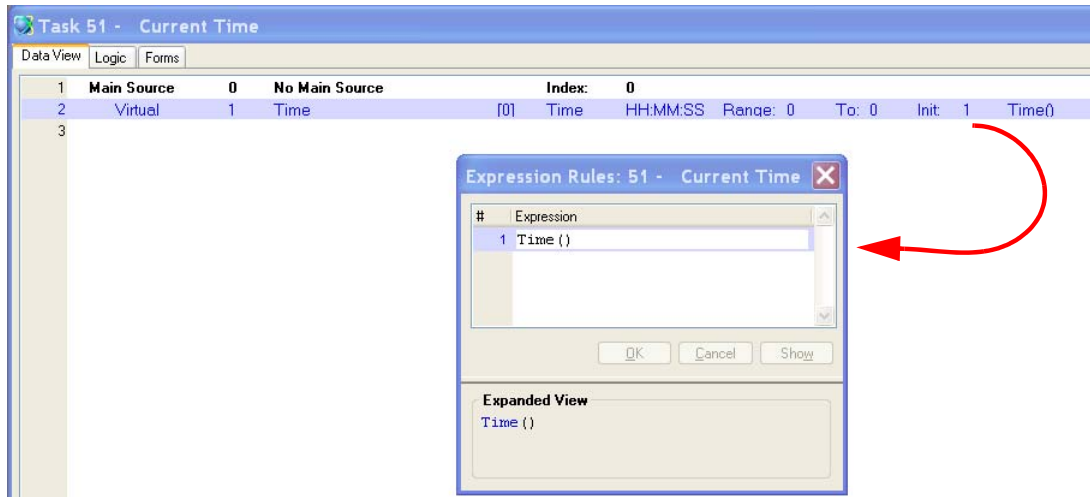
The **Date()** function is often used in an *Init* column, as shown here. This will set the field to the current date the first time the user sees the screen, and is also useful for setting timestamps for when a record was created or modified.

To use it, you just type **Date()**. At runtime, the function will be replaced by the current date.

## How do I Retrieve the Current Time?

Getting the current time in eDeveloper is done using the **Time()** function. **Time()** returns the system time, as it exists on the machine eDeveloper is running on.

### Using Time()



The **Time()** function is often used in an *Init* column, as shown here. This will set the field to the current time the first time the user sees the screen, and is also useful for setting timestamps for when a record was created or modified.

To use it, you just type **Time()**. At runtime, the function will be replaced by the current time.

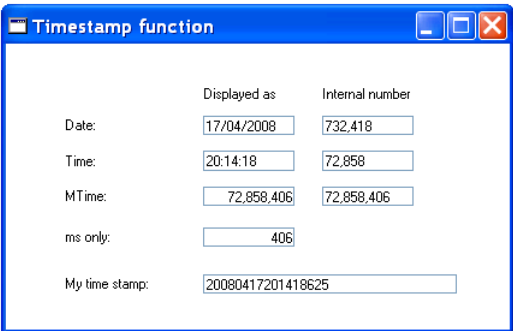
# How do I Retrieve the Current Time Using Milliseconds Precision?

The **Time()** function returns the number of seconds since midnight. It is stored, internally, as the number of seconds, though it is usually displayed in an HH:MM:SS format.

Sometimes though, you might want more precision, generally for creating a timestamp. The **mTime()** function does this. It returns the number of milliseconds since midnight.

There is no automated function to display the number of milliseconds. If you want to fetch just the milliseconds, you can use the expression:

```
mTime() - Time() * 1000
```



## Creating a timestamp using mTime()



You can create a numeric date/time stamp by using the expression shown here. This creates an 18-digit numeric code in the format:

```
YYYYMMDDHHMSSmmm
```

Alternatively, you can create the timestamp as an alpha string, but the numeric version stores in fewer bytes.



## How do I Increment a Date Value?

Dates in eDeveloper are stored internally as an integer representing the number of days since the year zero. So, to increment a date, you can just add some number of days to it. For instance:

```
Date() + 5
```

will return a date 5 days from now.

However, if you want to increment the date by some number of months or years, you would use the **AddDate()** function. It takes four parameters, as shown below:

<b>AddDate(Date, Years, Months, Days)</b>	
Date	The base date to add/subtract from
Years	Number of years to add or subtract
Months	Number of months to add or subtract
Days	Number of days to add or subtract

So, if you create an expression:

```
AddDate(B, 0, -1, 0)
```

Would subtract a month from the base date. So if you started with July 6, 2008, then the **AddDate()** function shown above would return June 6, 2008.

# How do I Increment a Time Value?

Dates in eDeveloper are stored internally as an integer representing the number of seconds since midnight. So, to increment a time, you can just add some number of seconds to it. For instance:

`Time() + 5`

will return a date 5 seconds from now.

However, if you want to increment the time by some number of hours or minutes, you would use the **AddTime()** function. It takes four parameters, as shown below:

AddTime(Time, Hours, Minutes, Seconds)	
Time	The base time to add/subtract from
Hours	Number of hours to add or subtract
Minutes	Number of minutes to add or subtract
Seconds	Number of seconds to add or subtract

So, if you create an expression:

`AddTime(B, -1, 0, 0)`

Would subtract an hour from the base time. So if you started with 14:08:03, then the **AddTime()** function shown above would return 13:08:03.

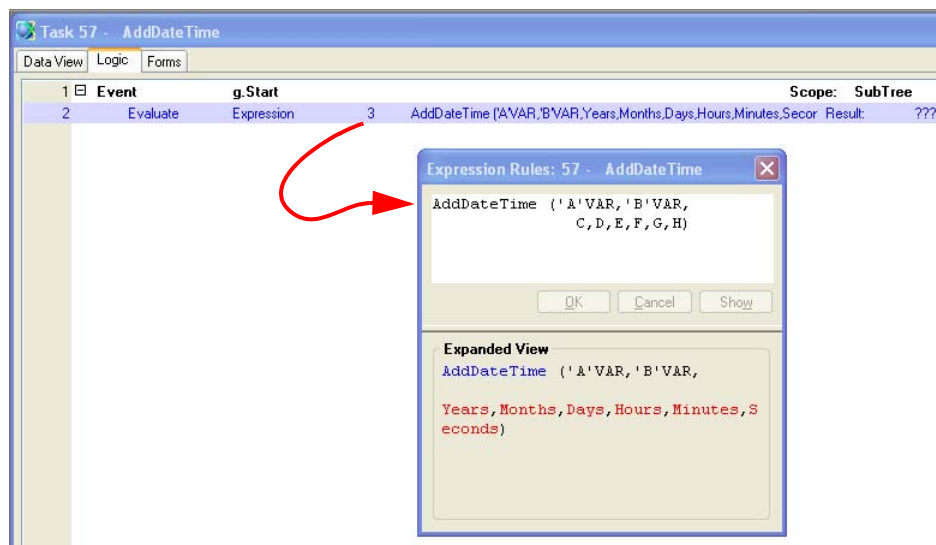
.

## How do I Increment the Value of Date-Time Combined Variables?

Adding an amount to a date-time variable can be a lot of work. For instance, suppose you are trying to determine the clock-out time for a worker who started work at 22:00:00. If you add 8 hours to the time, you also need to make sure to increment the date.

Fortunately, eDeveloper provides a function to treat the date and time as one unit. The **AddDateTime()** function allows you to add or subtract an amount from each of the parts of the date and time (years, months, days, hours, minutes, seconds) but automatically handles the rollover issues.

### Using AddDateTime()



**AddDateTime()** has the following syntax:

**AddDateTime**(DateVariable, TimeVariable, Years, Months, Days, Hours, Minutes, Seconds), where:

- DateVariable is a reference to the date you want to update.
- TimeVariable is a reference to the time you want to update.
- The other parameters are integers which will update the respective parts of the date and time. Use positive integers to add an amount, negative integers to subtract.

### About Variable References

You will notice in the example that the variables for the first two parameters are entered as *variable references*. That is, they are written in quotes, followed by the literal VAR: **'A'VAR** and **'B'VAR**. This is the format eDeveloper uses refer to a variable by address. This is necessary to update the variable from within an expression. Usually eDeveloper only updates variables with an *Update* operation, but in this case you are updating two variables at the same time so a reference is needed.

## How do I Calculate the Difference Between Two Datetime Values?

**DifDateTime()**

Date 1: 19/07/2011      Date 2: 19/04/2011     

Time 1: 13:31:54      Time 2: 14:48:39

**Difference**

Days: 90

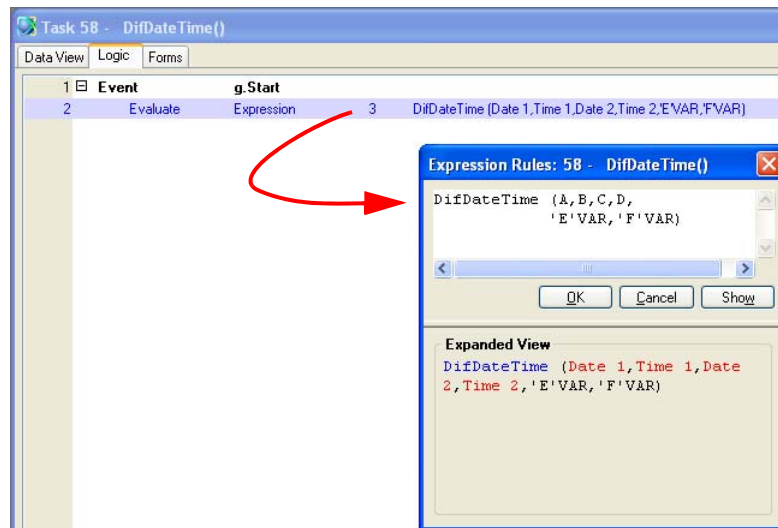
Seconds: 81795      Seconds in HH:MM:SS format: 22:43:15

Finding the differences between two date-time stamps can be a lot of work if you do it manually.

Fortunately, eDeveloper provides a function to treat the date and time as one unit for subtraction. The **Dif-DateTime()** function allows you to find the difference between two date-time pairs, as shown in this example.

**DifDateTime()** returns the number of days and number of seconds that are the difference between the two date-time pairs. You can convert the number of seconds into hours/minutes/seconds by treating it as a time variable and using the **Hour()**, **Minute()**, and **Second()** functions on it (See Chapter 12, “How do I Calculate the Hour\Minute\Seconds Portion of a Given Time Value?” on page 282).

### Using DifDateTime()



**DifDateTime()** has the following syntax:

**DifDateTime (Date1, Time1, Date2, Time2, DaysVariable, SecondsVariable)**, where:

- **Date1, Time1** are the first date-time pair

- *Date2, Time2* are the second date-time pair
- *DaysVariable* is the variable reference for the difference in days.
- *SecondsVariable* is the variable reference for the difference in seconds.

### About Variable References

You will notice in the example that the variables for the last two parameters are entered as *variable references*. That is, they are written in quotes, followed by the literal VAR: **'E'VAR** and **'F'VAR**. This is the format eDeveloper uses refer to a variable by address. This is necessary to update the variable from within an expression. Usually eDeveloper only updates variables with an *Update* operation, but in this case you are updating two variables at the same time so a reference is needed.

## How do I Calculate the Beginning of the Month of a Given Date?

It is very often the case that one needs to compute the first day of the month, especially for setting the date range for reports. There is a very quick way to do this, using the **BOM()** function.

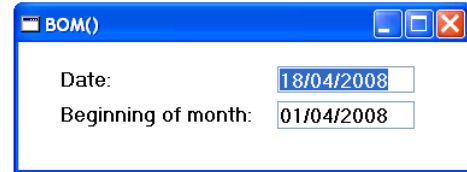
### Using the BOM() function

The syntax of **BOM()** is:

**BOM(*date*)**

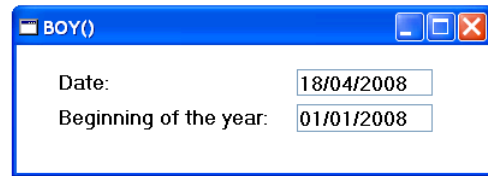
Where *date* is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns the first day of the month.

**See also:** Chapter 12, "How do I Calculate the End of the Month of a Given Date?" on page 277.



## How do I Calculate the Beginning of the Year of a Given Date?

It is very often the case that one needs to compute the first day of the year, especially for setting the date range for reports. There is a very quick way to do this, using the **BOY()** function.



### Using the BOY() function

The syntax of **BOY()** is:

**BOY(*date*)**

Where *date* is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns the first day of the year.

**See also:** Chapter 12, "How do I Calculate the End of the Year of a Given Date?" on page 278.



## How do I Calculate the End of the Month of a Given Date?

It is very often the case that one needs to compute the last day of the month, especially for setting the date range for reports. There is a very quick way to do this, using the **EOM()** function.

One of the nice things about **EOM()** is that it handles leap years correctly, as shown in the example.



The screenshot shows a window titled "EOM()". It contains two input fields. The first field is labeled "Date:" and contains the text "18/02/2008". The second field is labeled "End of month:" and contains the text "29/02/2008". The window has a blue title bar and standard window controls (minimize, maximize, close) on the right.

### Using the EOM() function

The syntax of **EOM()** is:

**EOM(date)**

Where *date* is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns the last day of the month.

**See also:** Chapter 12, "How do I Calculate the Beginning of the Year of a Given Date?" on page 276.

## How do I Calculate the End of the Year of a Given Date?

It is very often the case that one needs to compute the last day of the year, especially for setting the date range for reports. There is a very quick way to do this, using the **EOY()** function.



The screenshot shows a window titled "EOY()". Inside the window, there are two labels with corresponding text boxes. The first label is "Date:" and the text box contains "18/04/2006". The second label is "End of the year:" and the text box contains "31/12/2006".

### Using the EOY() function

The syntax of **EOY()** is:

**EOY(*date*)**

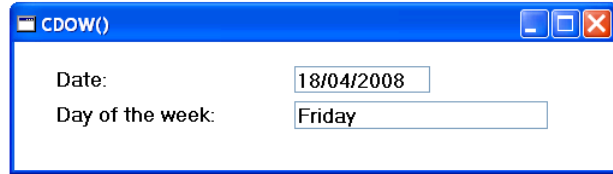
Where *date* is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns the first day of the year.

**See also:** Chapter 12, "How do I Calculate the Beginning of the Year of a Given Date?" on page 276.

## How do I Retrieve the Name of the Day of the Week of a Given Day?

It can be very difficult to figure out what day of the week a given date falls on, and more work to translate it into text. There is a nice function to do this though, the **CDOW()** function.

One of the advantages of using this function is that if your application is being distributed in other countries, it will return the date in the user's chosen language.



**Hint:** If you need to do computations based on the day of the week, it is better to use a different function, **DOW()**. This returns a number representing the day of the week, which is easier to use in programs.

**See also:** Chapter 12, "Date Pictures" on page 284.

### Using the CDOW() function

The syntax of **CDOW()** is:

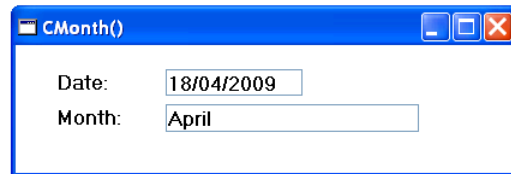
**CDOW(date)**

Where **date** is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns a string containing the name of the day of the week.

## How do I Retrieve the Name of the Month of a Given Day?

It can be very difficult to figure out what day of the week a given date falls on, and more work to translate it into text. There is a nice function to do this though, the **CMonth()** function.

One of the advantages of using this function is that if your application is being distributed in other countries, it will return the name of the month in the user's chosen language.



**Hint:** *If you need to do computations based on the month, it is better to use a different function, **MONTH()**. This returns the month number rather than the name, which is easier to use in programs.*

### Using the CMonth() function

The syntax of **CMonth()** is:

**CMonth(date)**

Where *date* is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date). It returns a string containing the name of the month.

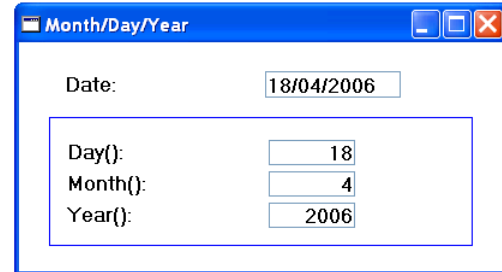
**See also:** Chapter 12, "How do I Calculate the Day\Month\Year Portion of a Given Date Value?" on page 281.

## How do I Calculate the Day\Month\Year Portion of a Given Date Value?

When you need to work with a date in a program, you usually want to deal with one “piece” of the date. For instance, you might want to summarize all the records for one month, or for one year.

There are three functions in eDeveloper that are used to break apart a date:

- **Day(date)** returns the day of the month
- **Month(date)** returns the numerical month
- **Year(date)** returns the numerical year



The screenshot shows a window titled "Month/Day/Year" with a blue title bar and standard window controls. Inside, there is a "Date:" label followed by a text field containing "18/04/2006". Below this is a bordered box containing three labels and text fields: "Day():", "Month():", and "Year()". The corresponding values in the fields are "18", "4", and "2006".

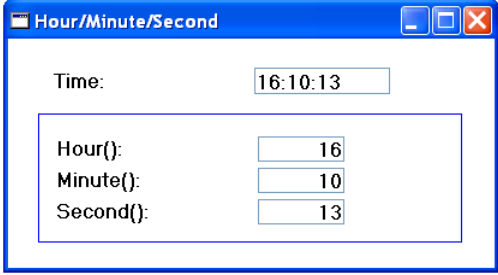
Where **date** is any date variable (or a hard-coded date, such as '05/06/2008'DATE, or an expression that evaluates to a date).

**Hint:** If you need to do computations based on the month, it is better to use a different function, **MONTH()**. This returns the month number rather than the name, which is easier to use in programs.

**Hint:** If you want to print the month or day of week, there are two other functions, **CDOW()** and **CMonth()**, which return the text version. See Chapter 12, “How do I Retrieve the Name of the Day of the Week of a Given Day?” on page 279.

## How do I Calculate the Hour\Minute\Seconds Portion of a Given Time Value?

When you need to work with the hour, minute, or seconds portion of a time, you can calculate the values by doing some arithmetic. However, there are some good functions to make the process a lot easier. Using the **Hour()**, **Minute()**, and **Second()** functions will easily return those parts of time field.



The screenshot shows a window titled "Hour/Minute/Second". Inside, there is a label "Time:" followed by a text box containing "16:10:13". Below this, there is a group box containing three rows: "Hour():", "Minute():", and "Second():", each followed by a text box containing the values "16", "10", and "13" respectively.

### The Time Functions

- **Hour(*time*)** returns the hour portion of the time, as a number.
- **Minute(*time*)** returns the minute portion of the time, as a number.
- **Second(*time*)** returns the seconds portion of the time, as a number.

Where *time* is any time variable (or a hard-coded time, such as '16:10:14'TIME, or an expression that evaluates to a time).

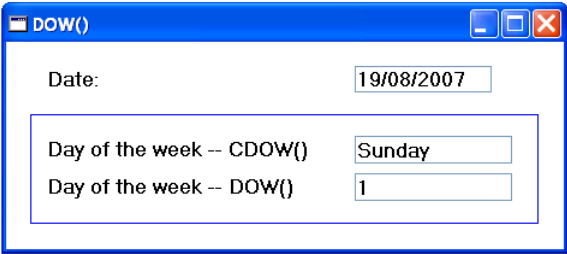
**Hint:** *If you need milliseconds, you need to use the MTime() function.*

**See also:** Chapter 12, "How do I Retrieve the Current Time Using Milliseconds Precision?" on page 268.

## How do I Calculate the Day of the Week of a Date, as a Number?

Some programming you do may be tied to the day of the week. For instance, if paychecks are delivered on Friday, you will want to know if a given date is a Friday.

It is safer to check for the day of the week though, if it is displayed as a number. This reduces spelling and capitalization issues, and also, your application might be deployed in a country that speaks another language.



A screenshot of a window titled "DOW()". Inside the window, there is a label "Date:" followed by a text box containing "19/08/2007". Below this, there are two rows of text. The first row is "Day of the week -- CDOW()" followed by a text box containing "Sunday". The second row is "Day of the week -- DOW()" followed by a text box containing "1".

So, to check the day of the week of a given date, use the DOW() function.

### Using DOW()

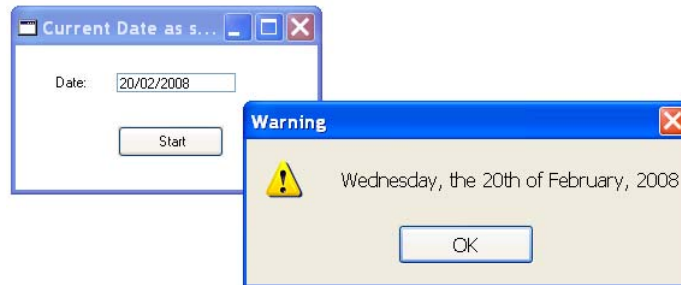
**DOW(date)** returns the day of the week, as a number between 1 (Sunday) and 7 (Saturday).

So in this example, since 19/08/2007 is a Sunday, DOW() returns the number 1.

**Hint:** If you need the day of the week as a string, use the **CDOW()** function.

**See also:** Chapter 12, “How do I Retrieve the Name of the Day of the Week of a Given Day?” on page 279.

## How do I Convert a Date Value to a String?



A date value in eDeveloper is basically numeric, and stored as the number of days since the year zero. When you display a date on a form, however, you normally do not need to convert it, because eDeveloper does that automatically, according to the picture property in the control.

However, when you want to include the date in a string -- for instance, in a *Verify Operation* box -- then you need to convert the date to a string. This is done using the **DStr()** function, as shown below.

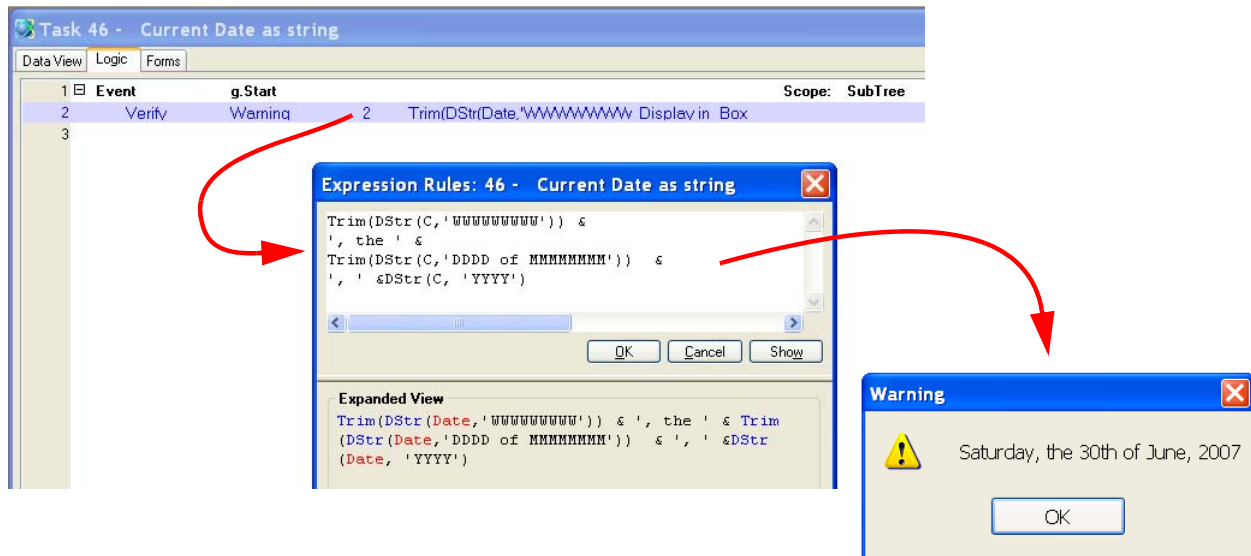
### Date Pictures

Whenever you display a date, you have a lot of formatting options in eDeveloper. The *picture* of the date determines how it is displayed. The picture is basically a kind of template that interprets the date. Letter placeholders are used to indicate how the date should be interpreted. For example, for the date “June 30th, 2007”:

Picture	How it displays as a string	Notes
MM/DD/YY	06/30/07	
DD/MM/YYYY	30/06/2007	
YYYYMMDD	20070630	
YYYY-MM-DD	2007-06-30	
YYYY	2007	
YY	07	
MMMM	June	
DDD	181	Julian date (nth day of the year)
DDDD	30th	
WWW	Sat	
WWWWWWWWW	Saturday	
##/##/####	Depends	Depends on how Settings->Environment-> International ->Date Mode is set.



## Using DStr()



The syntax of **DStr()** is:

**DStr**(date, picture)

Returns: A formatted date string. **DStr()** uses pictures to format the string.

**See also:** Chapter 12, "How do I Calculate a Date Value That Is Stored In a String?" on page 286.

# How do I Calculate a Date Value That Is Stored In a String?

In most cases, eDeveloper will handle date conversion issues for you. That is, when you are displaying a date on a form, reading a date with a form, or reading it from a data source, you don't need to manually do the conversion. There are a few scenarios where you may have to parse a string to extract the date, however.

In those cases, you need to use the **DVal()** function. This function uses pictures to interpret the string and find its date equivalent. It is up to you to be sure that the picture matches the date format in the string. For instance:

Data in a string	DVal() result
DVal('06/30/07' DATE, 'MM/DD/YY')	June 30, 2007
DVal('06/30/07' DATE, 'DD/MM/YY')	Invalid result

**See also:** Chapter 12, "How do I Convert a Date Value to a String?" on page 284.

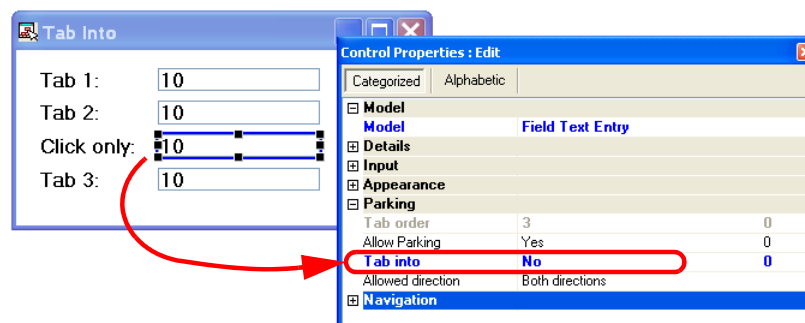
# Chapter 13: Handling GUI

## How do I Enable the End-User to Park on a Control Only by Mouse?

By default, fields you place on an online form will allow the user to tab from one to the next. However, if you would like to make a field only accessible by mouse, you can do that by setting the control's **Tab Into** property to No.

You can also allow tabbing into the field during certain conditions, by entering an expression that evaluates to TRUE when you want to allow tabbing. This is useful when you want to have an edit field the user will be able to park on to copy the text for example, but you would not want this field to be part of the tabbing cycle.

### Setting Tab Into



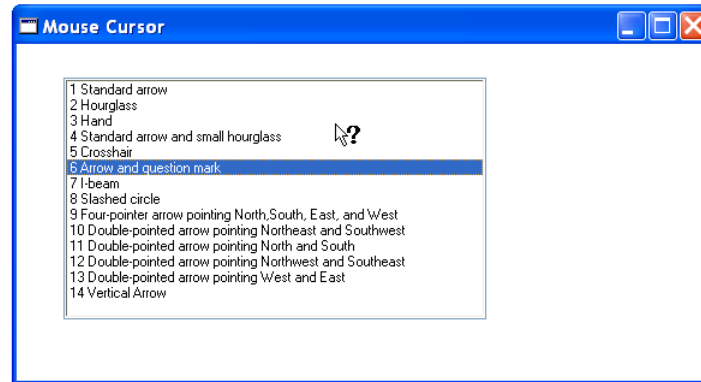
1. Select the control you want to change.  
If you want, you can select a group of controls, using **Ctrl+Click** or by rubber-banding them, and change them all at the same time.
2. Go to the Control Properties (**Alt+Enter**, or just click on the pane if it is already open).

- 3.** Go to **Parking->Tab into**. Type **N** to set the property to No.

Alternatively, you can click on the field to the right of the Yes/No field, and zoom. This will bring you to the *Expression Rules*, where you can enter an expression that will evaluate to TRUE when you want to allow the user to tab into the field.

Now, the user will not be able to tab into the field, but can still click on it.

## How do I Change the Mouse Cursor Icon?



In most situations, eDeveloper will set the mouse cursor as you would expect in a Windows application. However, you can change the default behavior when you have a special need to do so.

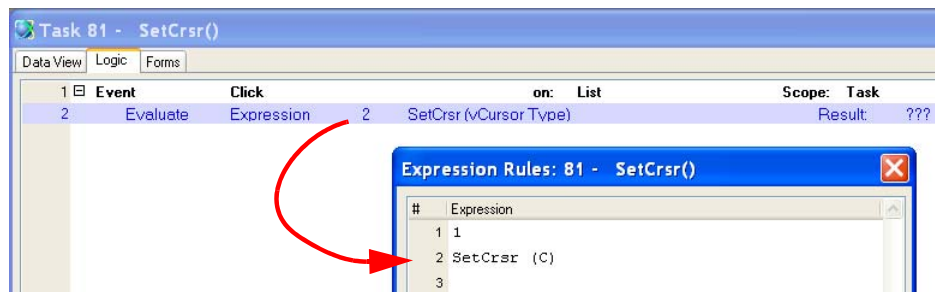
eDeveloper allows you to choose from a set of 14 different icons. These icons are standard within the Windows environment. Individual users may alter their own Windows setup, so that, for instance, the standard arrow is bigger or uses a different symbol.

To change which icon is currently in use, you use the **SetCrsr()** function, which has the syntax:

**SetCrsr(number)**

Where **number** is a value from 1-14, as shown above.

### Using the SetCrsr() function



1. Go to the logic unit where you want to change the cursor shape. In this case, we are changing the cursor when the user clicks on the list of cursor types.
2. Create an Evaluate Expression operation.
3. Set the Expression to

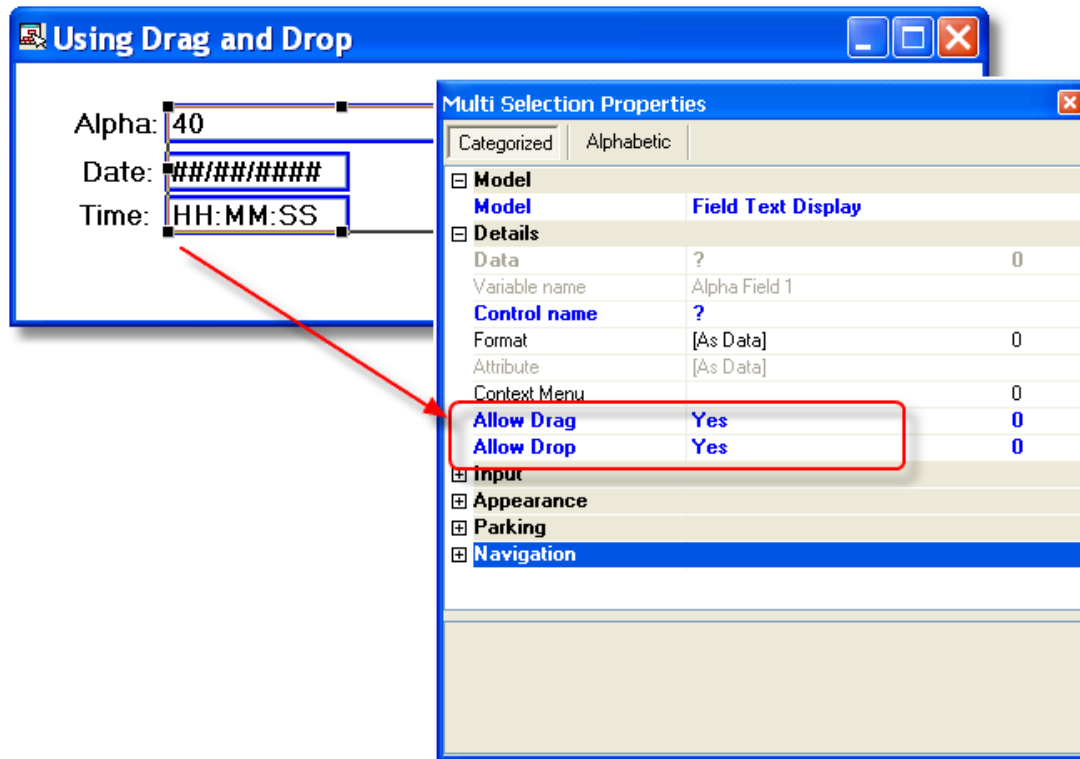
**SetCrsr (number)**

Where *number* represents the shape of the cursor you want. It can be a hard-coded number from 1-14, or, as in this case, held in a numeric variable.

Now, whenever the logic unit is executed, the cursor will change shape.

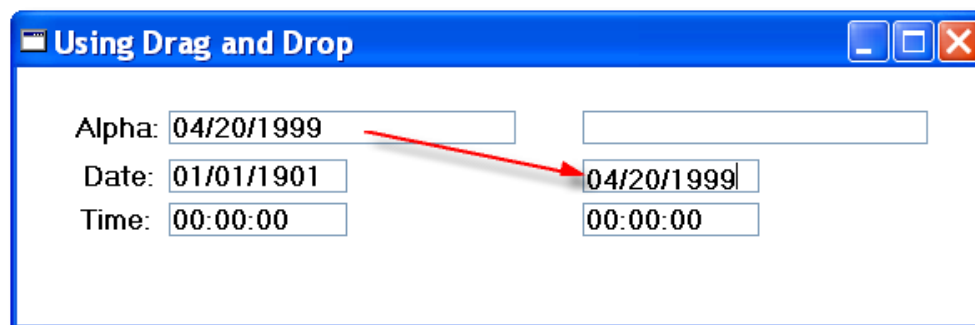
**Note:** Be sure to turn the mouse icon back to its default when you are done with the specific procedure that called for a different icon.

## How do I Copy Data Within an Application by Dragging It?



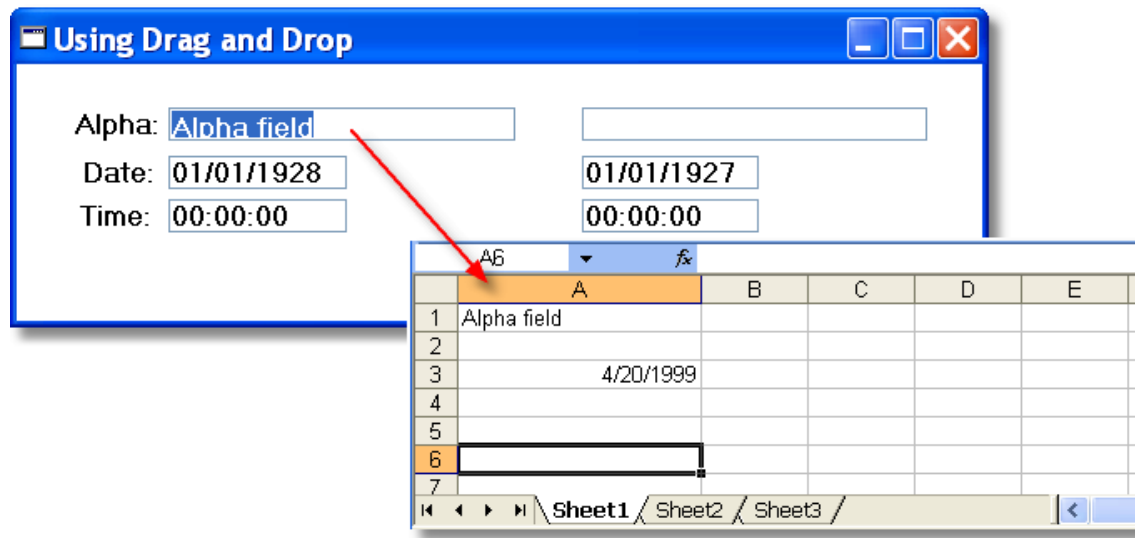
To allow drag and drop in an application, set the **Allow Drag** and **Allow Drop** properties to Yes.

If that is all you do, you can drag and drop data between the fields and eDeveloper will do the conversion if possible. For instance, in this example, we dragged a date from an Alpha field into a Date field.



To do more advanced drags and drops, you may need to use some of the drag and drop functions and handlers in eDeveloper. The *Drag Begin* and *Drop* event handlers allow you to capture when a Drag or Drop is taking place. The *DragSetData* and *DropFormat* functions give you more control over the format.

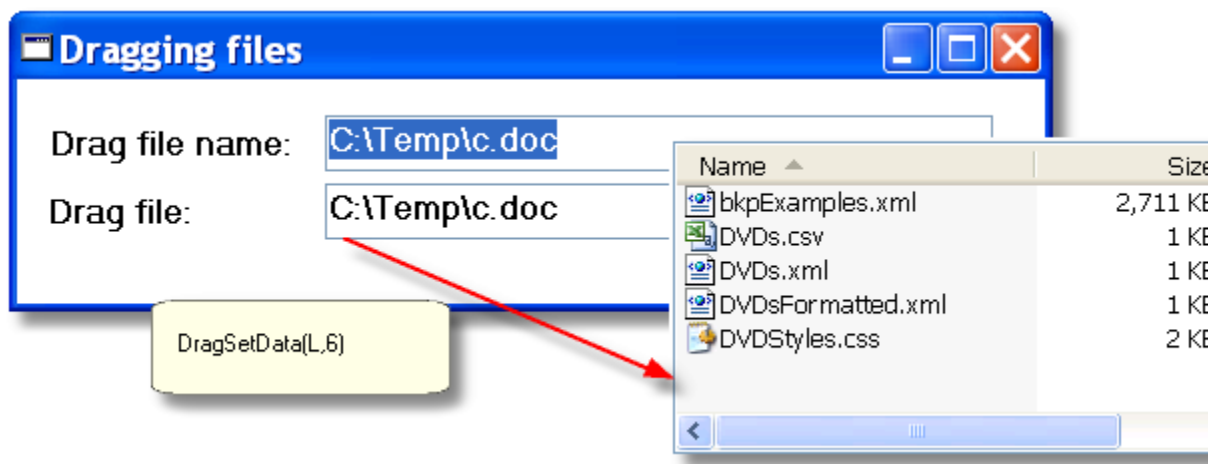
## How do I Drag Data From eDeveloper to External Applications?



In order to allow drag and drop into an external application, you need to have the **Allow Drag** property set to **Yes** in the control you want to drag. eDeveloper will handle the rest.

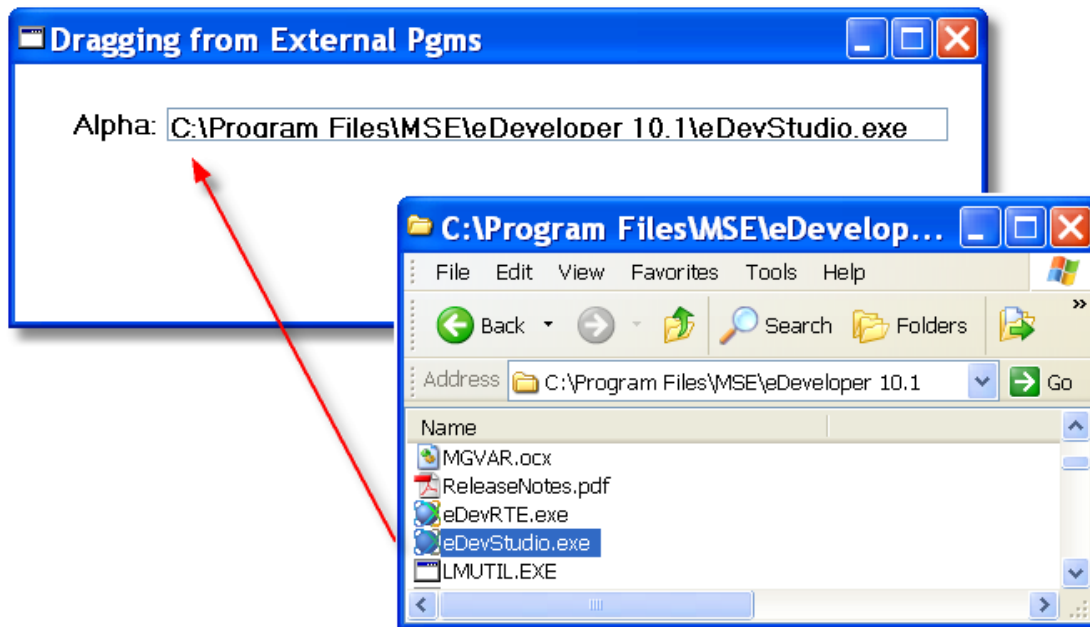
To do more advanced drags and drops, you may need to use some of the drag and drop functions and handlers in eDeveloper. The **Drag Begin** and **Drop** event handlers allow you to capture when a Drag or Drop is taking place. The **DragSetData** and **DropFormat** functions give you more control over the format.

For example, if you want to drag a file into a browser window, you cannot just drag the text name of the file; you need to signify to the browser program that what you are dragging is in fact an entire file. You do this by using the **DropSetData** function within a DragBegin event handler.

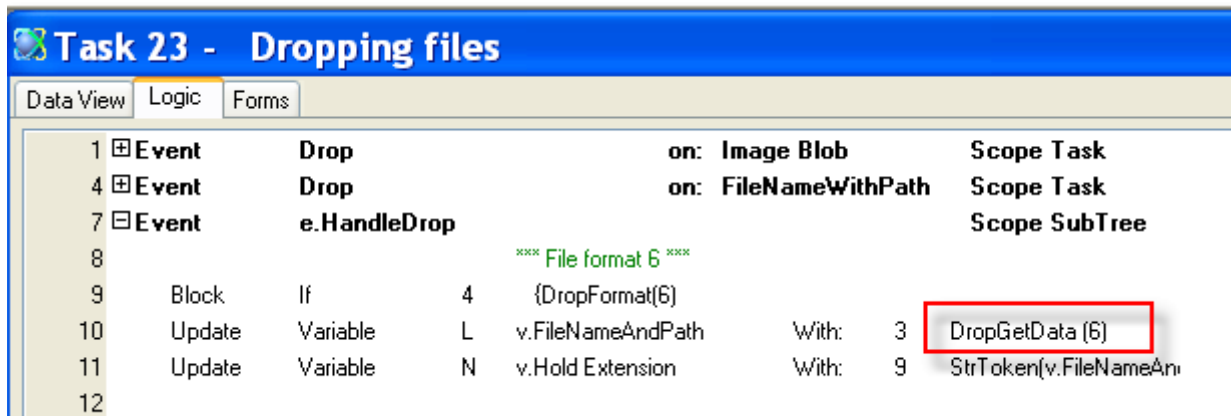




## How do I Retrieve a Full Path Name From a Dragged File?



When you drag a file into an alpha field from Windows, the full file name comes with it automatically. If you select two files in the file browser, you will get both file names separated by a comma.

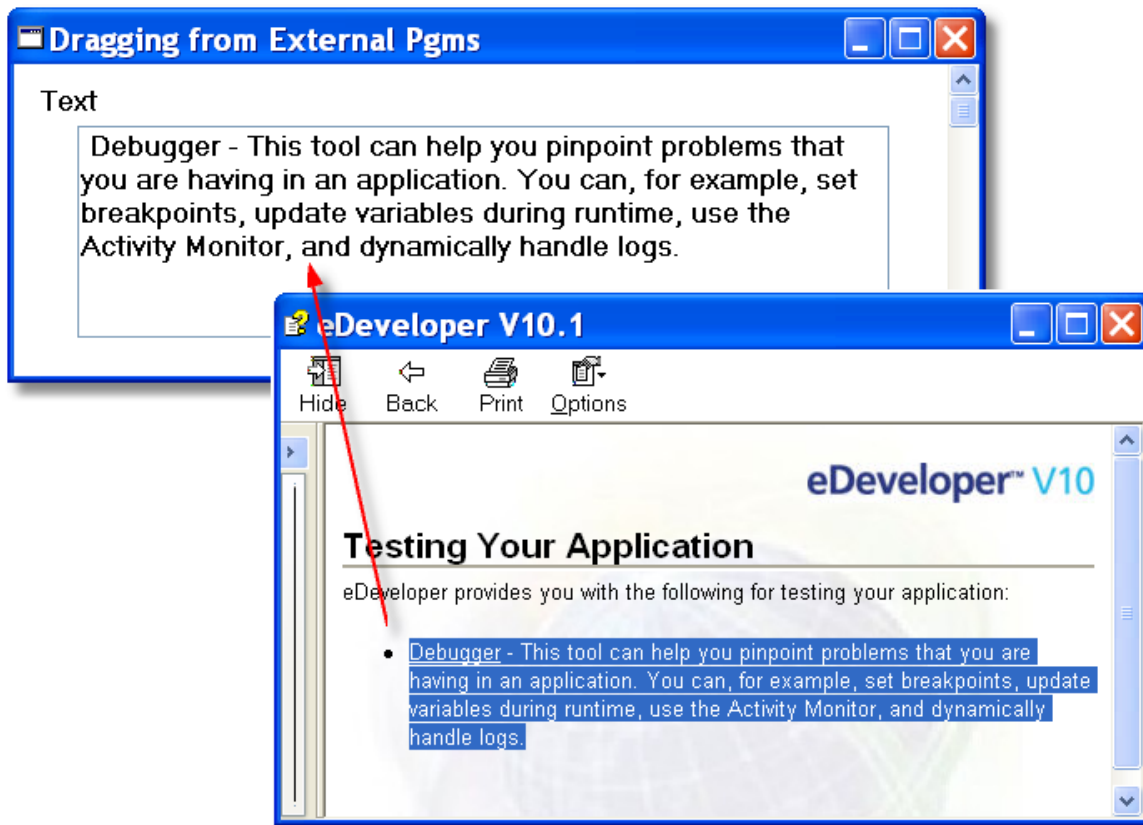


To fetch the file name manually, use the **DropGetData(6)** function.

In this code snippet:

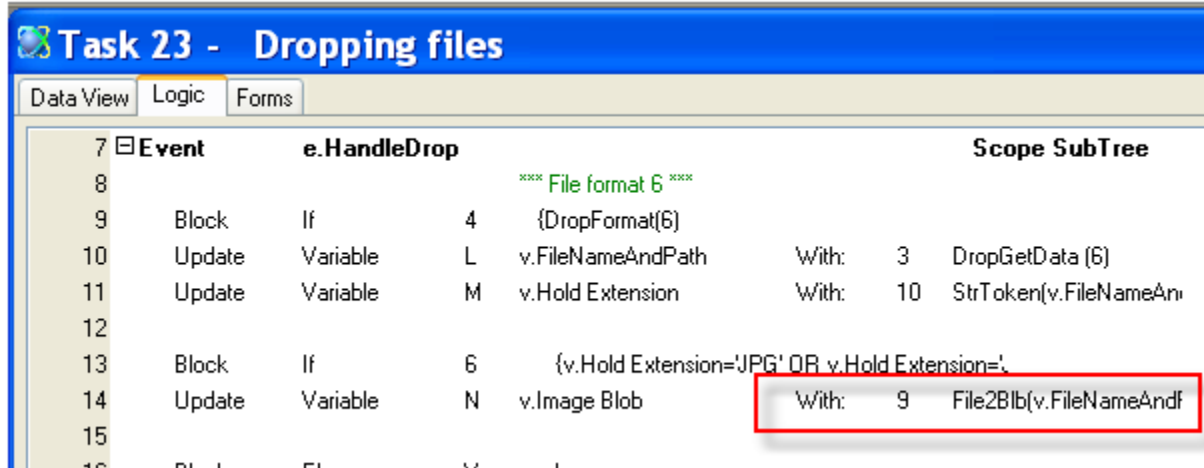
- We check that, in fact, a *file* is being dragged by using **DropFormat(6)**.
- We fetch the file name of the file by updating the file name with **DropGetData(6)**.

## How do I Drag Data From External Applications to eDeveloper?



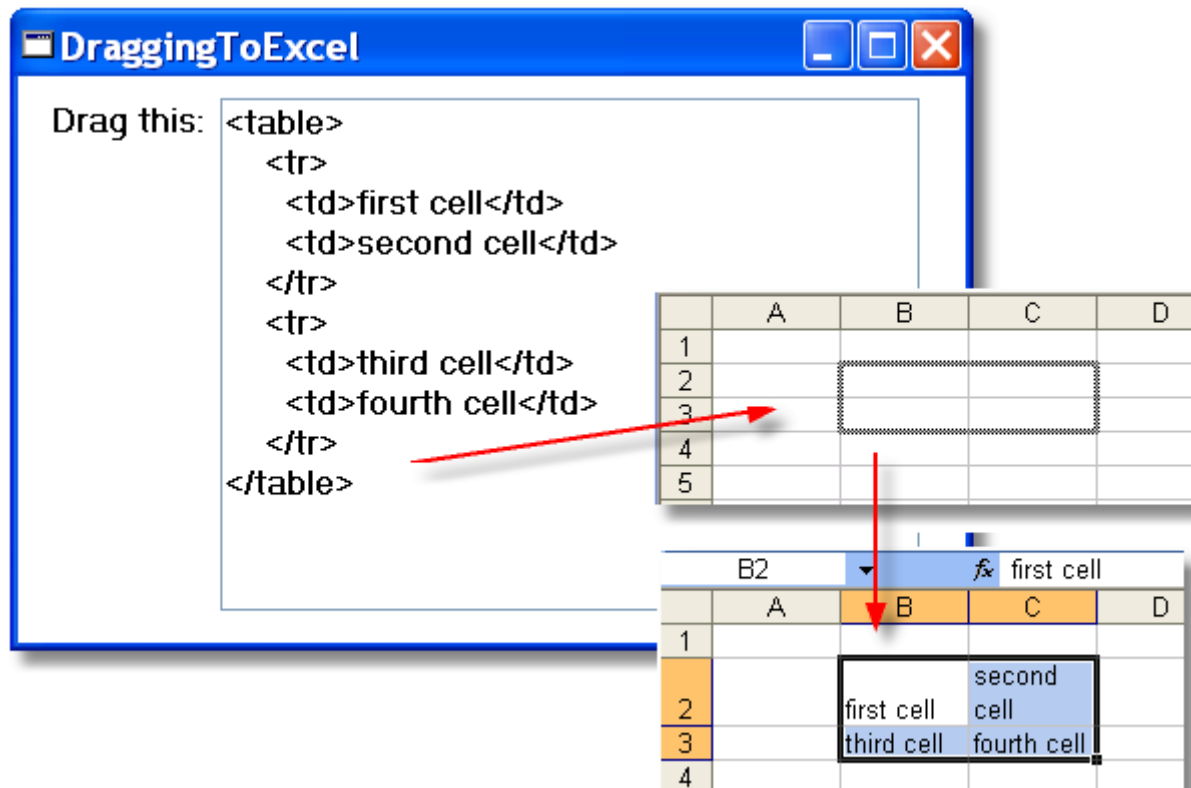
You can drag text from other applications into eDeveloper by selecting it and dragging it, if **Allow Drop** is set to **Yes** on the text control. How well this works depends on the internal operations of the application

you are dragging data from, because as with any drag and drop operation in Windows, the two fields have to match somewhat in format.



In some circumstances you will need to use the **DropGetData()** function to handle the external data format. For instance, suppose we are dragging a JPG from the file browser into an eDeveloper field. We can fetch the file name using **DropGetData(6)**, then check the file extension and discover it is a JPG file. But to display the JPG file, we need to convert the file into a BLOB, so we do a **File2Blob()**.

## How do I Drag Data in Table Format From eDeveloper to Excel?

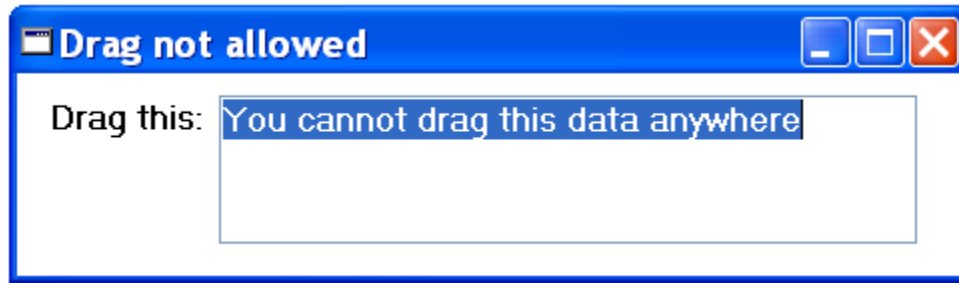


When you drag data from eDeveloper into Excel, Excel will try to figure out the format and use it accordingly. In this example, we formatted some table data using HTML format, and just dragged the text into Excel. Excel figured out that we meant to have a 4-cell table and dropped the data accordingly.



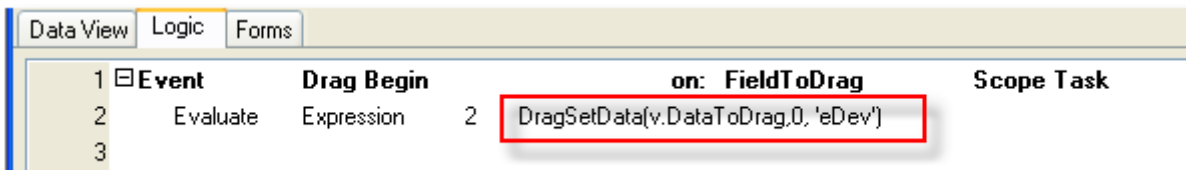
If you want to allow a user to drag an eDeveloper Data source table, you will need to format the eDeveloper table data into HTML or XML in an alpha text field. Then use the **DragSetData()** function to send the text data which will drag.

## How do I Allow Drag and Drop Only Within eDeveloper so That External Applications Cannot Retrieve the Data?



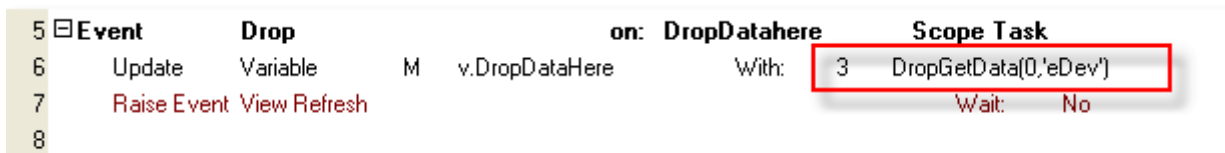
External applications will not read dropped data unless the data format matches a data format they know how to handle. Each application handles the various data formats differently, and if the application does not recognize a given data format, it will not allow the drop.

So, if you use format 0, the “User-defined format”, then no external application will recognize the data format and drop will not be allowed.

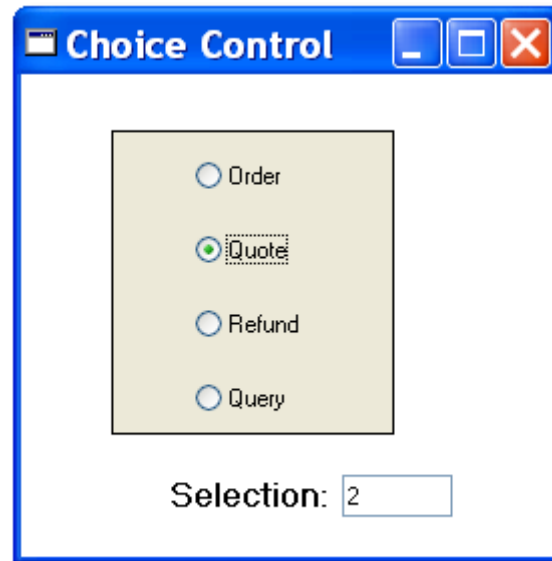


In this example, we used DragSetData() to set the format to zero. We also set Propagate to No, so the built-in drag function won't work.

Once we do this, the data will not drag to another field in eDeveloper either, unless we use the same user format within the Drop handler.

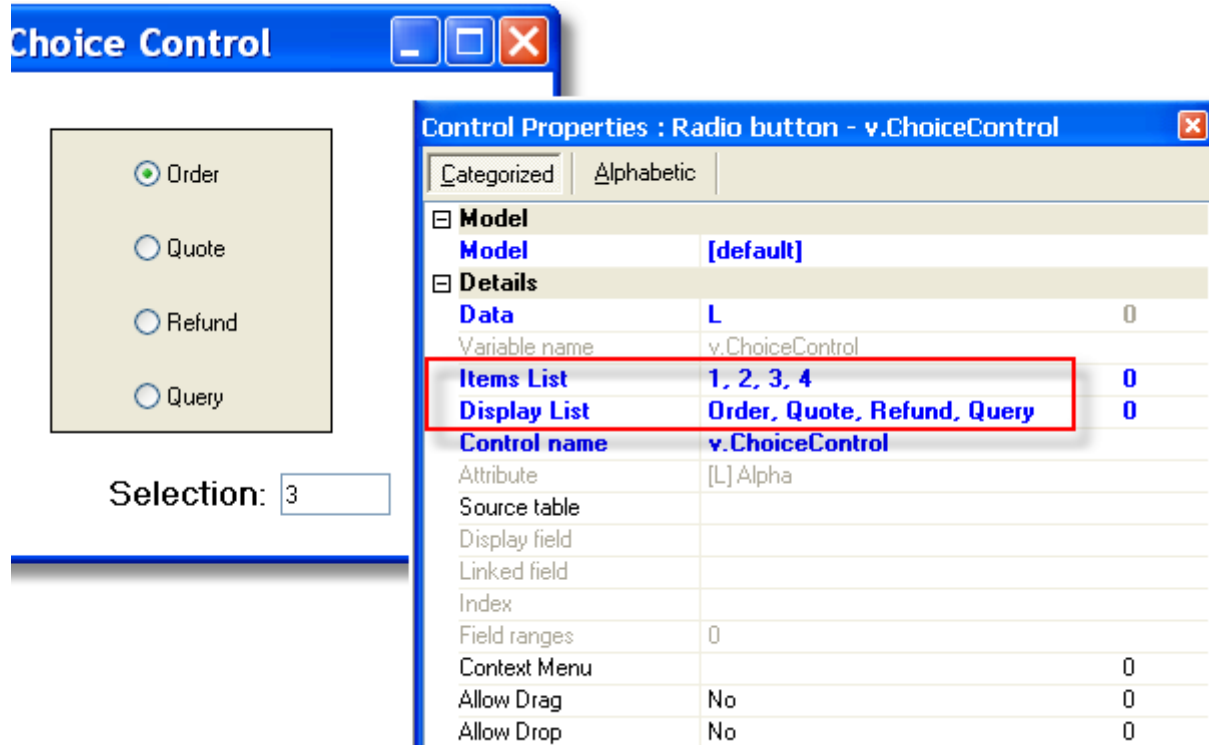


## How do I Display Different Text Than the Stored Values in a Choice Control?



When you are using choice controls, it is good to have the internal code be something different than what is displayed on the screen. This is especially important when you are creating applications that will be deployed in several different languages. In this example, for instance, we have four types of orders, Order,

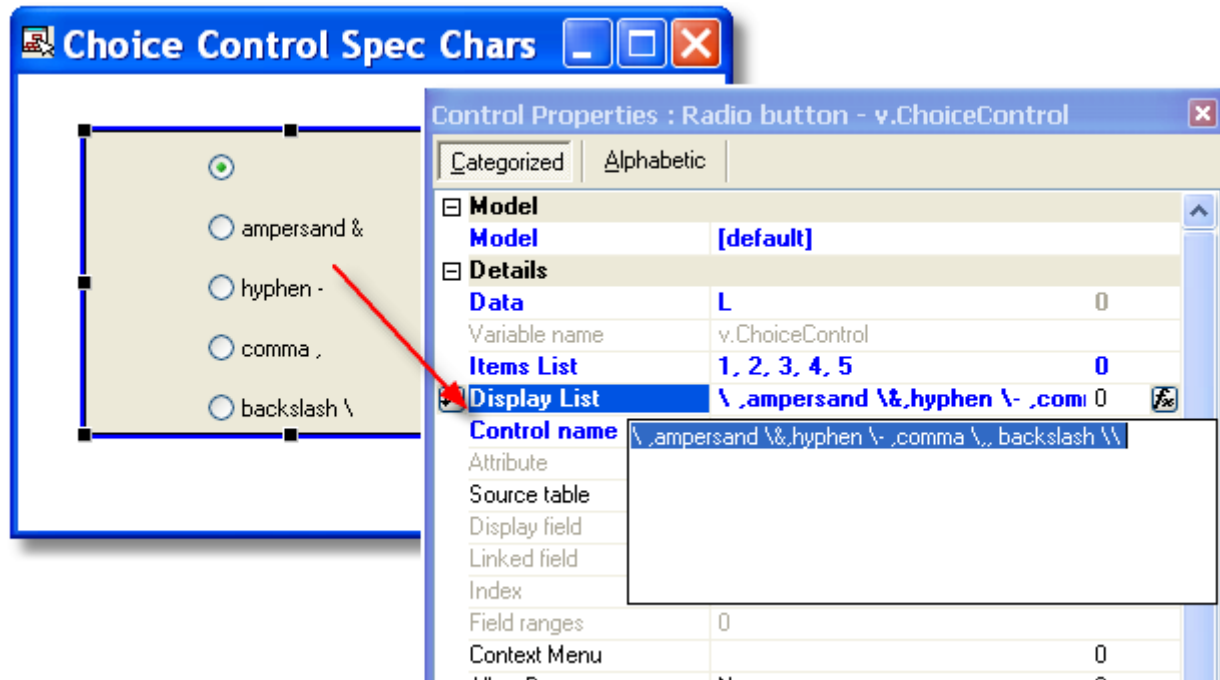
Quote, Refund, and Query. However, internally to the program, the records are stored with a type code of 1, 2, 3 or 4.



This is all encoded in the control itself, as shown here. The *Items list* property lists, in order, the values of the Choice control. The *Display list* property lists, in the same order, what will display onscreen.

While these are hard-coded in this example, you can also set them using expressions, or from a source table.

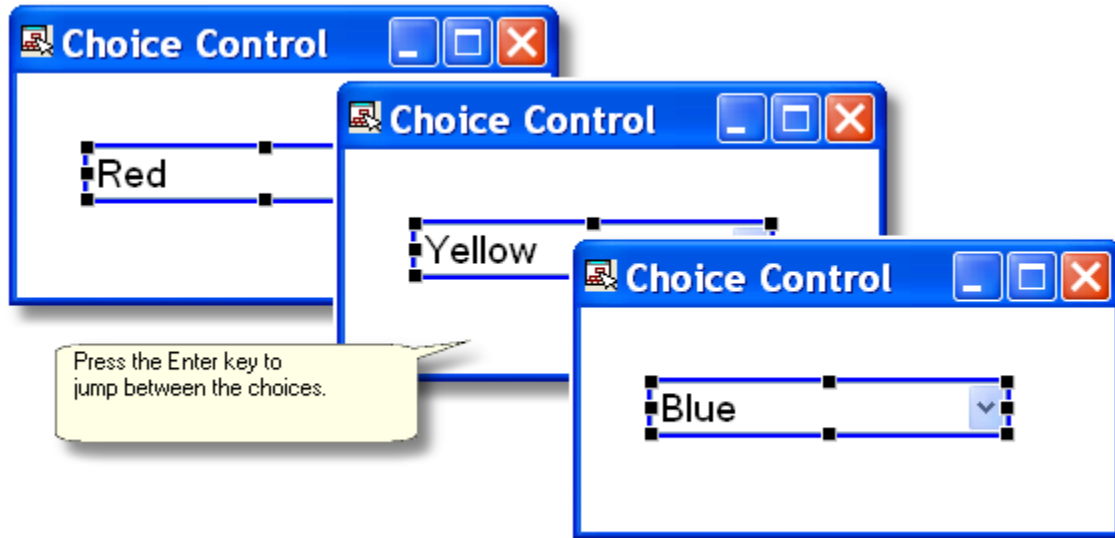
## How do I Display Special Characters in a Choice Control?



Some characters are used internally to choice controls. Commas, for example, are used to separate the items on the display list. You can still use these characters in your list: you just need to precede them with an escape character, the backslash. This applies to a blank, ampersand, and comma, as shown here.

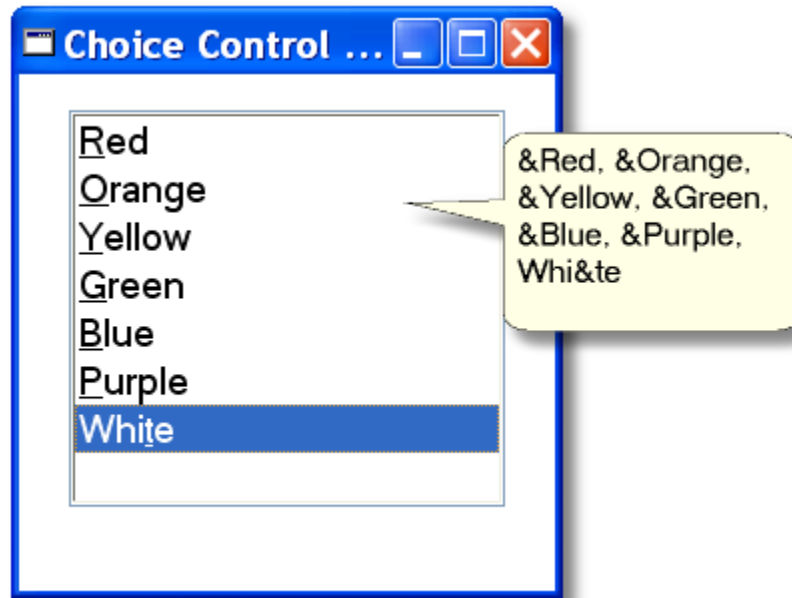


## How do I Switch Between Choice Control Options in the Form Editor?



While you are parked on a choice control in the Form editor, you can jump between the choices by simply pressing the **Enter** key. This is particularly useful when you are working with a Tab control, because as the Tab control option changes, different controls attached to the Tab become visible.

## How do I Set Accelerators to Choice Control Options?

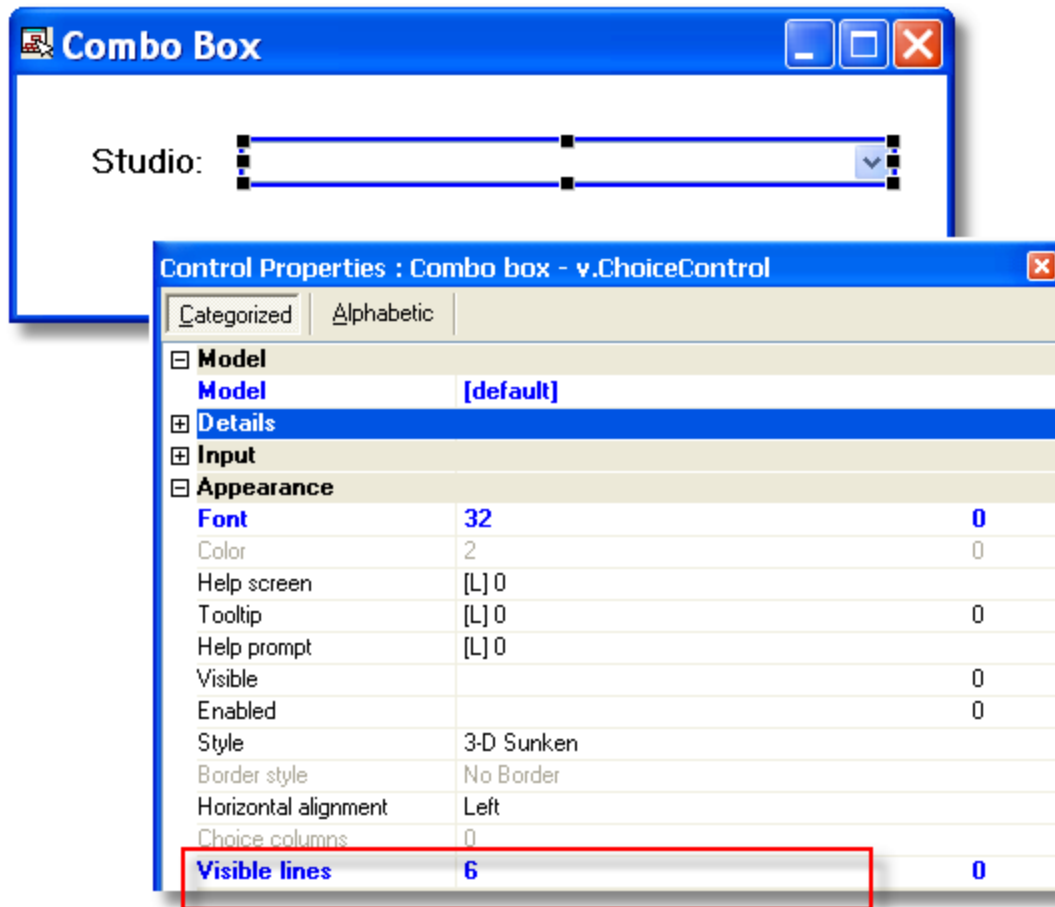


Each item on a choice control can have an accelerator key. When you set an accelerator key, pressing that key will automatically make the control jump to that choice. For instance, in this example, pressing “t” will select “White”.

You set the accelerator key by adding an ampersand in front of the value to be displayed.

**Note:** If you need to display an ampersand, precede it with a backslash.

## How do I Limit the Length of the Displayed Drop Down List of a Combo Box?

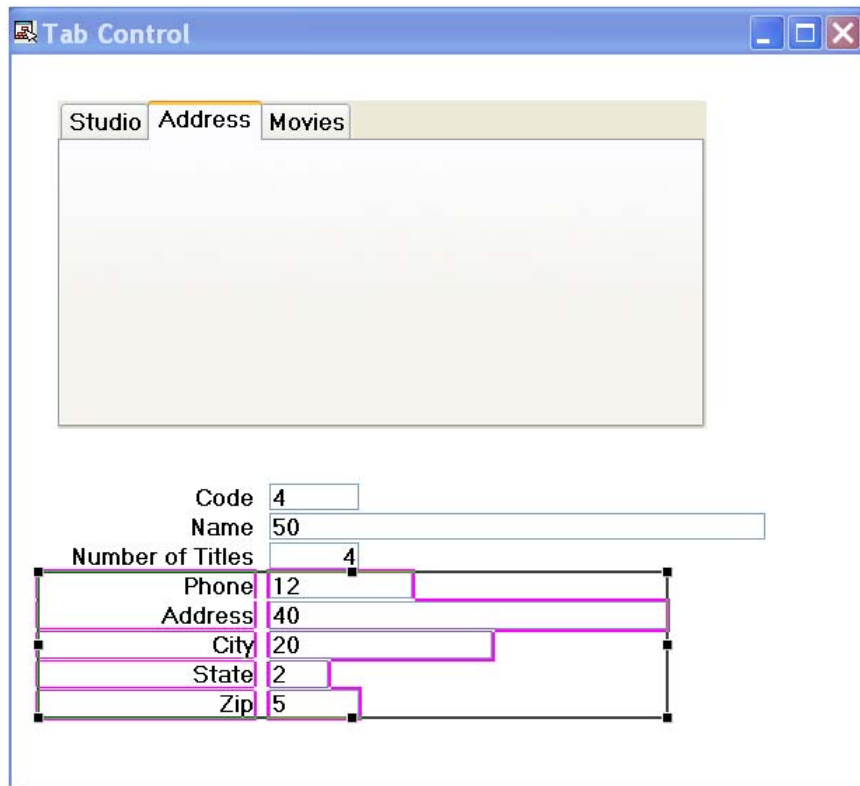



Sometimes there can be a lot of choices in a Combo Box. In order to limit how many lines show at one time, you can enter a value in the *Visible lines* property. In this example, a maximum of 6 lines will show to the user.

## How do I Associate Controls to Different Tabs in a Tab Control?

When you create a tab control, you can associate controls with one tab, so they only show up on the one tab, or you can associate the controls with all tabs, so they show up on every tab.

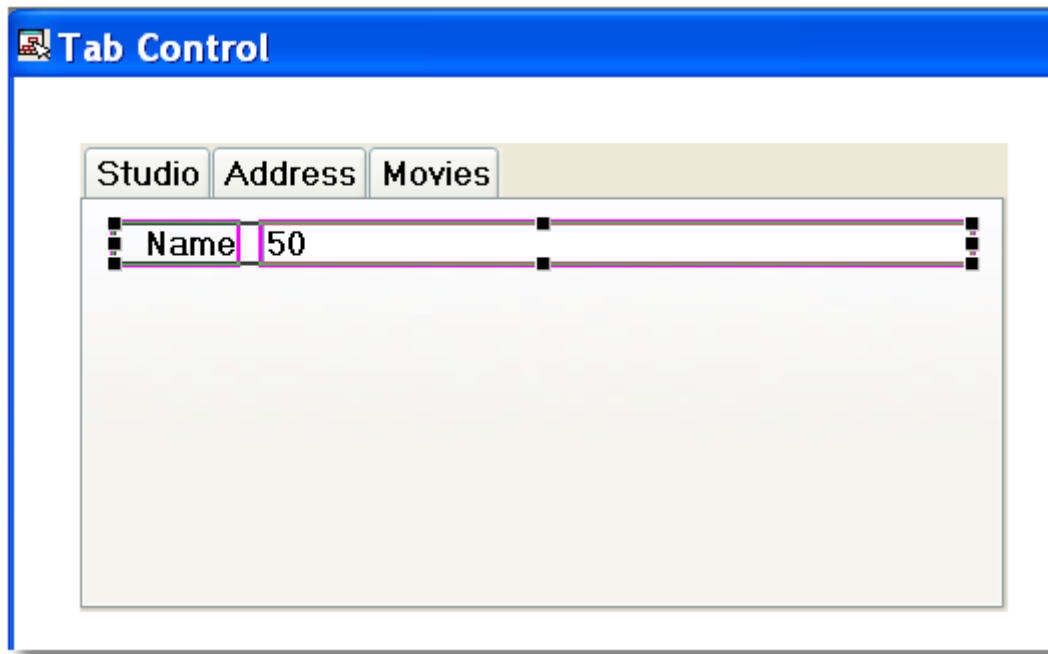
### Showing controls on one tab




1. Select the tab control.
2. Press **Enter** until you have selected the desired tab.
3. Select the items you want to connect to the tab.
4. Select the link  icon, then click on the tab. The items you have linked will show up in pink.
5. Move your selected items onto the tab folder.

Now, the items will only show up on the selected tab. In this example, the address items will only show up on the Address tab.

## Showing controls on all tabs

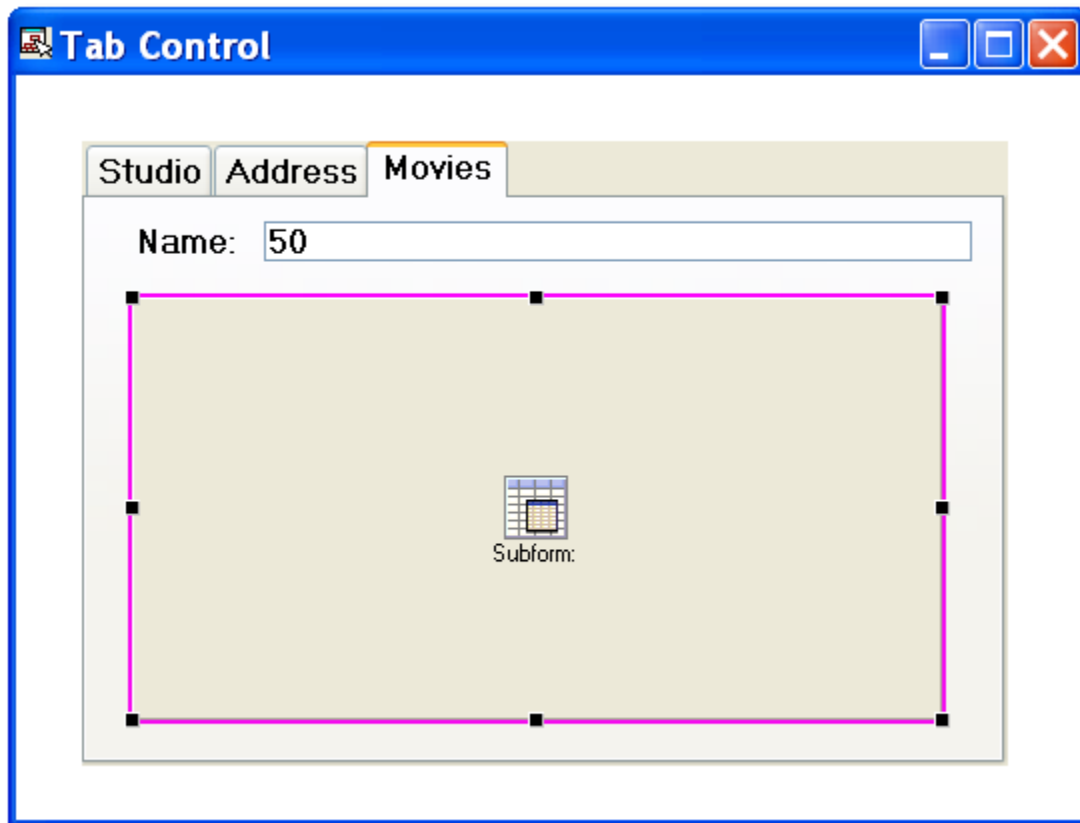



1. Select the tab control.
2. Press **Enter** until no tab is selected (as shown above).
3. Select the items you want to connect to the tab.
4. Select the link  icon, then click on the tab. The items you have linked will show up in pink.
5. Move your selected items onto the tab folder.

Now, the items will show up on all the tabs.

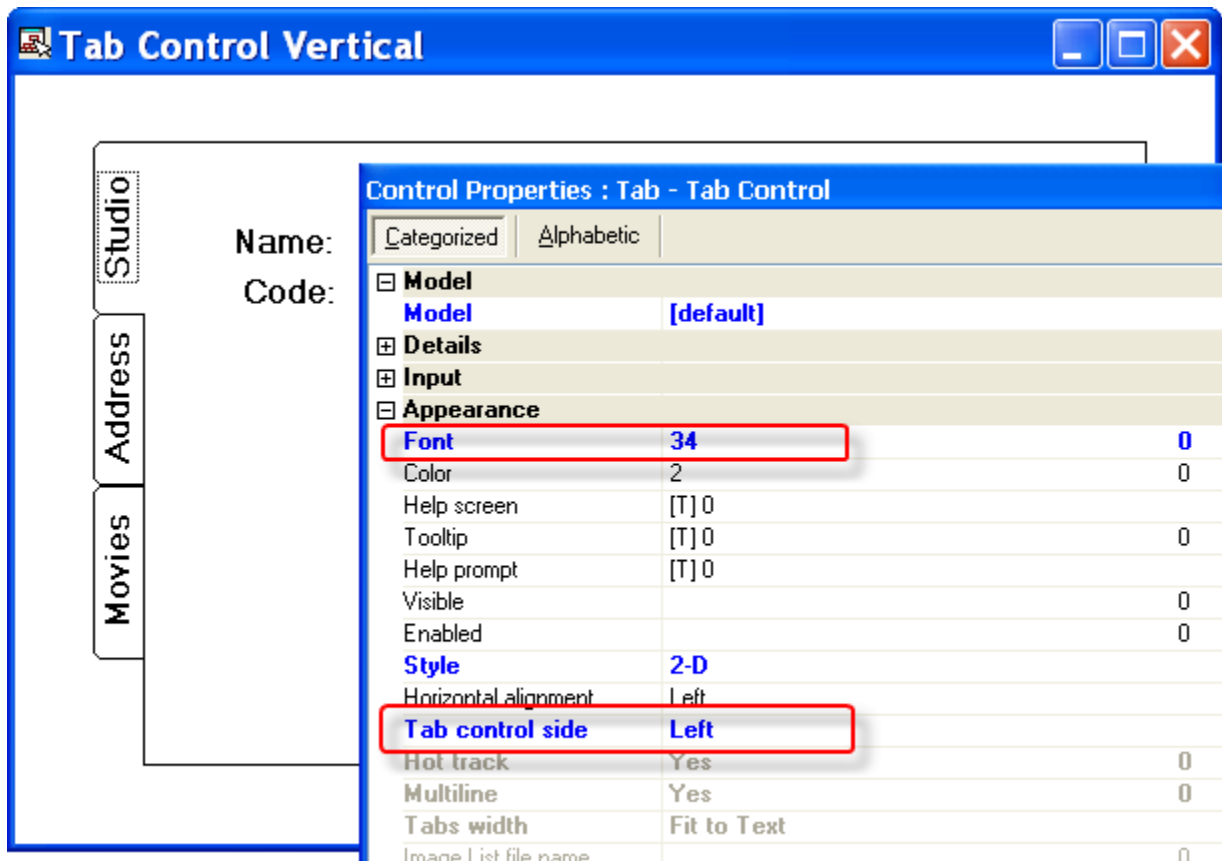
## How do I Associate a Task to a Specific Tab in a Tab Control?

Often you will want to display data on a tab that is not part of the current task. This is especially true when the data to be displayed is a table of data, or is complex data from another record. The Subtask control makes this very easy to do. All you need to do is associate the Subtask control with the desired tab.

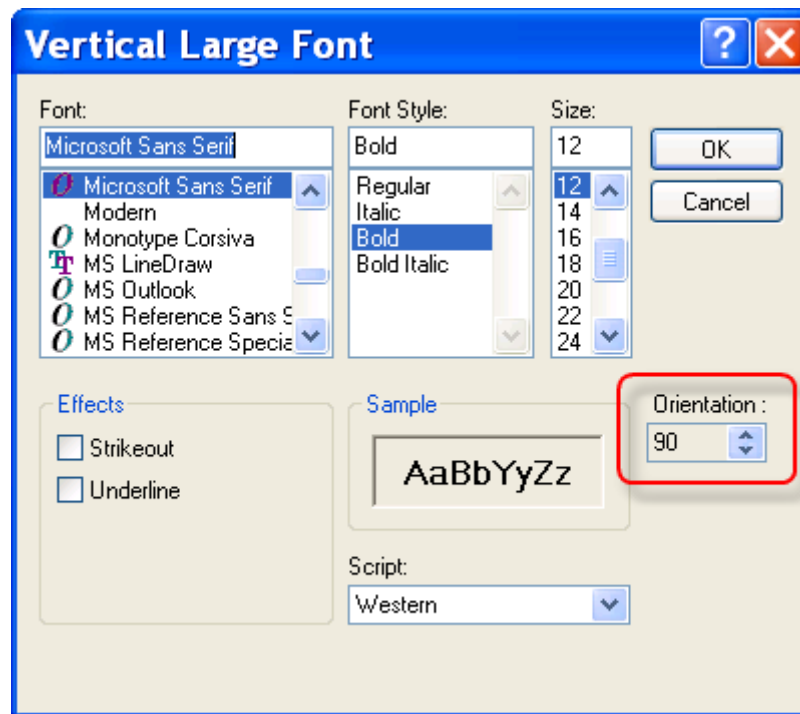


1. Select the tab control.
2. Press **Enter** until the desired tab is selected.
3. Select the Subform icon  from the Controls palette.
4. Drop the subform onto the tab.
5. Set up the subform to point to the desired subtask.

## How do I Design Vertical Tab Controls?



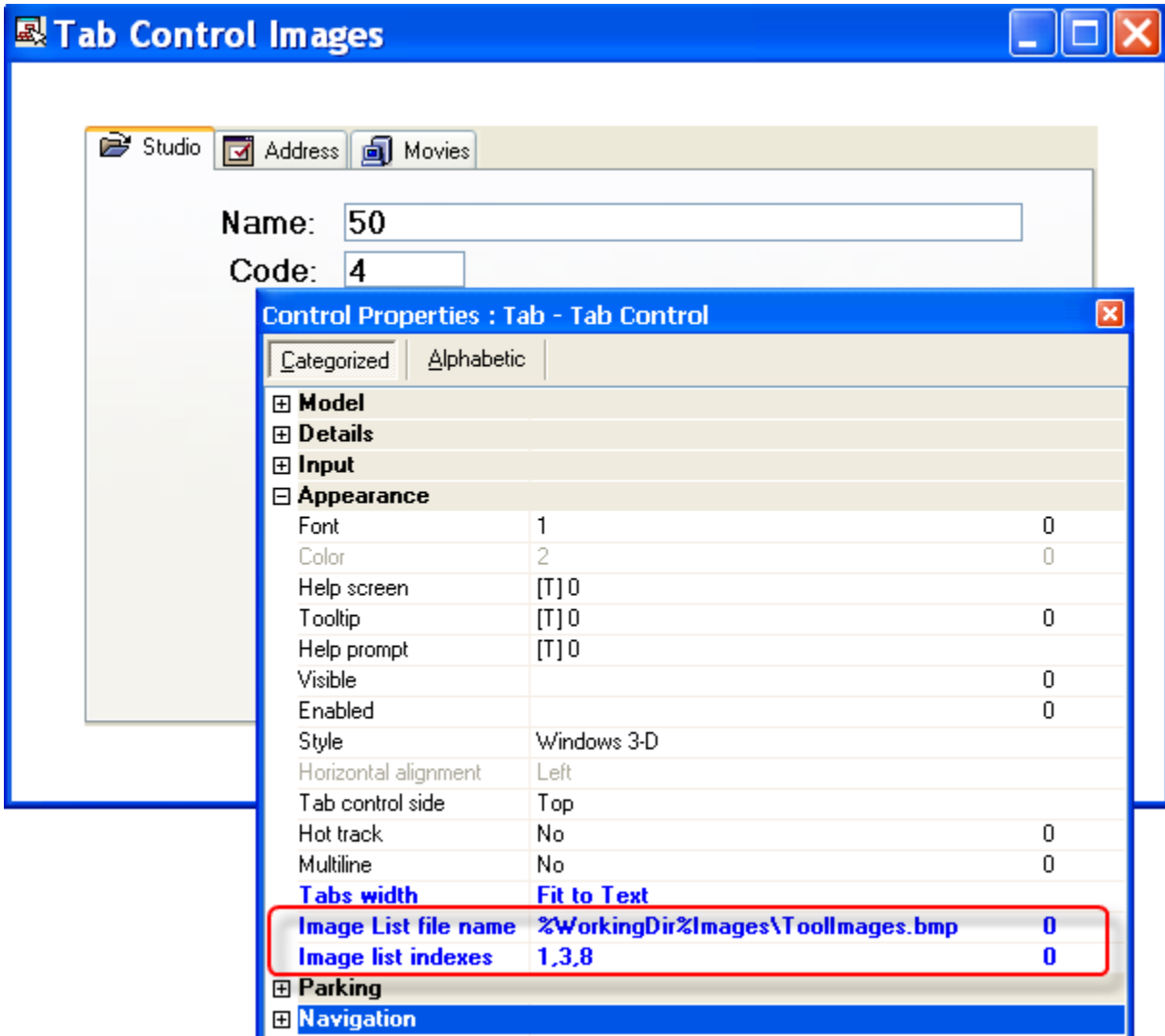
You can have the tabs on your tab control run vertically by changing the property **Tab control side** to **Left** or **Right**.



When you use vertical tabs, you also need to change the tab control font, so that the letters run the correct direction. In this example, we used a font that was rotated 90 degrees.



## How do I Associate Images to Different Tabs in a Tab Control?



You can, if you want, associate images with tabs along with the text.

You do this by specifying an *Image List file name* in the Control properties for the tab control. This image file is basically a series of images strung together into one file, similar to the images used in tree controls or for push buttons. This particular file looks like:



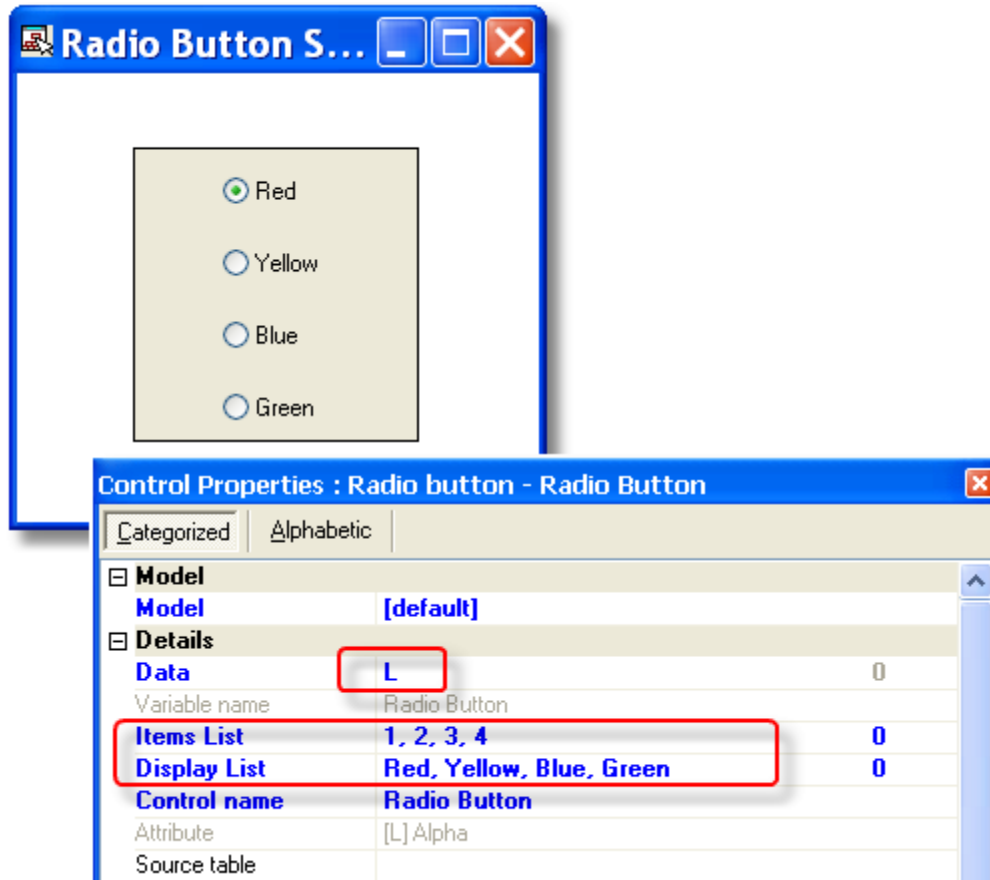
As you can see, there are a lot of images in the file (there are actually a lot more than are shown here). Each image is 16x16 pixels.

We specify which of these pictures should be used for which tab in the *Image list indexes*. In this example, we used images 1, 3, and 8 for the 3 tabs.


## How do I Implement a Radio Button?

You can implement a radio button in two different ways. You can have the button contained in one control, in which case it will look like the traditional rectangular radio button. Or, you can have it contained in several controls, which gives you more flexibility in screen design. Let's look at both options.

### Containing a radio button in one control

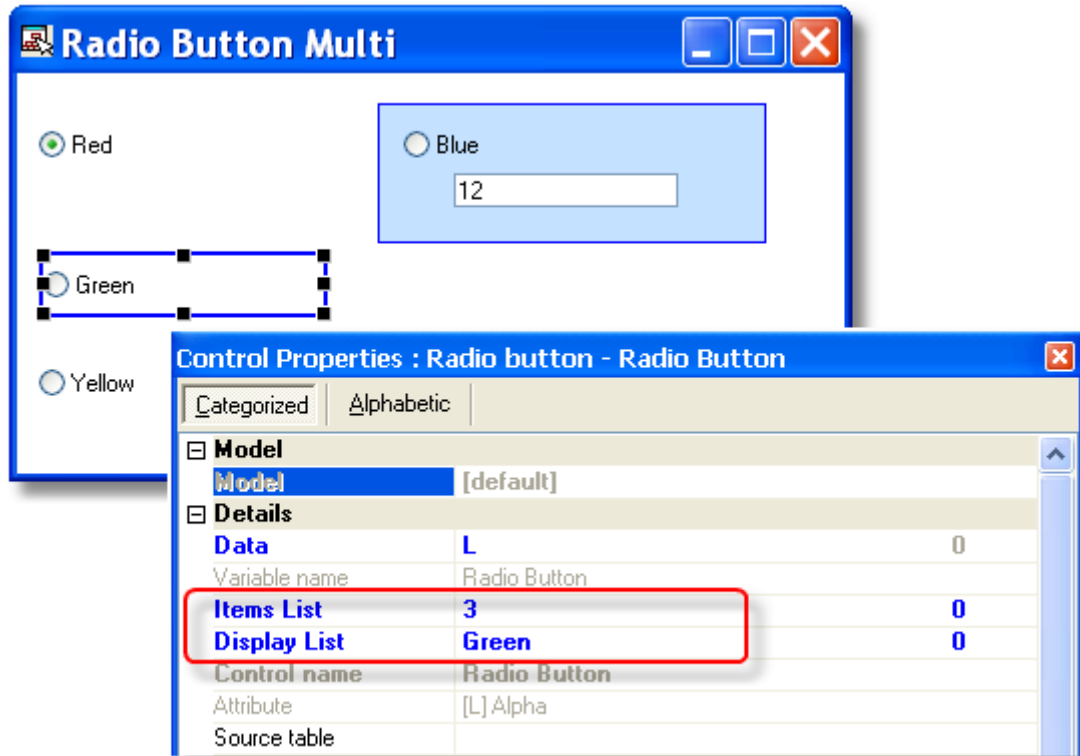


You can create a radio button that is totally contained in one control. This sort of radio button will be a rectangular box. To create this kind of radio button:


1. Create a virtual that will hold the choice. In this example, it is variable `L`.
2. Select the Radio Button  from the Controls palette, and drop it on your form.
3. Select your virtual for the Data property
4. Enter your *Items list* according to what data you want sent back to the virtual.
5. Enter your *Display list* according to what you want displayed in the Radio Button.
6. You can also set other properties to determine the number of columns, the font, and other display properties.

Now, the user will be able to choose from the valid values in the radio box at runtime.

### Using several controls for a radio button



Alternatively, you can display the radio button using several controls. This gives you the ability to spread the display over several parts of the form. To do this:

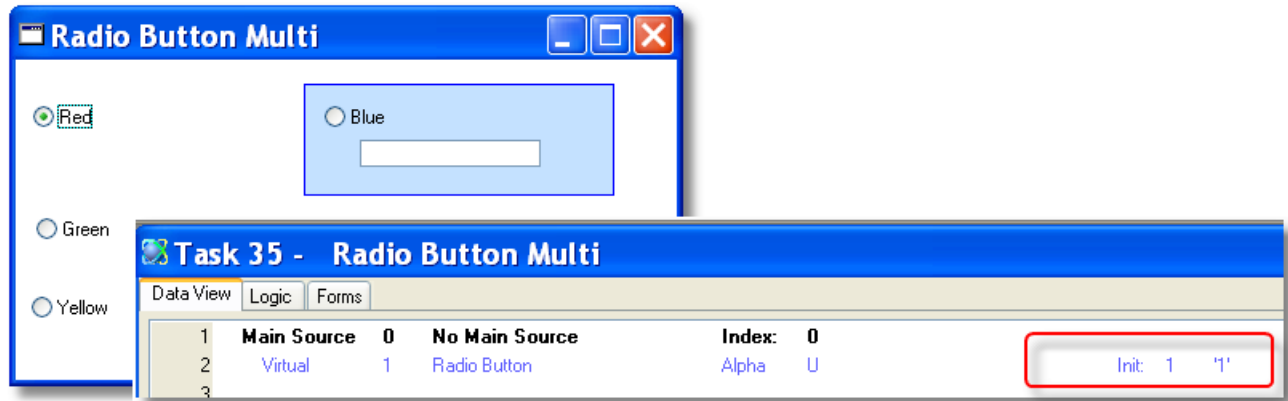
1. Create a virtual that will hold the choice. In this example, it is variable L.
2. Select the Radio Button  from the Controls palette, and drop it on your form.
3. Select your virtual for the Data property
4. Enter only one item for your *Items list* according to what data you want sent back to the virtual. In this example, the virtual will contain '3' if the color Green is chosen.
5. Enter only one item for your *Display list* according to what you want displayed in the Radio Button. This part of the radio button will display 'Green'.
6. You can also set other properties to determine the number of columns, the font, and other display properties.

Choosing the same variable for each of the radio buttons “hooks” them together, so they act as a unit. That is, if you click on one of the buttons at runtime, the previous choice is blanked out. Also, when you are

working with the buttons, they move as a unit. However, they are still separate controls and you can edit and move them separately by pressing **Ctrl-Click**.

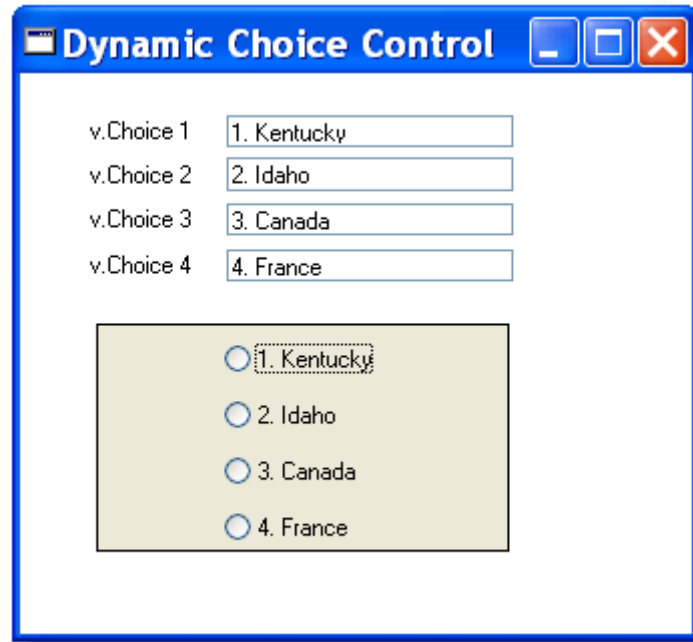
## How do I Set the Default Option for a Choice Control?

When you create a choice control, by default nothing will be selected, so the control will show as a blank, or as nothing selected. This can look strange to the user, who may expect the “first” item to be selected by default.



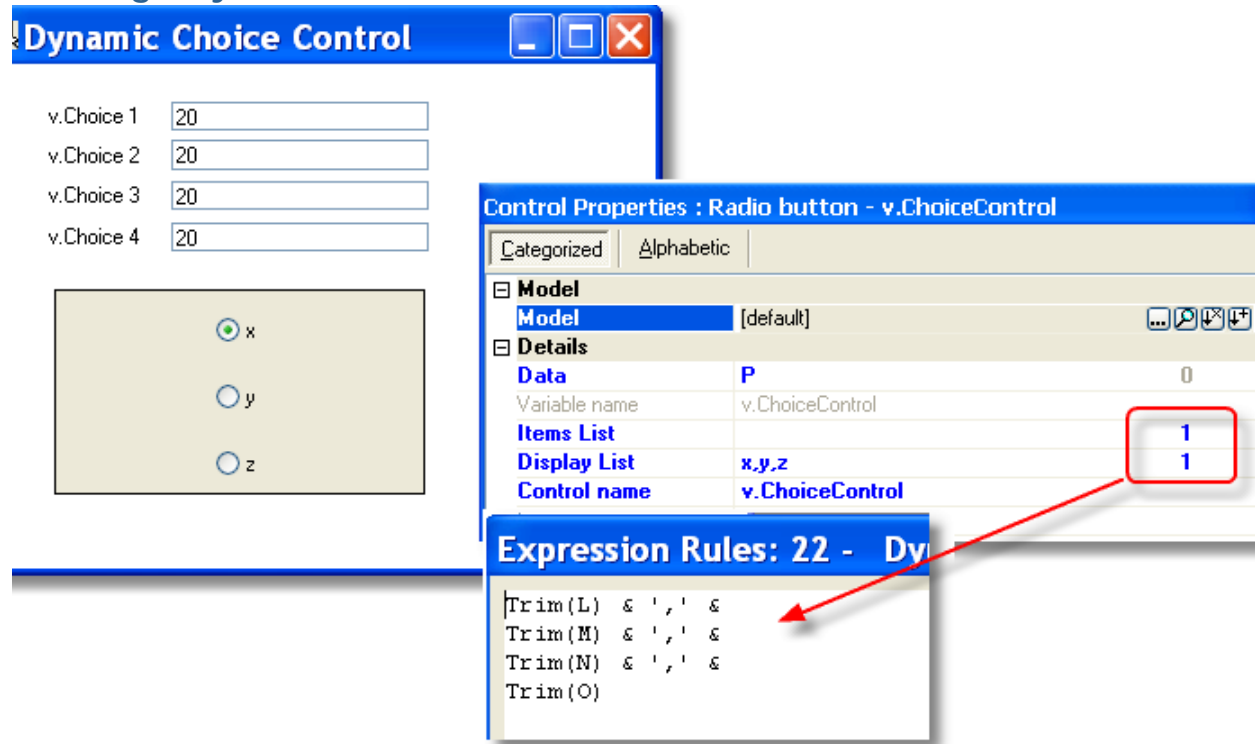
However, you can set the default option quite easily. Remember that a choice control is still a piece of data, and can be initialized like any other piece of data. In this example, the “Red” option (in the *Display List* property) is associated with ‘1’ (in the *Items list* property). So we initialize the Radio button virtual to ‘1’ in the Init column.

## How do I Dynamically Set the Option List of a Choice Control?




While you can hard-code the choices for an option list, you can also set them dynamically, using Expressions. In this example, we dynamically created the radio button from four user entries.

## Creating a dynamic choice control



To create a dynamic choice control:

1. Create a virtual that will hold the choice. In this example, it is variable P.
2. Select the Radio Button  from the Controls palette (or whatever choice control you are using), and drop it on your form.
3. Select your virtual for the *Data* property. In this example, it is variable P.
4. Create an expression for you *Items List* property, that will evaluate into the comma-delimited string you need at runtime.
5. Create an expression for you *Display List* property, that will evaluate into the comma-delimited string you need at runtime. In our example, we made it the same as the Items list.

Optionally, you can also enter some data string in the Display list. This value will show up in the Studio, and might make it easier to understand the form for the programmer. In our example we used "x,y,z".

Now, the choice control values will be determined at runtime, according to the values in the expression.



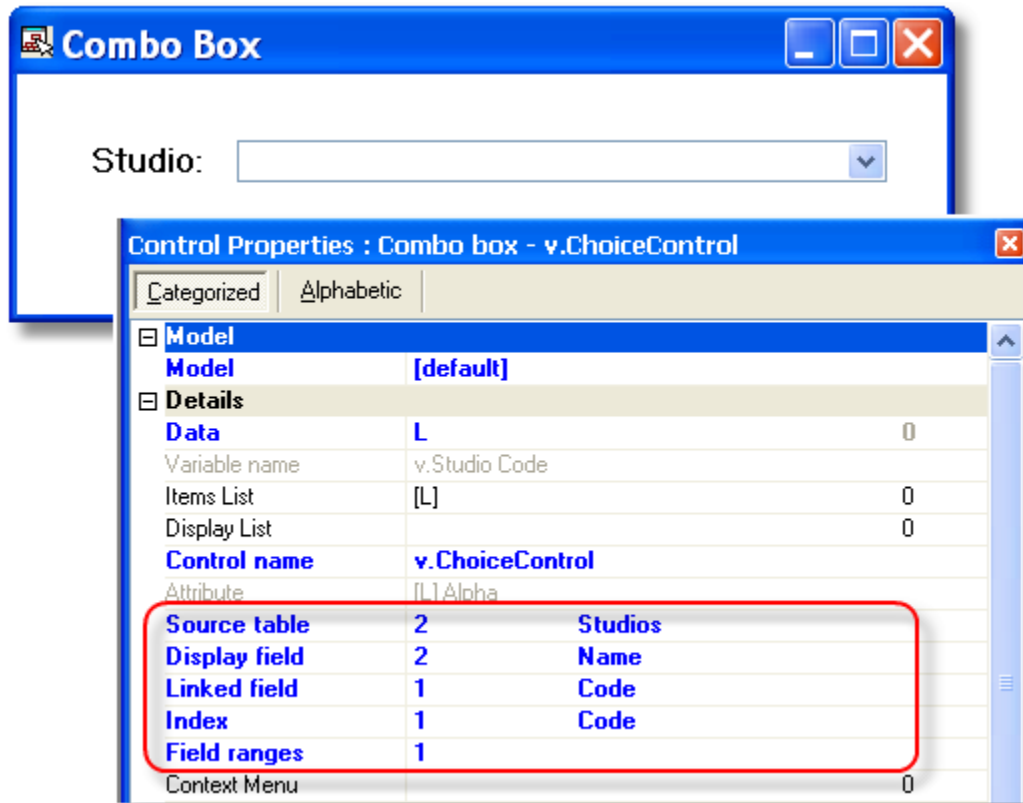
## How do I Implement a Choice Control Whose Data Comes From a Database Table?



Often it is useful to have a control where the data is chosen from a database table. That way, when the choices change because of table updates, the choices change automatically also.

In this example, we are choosing a studio from the studio table. While the user sees only the studio name, the studio code is what is brought back into the task. Let's see how to do it.

## Creating a choice control tied to a database table



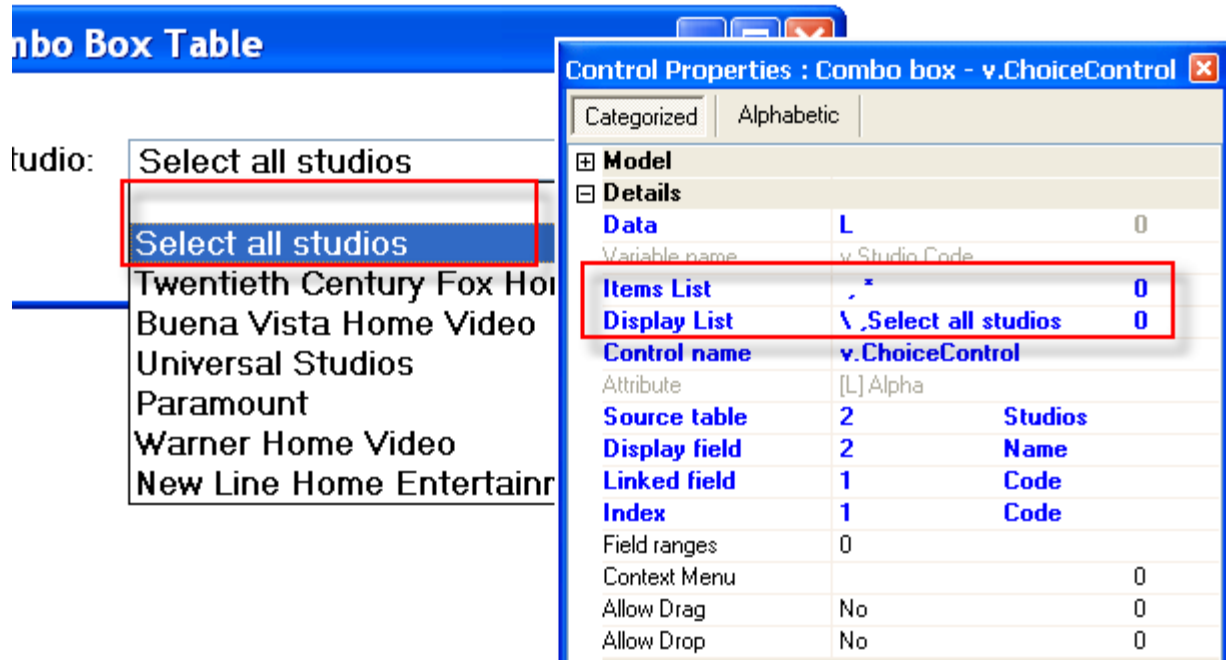
1. First, create the virtual that will hold the data. In our example, that is `v.Studio Code`.
2. Select the Combo Box (or other choice control) from the Controls palette and drop it onto your form.
3. Select the virtual for the Data property of the choice control (`L`, in our example).
4. Zoom from the *Source table* property to select the table you want to use. In our example that is `Studios`.
5. Zoom from the *Display field* property to select the field you want to show to the user. In our example, we used `Studio Name`.
6. Zoom from the *Linked field* property to select the field you want brought back in to the virtual. In our example, that is the `Studio code` field.
7. Zoom from the *Index property* to select the display order of the fields. If you are using Field ranges, it is important that the index match the range, or you will have performance problems. For instance, you don't want to have the index search the table by `Studio code`, while your field range is filtering by `Zip code`.
8. Zoom from *Field ranges* to limit the number of entries in the choice control, if applicable. For instance, we might want only the `Studios` that are located in California.

Now, when the task is run, the choice control will be populated by live data from the table.

**Note:** If the table is large, using the table in a choice control can present an performance issue. This option is best used for smaller tables, or tables where there are good efficient indexes. Also, note that the tables used in choice controls do not show up on the declared tables list.

## How do I Combine Additional Options With a Data Bound Choice Control?

A Choice control often gets its options from the data. For instance, the control may get the options from the Range property of a virtual, or it may link live to a table. However, for Combo boxes and List boxes, you also have the option of adding additional values to the list, in addition to the values that come from the Range or Data source table.



In this example, we are linking to Source table 2, which is our Studios.

1. We use the *Display List* property to add two more items to the list: a blank (specified by adding backslash and a space), and the words "Select all studios".
2. In the *Items List*, we also add two entries, a blank (if a blank is selected), or an asterisk (if all studios are selected).

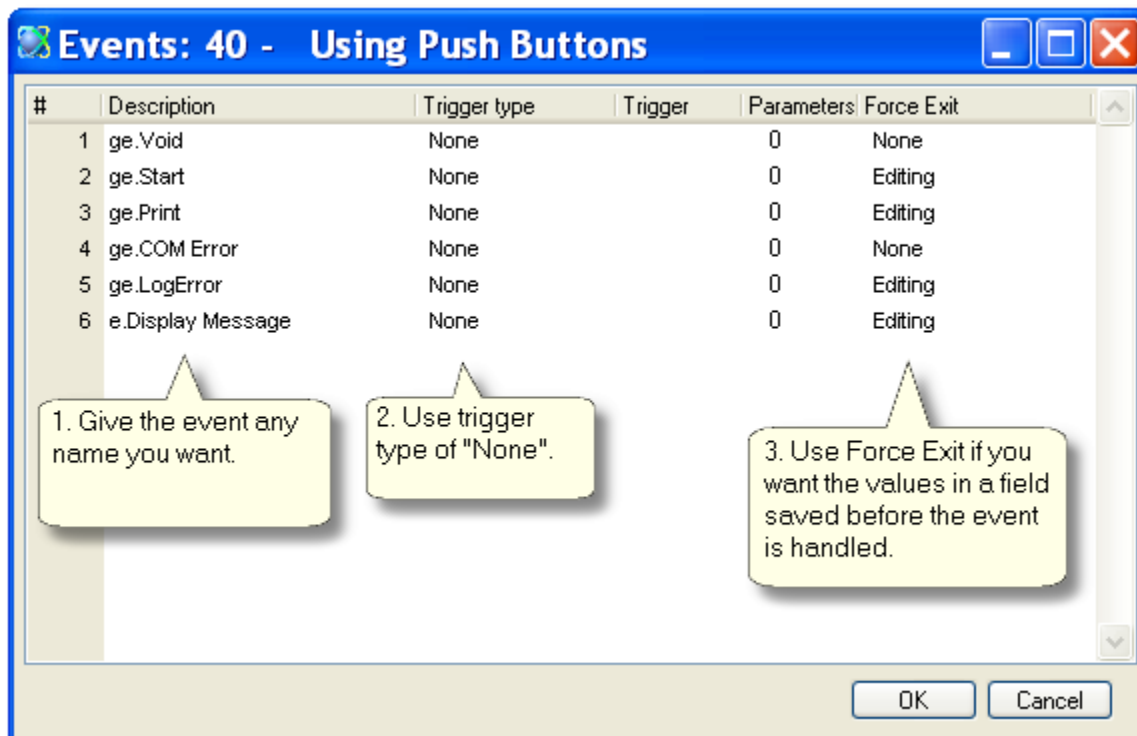
## How do I Implement Logic With Push Buttons?

There are several steps to implementing logic with push buttons:

1. Create you user events
2. Put your push buttons on the form
3. Create a logic unit to perform the desired logic when the event is raised.

Let's see how to do each of these steps.

### 1. Create your user event



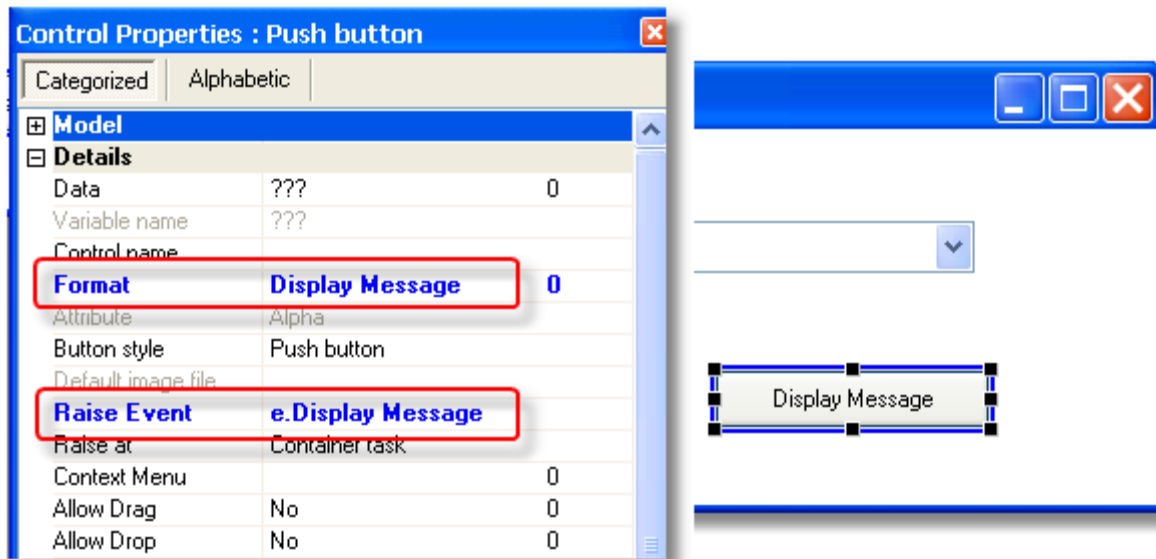
First, you need to create you user events. You do this in **User Events** (**Ctrl+U**). Press F4 to open up a line, and fill in the fields as follows:


1. In the *Description* column, give the event any name you like.
2. Select **None** for the *Trigger type*. You don't need a trigger since this event will be raised by the push button directly.
3. For *Force Exit*, you will usually want **Editing** or **None**. Force exit = Editing causes the value in the edit field to be saved into the variable before the push button is pressed, which is generally what you want.

Enter as many events as you need for your push buttons.

**Note:** You can also just reuse global events. In the example above, the global events are marked with the prefix “ge.”. We have some standard global events that we use in many programs, such as `ge.Print` and `ge.Start`. If you use global events, you can even attach them to models, so you can automatically create buttons that raise the desired event.

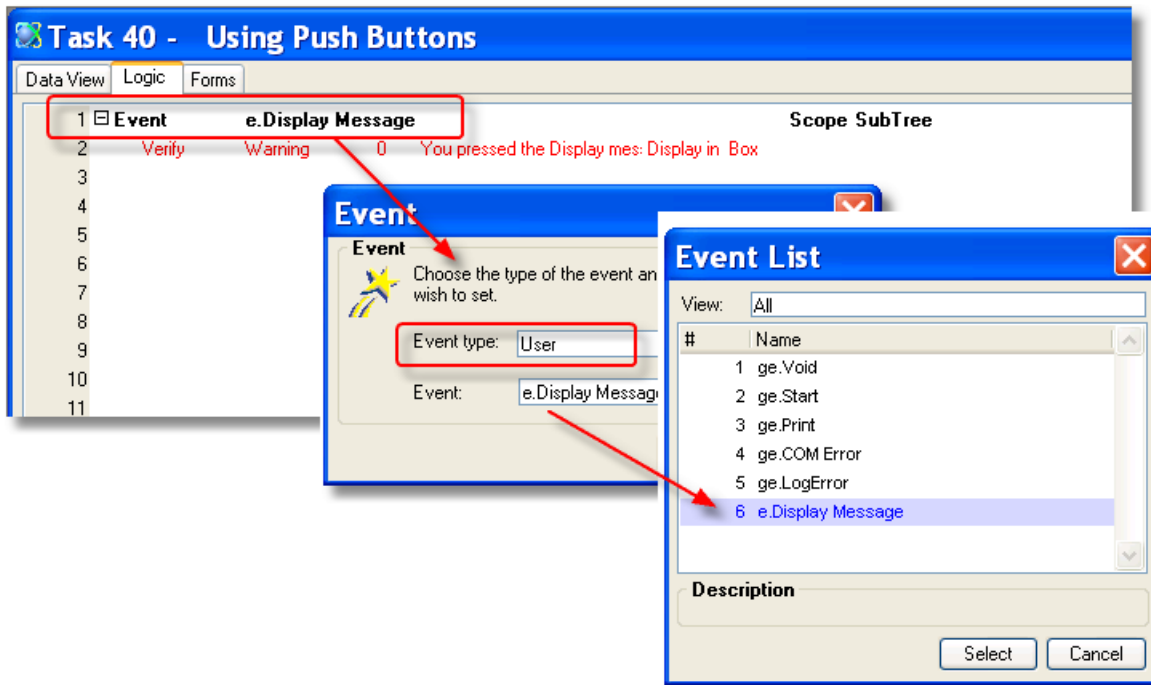
## 2. Put your push buttons on the form



1. Select the push button control  from the Controls palette by clicking on it.
2. Click again on your form to drop the push button there. Resize it as needed.
3. In the push button *Control properties*, type in the *Format*. This will be the text that displays on the push button.
4. In the push button *Control properties*, zoom from the *Raise Event* property, to select your event.

Now, when the user presses the push button the event will be raised. In our example, when the user presses the *Display Message* button, the event *e.Display Message* will be raised. Now we need to create a logic unit to handle the event.

Create a logic unit to perform the desired logic when the event is raised

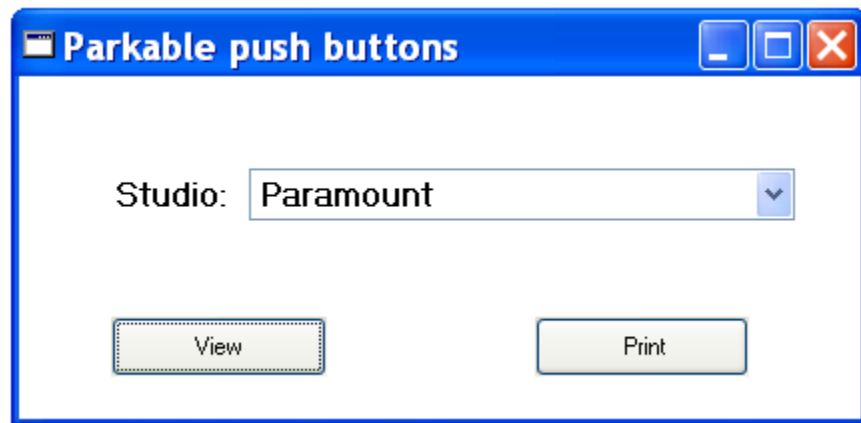


1. Go to the *Logic* tab.
2. Press **Ctrl+H** to open up a header line.
3. From the combo box, select *Event* (or type E). An event box will appear.
4. Select Event type: *User*, then tab.
5. An event list will appear. Choose your desired event.
6. Use the lines under the Event logic unit to add operations that will be performed when the event is raised.

Now you have an operational push button.

**Note:** In our example, we used a non-tabbable push button. You can also attach push buttons to variables so you can tab to them, by specifying the variable in the Data control property. When you do that, the Format property needs to be specified slightly differently, because the Format actually becomes a Picture for the data field. See Chapter 13, “How do I Allow Keyboard Navigation to a Push Button?” on page 324.

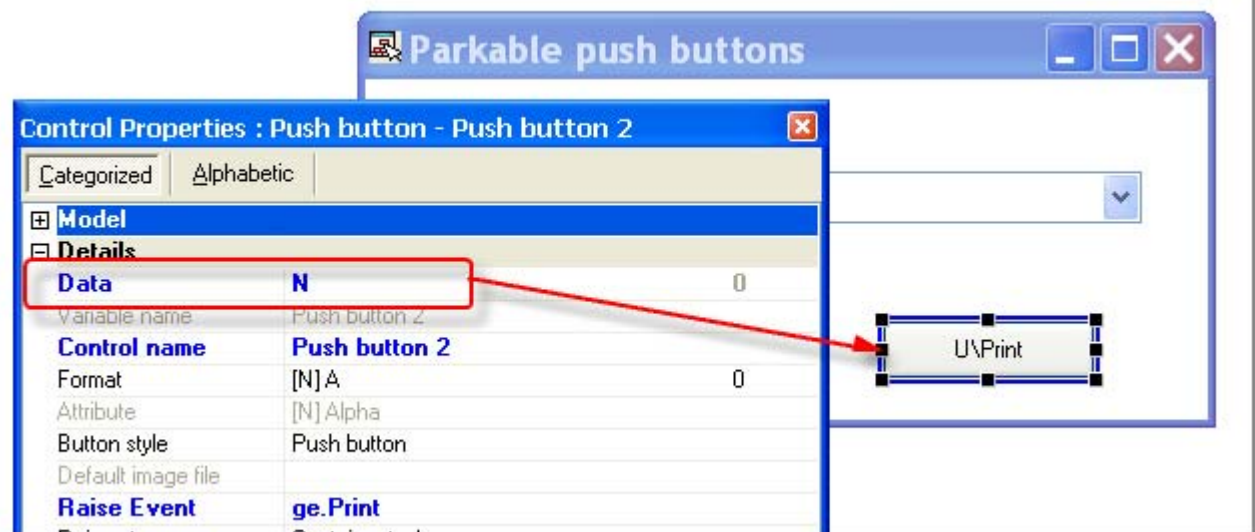
## How do I Allow Keyboard Navigation to a Push Button?



You can specify a push button by simply dropping one on your form, and setting the push button control's Raise Event property to whatever event you want. The push button will work whenever the user clicks on it with the mouse, or uses an accelerator key (hot key) to push it. However, the user will not be able to tab into the push button.

If you want a push button that can be tabbed into, you need to associate the push button with a variable. You do this by specifying a variable in the Data property of the push button control.


### Creating a parkable push button



1. In the Data View tab of the task, create a virtual that you will use for the push button
2. Drop a push button onto your form.

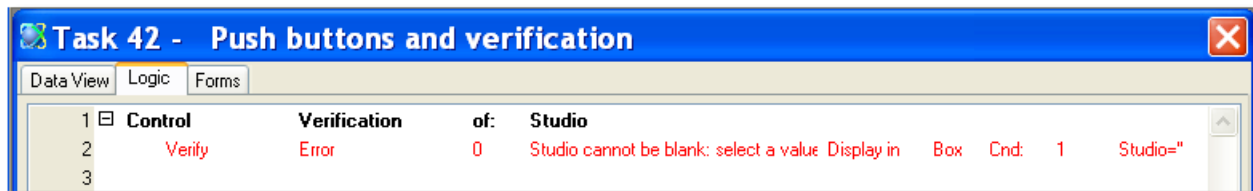


3. From **Control Properties->Data**, zoom to select the virtual for this button.

Now, the user will be able to tab to this push button. When you select *Display Tab-order* , the button will show up with the number in red, indicating it is tab-able and that you can change the tab order if you want.

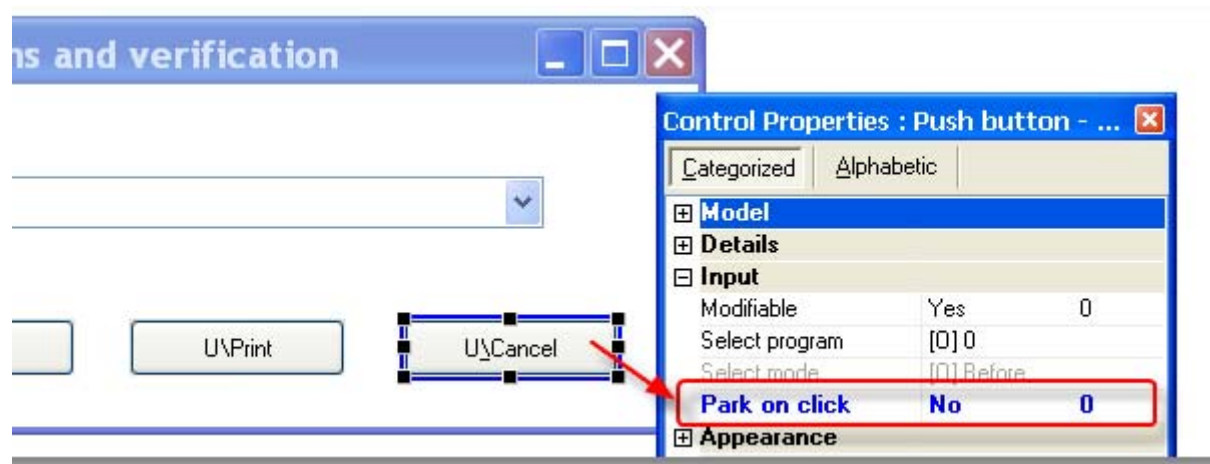
**Note:** When you create a tab-able push button, the text on the button is actually text within the variable, and needs to be specified differently than for a non-tabable push button. There are several methods for doing this: see Chapter 13, “How do I Specify the Text on a Parkable Push Button?” on page 330.

## How do I Skip Verification Logic From Being Executed When a Push Button is Pressed?



Often, controls on a form will have verification logic attached to them. For instance, in this example, we cannot pass by the Studio control without selecting a studio. Normally, this is what we would want, so the user does not try to print, for instance, without having selected a record to print.

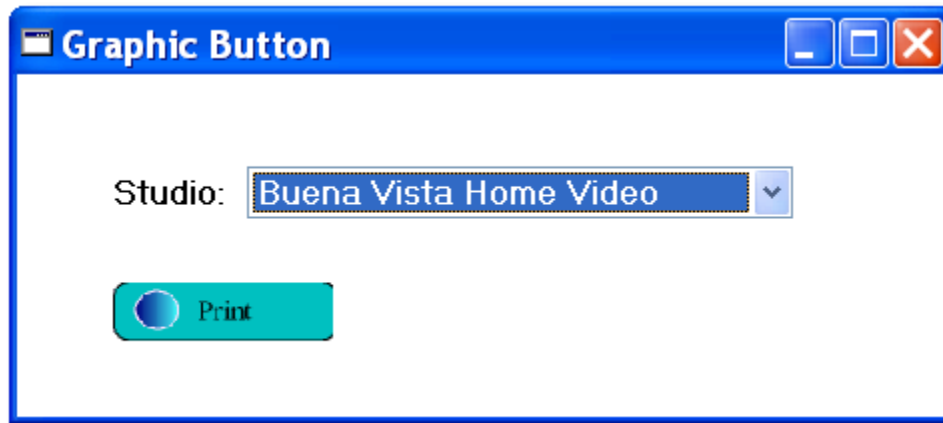
However, sometimes you may also want the user to be able to reach a push button even though the screen has errors. For example, you might want to allow the user to reach the “Exit” or “Cancel” button.



The way you do this is to set the *Park on click property* to **No**. The push button will work as before, but the logic units for the controls that were “skipped over” will not be executed.

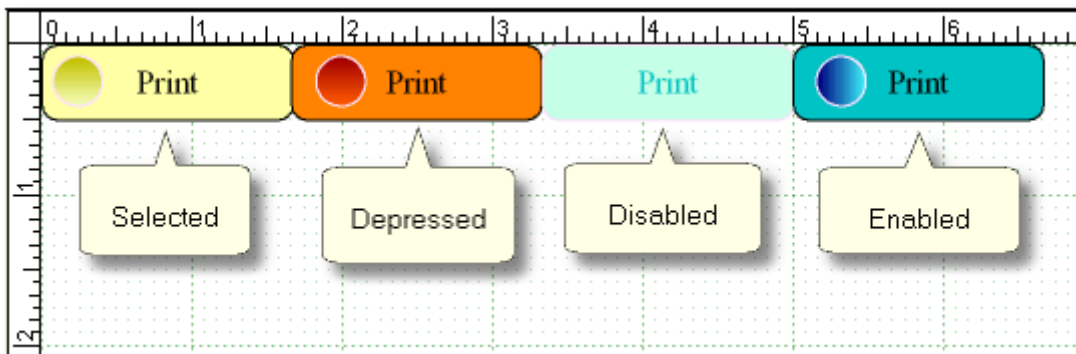
**Note:** This doesn’t have an effect when the user is using the keyboard. So, if the user tries to tab past a blank “Studio” field in our example, the user will still get an error message. For this reason we used an accelerator key with our push button, so **Alt + C** will push the button, for keyboard-centric users.

## How do I Create Image Buttons?

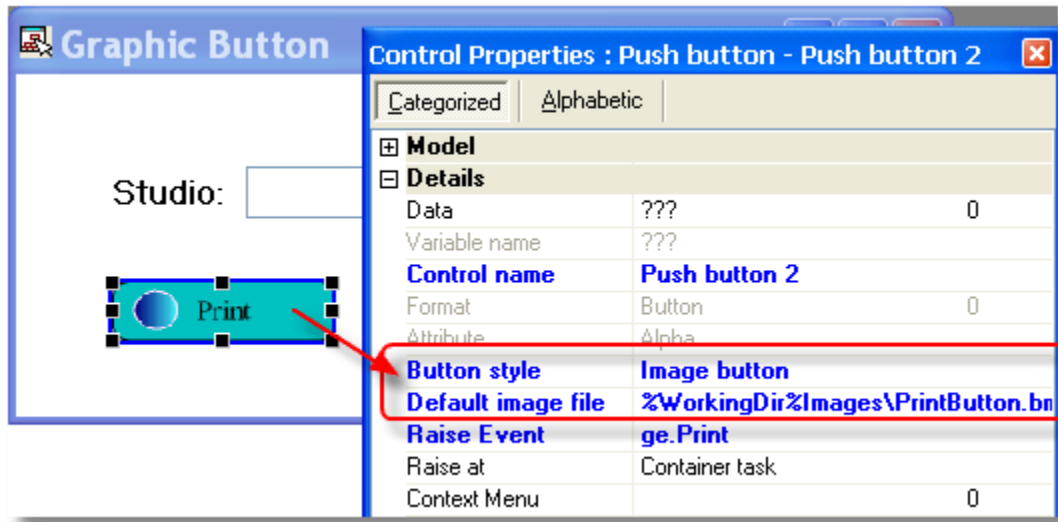


You can use graphic image buttons in eDeveloper if you like. A graphic push button consists of a .bmp file, that has 4 sections of the same size. Here we created one in an image editor, by repeating the same image 4 times and setting each to a different set of colors.

At runtime, eDeveloper uses a different part of the .bmp file, depending on what is going on with the button. The fourth section is the one that shows normally to the user, when the button is not being parked on and it is enabled. Section 3 appears when the button is disabled. Section 2 appears when the button is being pressed. Section 1 appears when the user tabs into the button (which only occurs if it is a button tied to a variable).



## Creating an image button

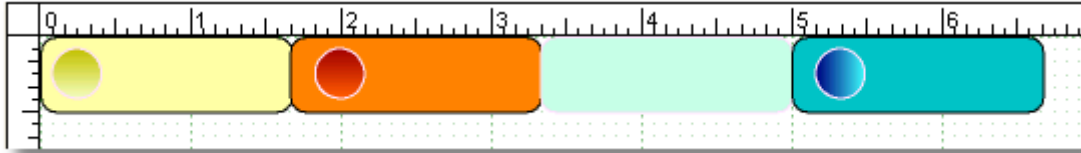


1. Drop a push button onto your form.
2. For the *Button style* control property, select **Image button**.
3. For the *Default image file*, zoom to select the file. Or, preferably, enter the file name using a logical name to point to the correct directory.

Now, eDeveloper will use the specified .bmp file to display the push button.

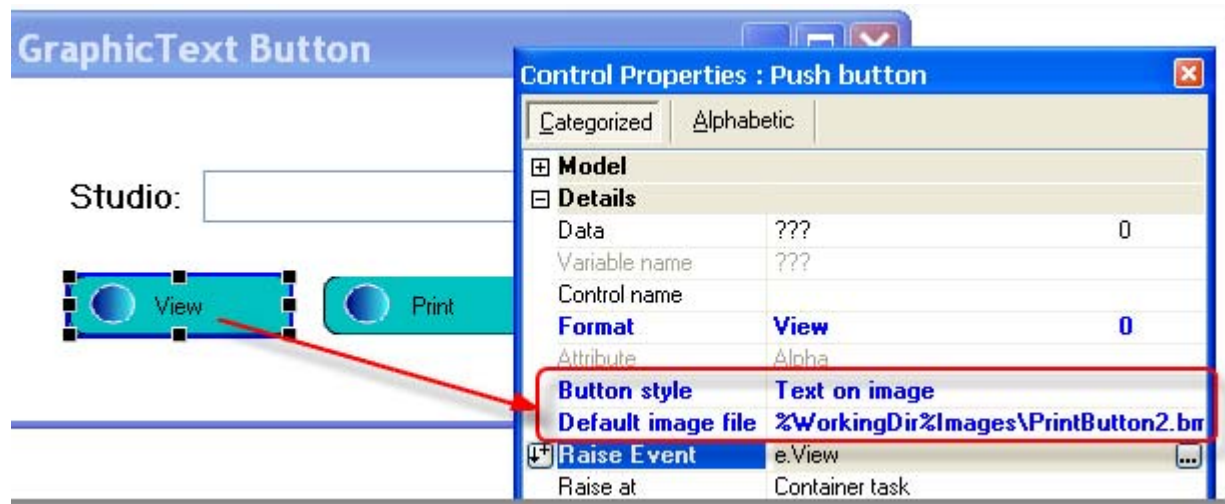
## How do I Combine Image and Text on a Button?

It is often useful to use the same background image for push buttons, while specifying the text for the button in eDeveloper. This allows you to use fewer .bmp files, and also to do language translation at runtime.



The push buttons used for a Text on image button are the same format as those used in an image button, but without any text added. See Chapter 13, “How do I Create Image Buttons?” on page 327 for more information on how to create these.

### Specifying a Text on image button



1. Drop a button onto your form.
2. In **Control Properties->Button style**, select *Text on image*.
3. In **Control Properties->Default image file**. Or, preferably, enter the file name using a logical name to point to the correct directory.
4. In **Control Properties->Format**, type in the text you want to show on the button.

Now, your button will appear with your text superimposed on it.

## How do I Specify the Text on a Parkable Push Button?

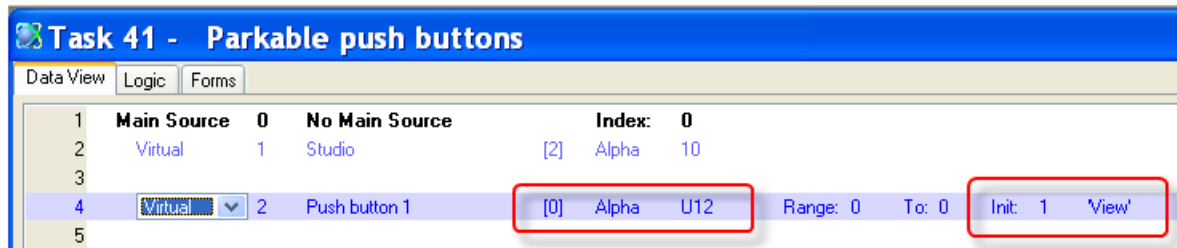
When you are creating non-parkable push buttons, you specify the text on the button simply by typing it into the Format property for the push button control. However, on a parkable push button, the push button is actually attached to a variable, and the contents of the variable are what displays inside the push button.

So, for this kind of button you can specify the text:

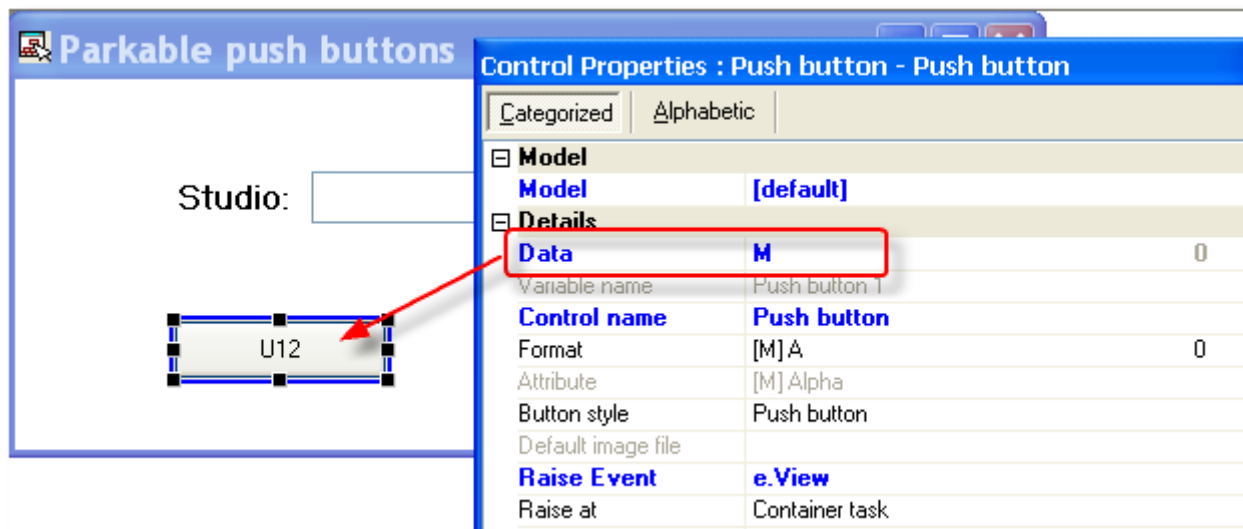
- By specifying an Init on the variable
- By specifying a Default value on the variable
- By changing the Format picture on the variable's push button control

We will show you how to do each of these below.

### Using an Init to specify push button text



1. Create an alpha virtual, that is long enough to hold the text of the push button.
2. Use the Init: property to specify the text. Here it is 'View'.



3. Drop a push button onto your form.

- 4. Select your virtual in the Data property of the push button. When you view this button in the Studio, you will not see the text
- 5. When you see the push button in the Studio, it will not display any text; you will just see the data picture. However, it will display properly at runtime.

Using a default value to specify push button text

Task 41 - Parkable push buttons

Data View

Logic

Forms

	Main Source		No Main Source		Index:	
1		0			0	
2	Virtual	1	Studio	[2]	Alpha	10
3	Virtual	2	PB with default value	[0]	Alpha	U12
4						
5						
6						

Local Variable Properties Alpha : ...

Categorized

Alphabetic

Model

General

Details

Input

Appearance

Style

Def/Null

Null allowed

No

Null value

Null display

Null default

No

Default value

Cancel

Database default

This works exactly like the option using an Init. However, instead of specifying an init, you type the text into the Default value property of the variable.

This method is good in that you can use it as part of a model. However, it is not very obvious where the button is getting its value, since you can't see it in the Data View section.

Using the picture to specify push button text

Task 41 - Parkable push buttons

Data View

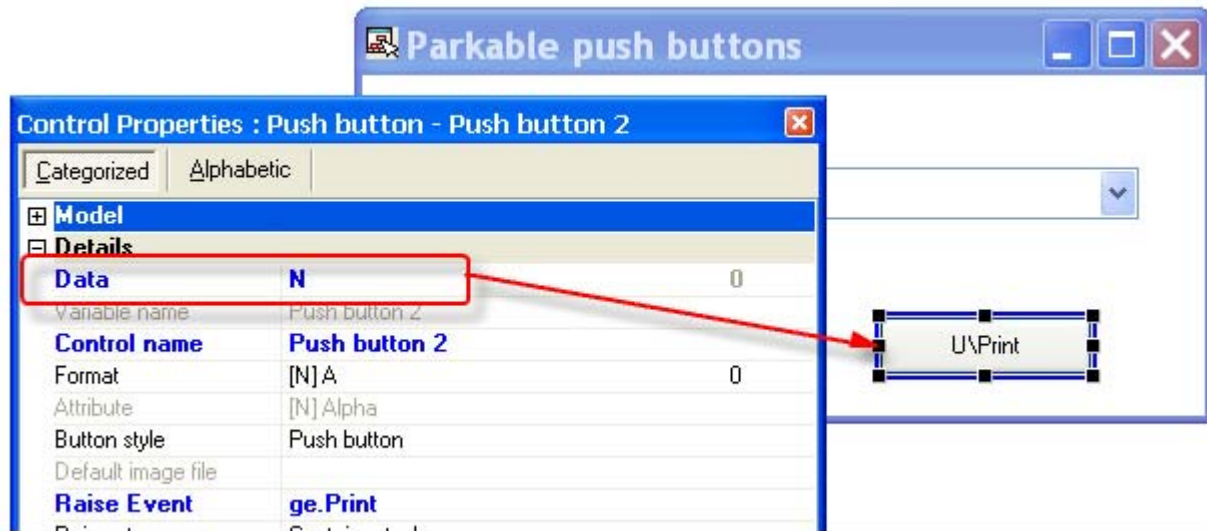
Logic

Forms

	Main Source		No Main Source		Index:			
1		0			0			
2	Virtual	1	Studio	[2]	Alpha	10		
3								
4	Virtual	2	Push button 1		Alpha	U12	Init:	1 'View'
5								
6	Virtual	3	Push button 2	[0]	Alpha	U\Print	Range:	0 To: 0 Init: 0

- 1. Create an alpha virtual.

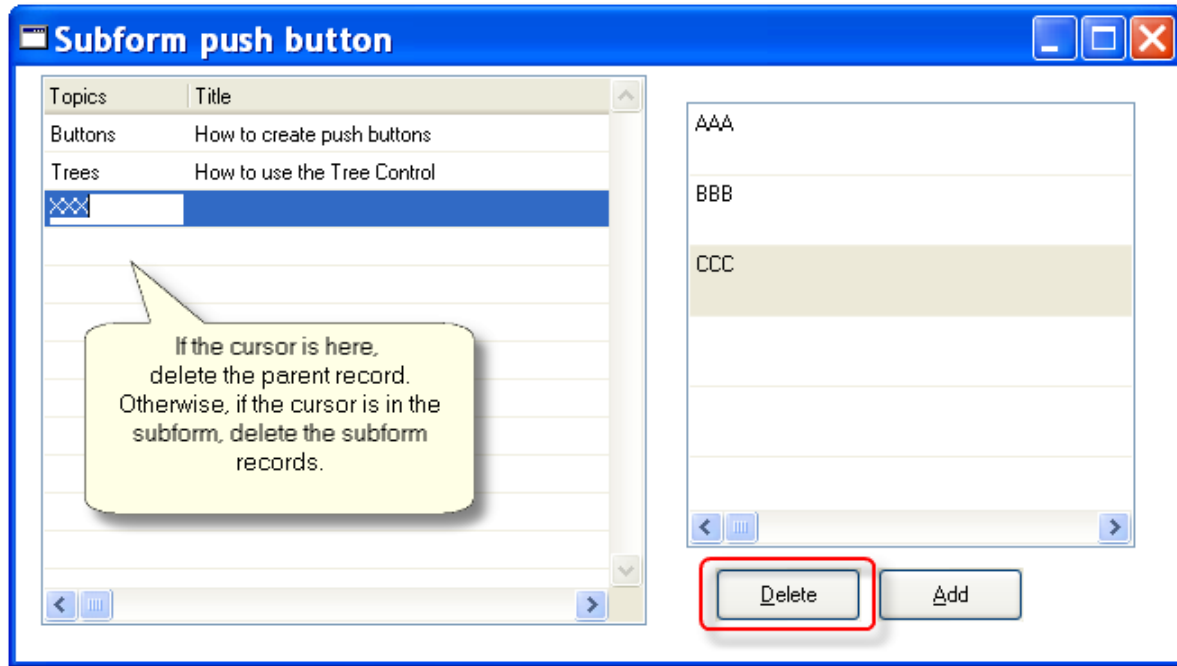
2. Specify the Picture of the virtual so that it has at least one placeholder character (in this case, a “U”). Use the rest of the Picture to specify the text that should be on the button. Upper case characters need to be preceded by a backslash.



3. Drop a push button onto your form.
4. Zoom from the Data property to select your alpha virtual.
5. Now you will see that the push button inherits the format from the virtual. Using this method, you can see the text of the button while you are working in the studio.



## How do I Set Up One Push Button to Affect Either the Subform or its Parent Task?



Sometimes a given push button can be expected to do different things, depending on where the current focus is. For instance, if you have a parent task that is sitting on a list of records, and each record has some child records, a “delete” button would be expected to delete the parent record or the child record, depending on where the cursor was parked.

For instance, in the example above, if the “Delete” key is pressed, should that refer to the “XXX” record, or the “CCC” record? The user would probably expect that since the cursor is on “XXX”, that the parent record will be deleted (along with all its children). However, this is actually forcing the push button to do double duty, depending on where the cursor is parked.

eDeveloper has a property to handle this, which is called *Raise at*. It has two values:

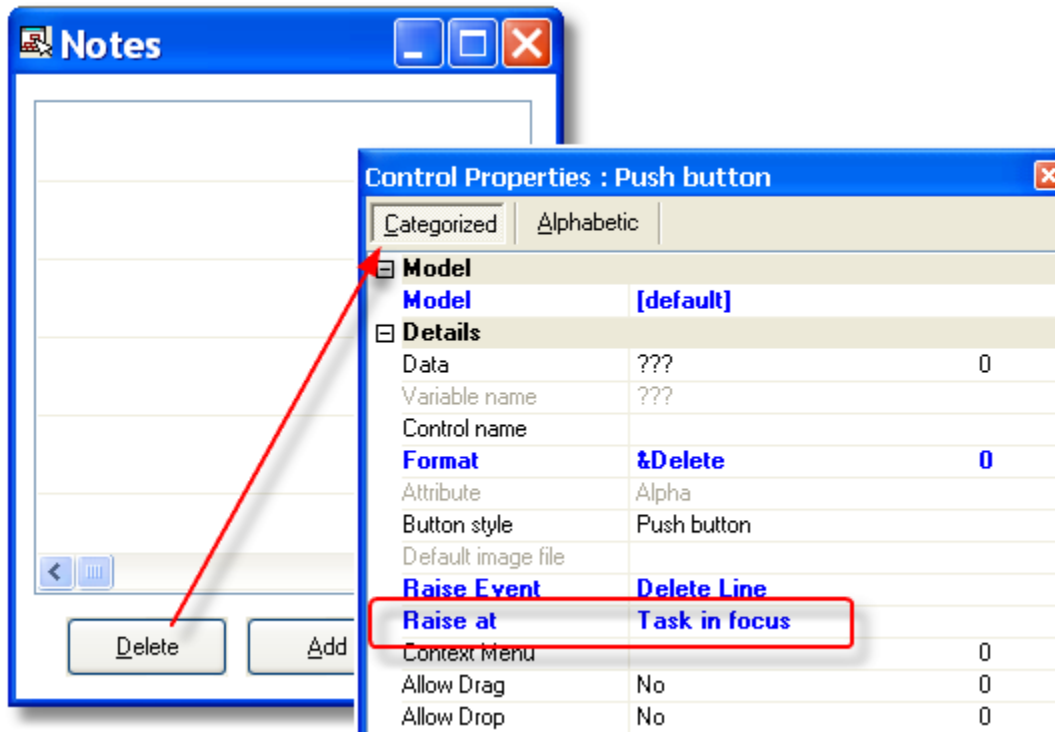
*Container Task*: the event will trigger in the task that has the push button.

*Task in focus*: the event will trigger in either the parent task or the subtask, depending on which has the focus.

*Task in focus* will do exactly what we need in this instance.

**Note:** The Task in focus option is only applicable for non parkable buttons (Allow Parking=No).

## Using the Task in focus property



1. Go to the push button you need to alter.
2. For the *Raise at* property, select *Task in focus*.

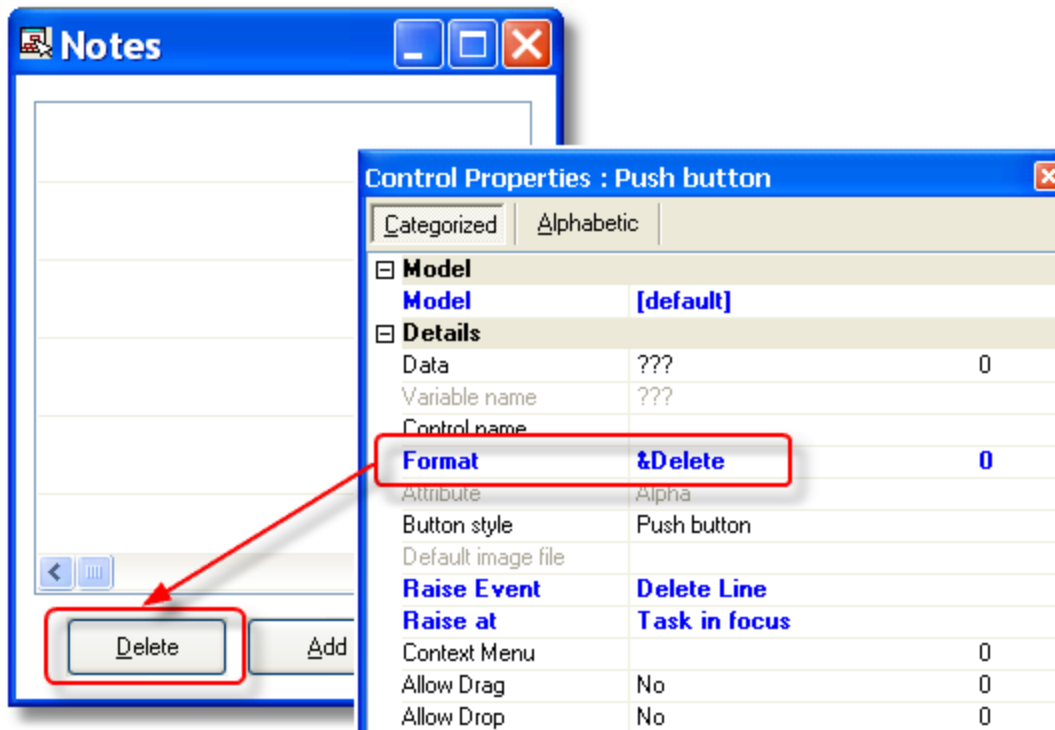
Now, the event will be raised at either the parent or the child task, depending on which currently is in focus.

## How do I Set Accelerators to Push Buttons?

It is useful to set *accelerators* (aka *hot keys*) to push buttons, because many people are more focused on the keyboard than the mouse, especially folks whose job is data-entry. This is especially important when you are using non-parkable push buttons; if you use an accelerator, then the user does not need to remove her/his hands from the keyboard to pick up the mouse.

Fortunately, eDeveloper makes this easy. If you precede any letter in the button text with an ampersand (&) then that key will become an accelerator key. For instance, a button containing &C will have C as part of the text, and **Alt+C** will press that button.

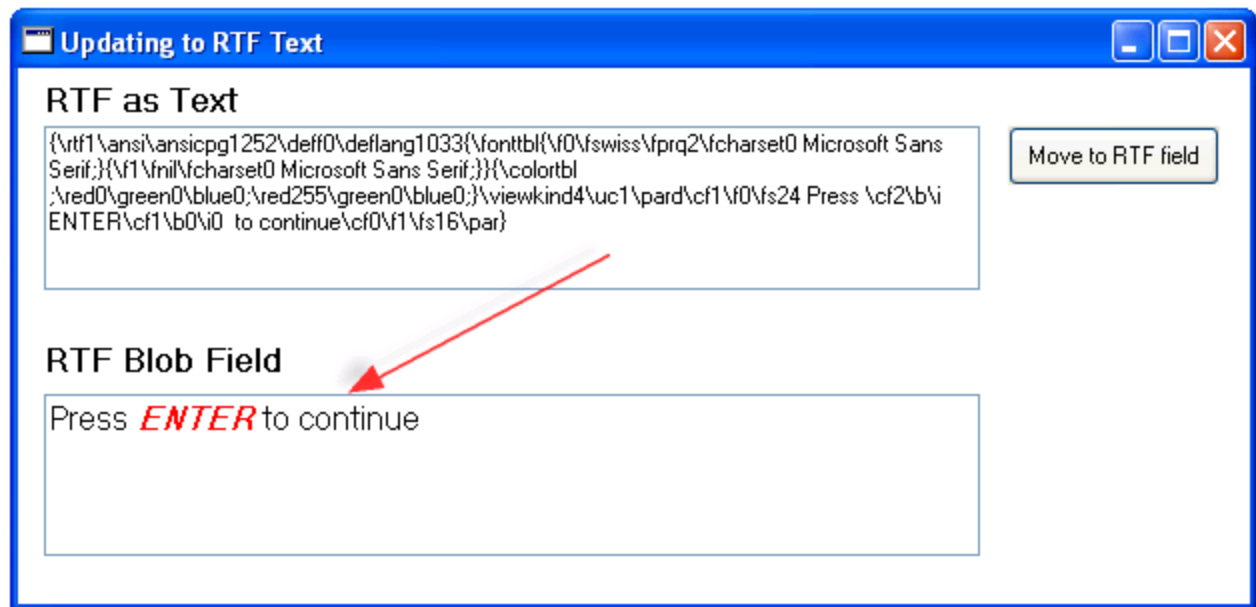
### Setting an accelerator to a push button



1. Go to **Control Properties->Format**.
2. Insert a & in front of the character you would like to use for an accelerator key. In this instance, that is the D.
3. That character will then be underlined onscreen, and Alt+ that letter will press the button.

**Note:** If you are using a parkable button, the syntax will be slightly different, because the first character is usually the accelerator key, and it is usually also capitalized and therefore will need to be preceded by a backslash. So, if this were a parkable button, the syntax would be U&\Delete.

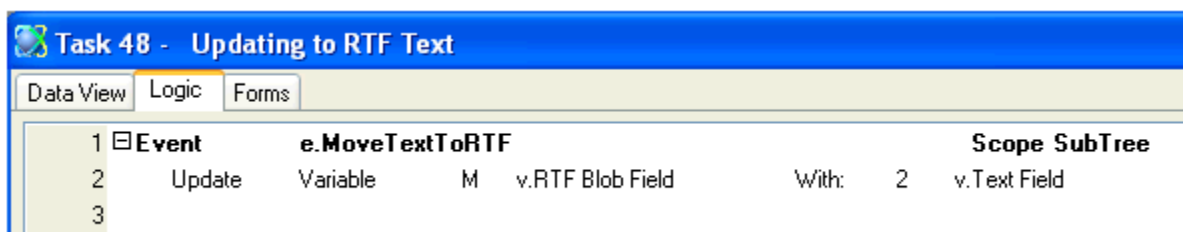
## How do I Dynamically Create Rich Formatted Text?



Rich Formatted Text, or RFT, is basically just a text markup language, like HTML or XML. Therefore, you can move the formatted text to a variable, and it will be displayed by eDeveloper with the fonts and colors you would expect.

The user can edit an RTF field at runtime by using options on the right-click menu. For most applications, you only need to provide the RTF field on the form, and the user will format the data. If you need to initialize the field, you can initialize it with some simple text by must using an Init with some alpha data. Or you can use the Default Value property of the variable.

However, if you want to, you can initialize the field with nice formatted text, as shown in this example. Here we have some RTF text that is held into a text field. When the user presses the button, an Update operation copies the value into a BLOB field, as shown below.

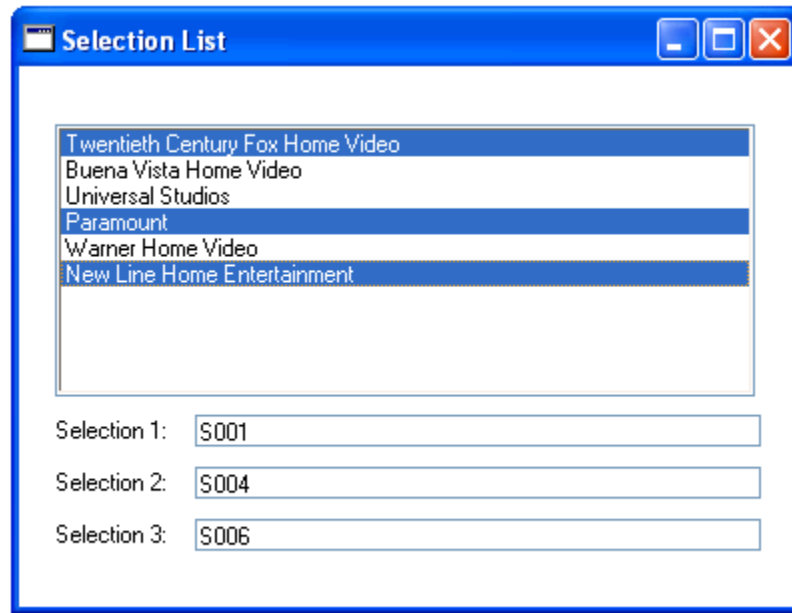


You don't need to do anything special to move the text into a BLOB field. You can use an Update operation or an Init, or use an expression in the Control's *Data* property.

To have the Blob display on the form, select the RTF Text  edit control, or set up the BLOB variable with the GUI Display Property = Rich Edit.

**Hint:** *You do not need to learn a lot about the RFT syntax to use it. Just edit an RTF Blob in eDeveloper, then use `BLB2FILE()` to save the blob in a text file. You can cut and paste the results.*

## How do I Retrieve Data From a Multiple Selection List Box?



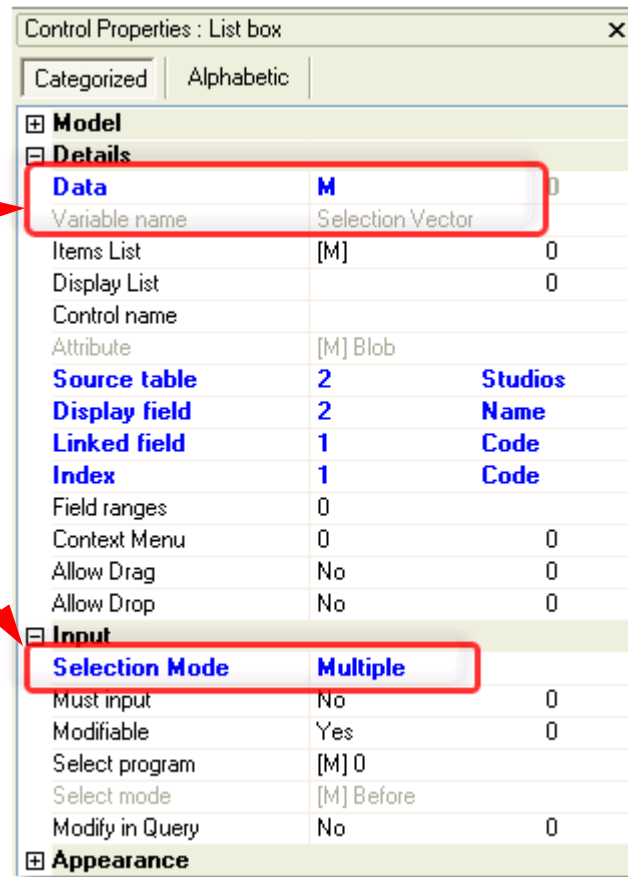
You can allow the user to select multiple items at one time from a selection list box. A multiple selection list is set up in the same way as a single selection list box, except in the Control Properties:

- *Selection Mode* = Multiple
- The Data property points to a vector.

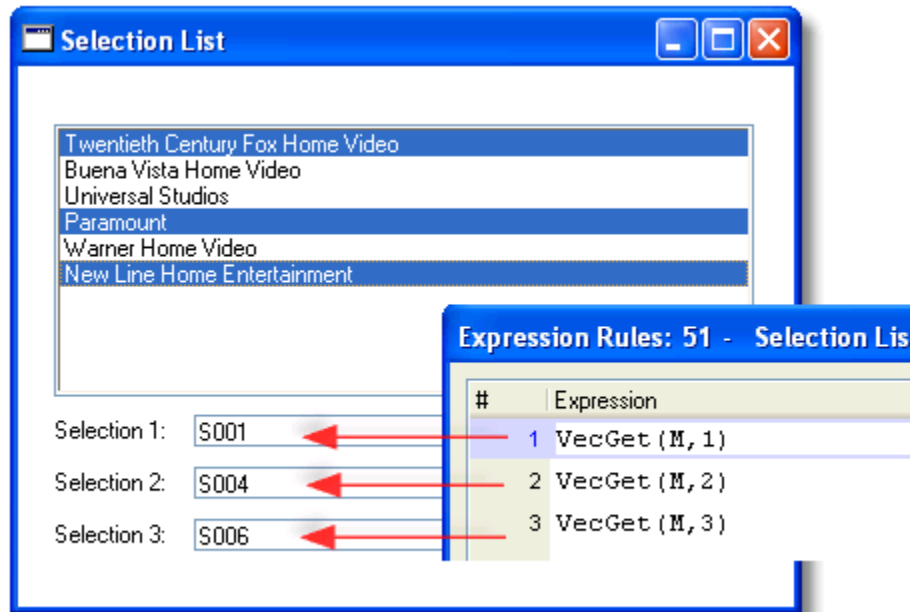
Let's look at an example.

## Using a Multiple Selection List box

1. Use a vector for the Data property.  
In our example, we used a simple alpha vector of text fields.
2. Set *Selection Mode* control property to *Multiple*.



Now, you can access the items in the vector, when they are selected, by using the **VecGet()** function.



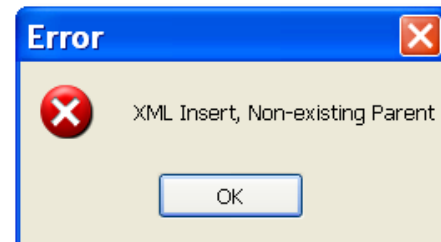


## Chapter 14: XML

---

### How do I Create an XML Doc from Scratch?

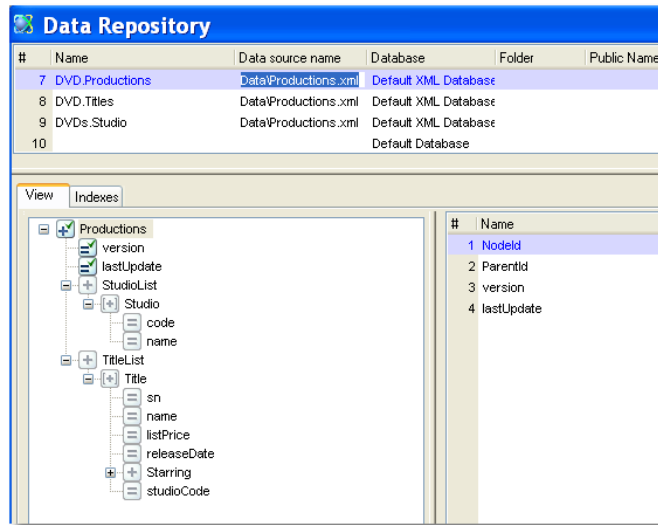
For the most part, you can handle an XML document that was defined in an XML view as you would any other data source. However, with an XML documents, all data is hierarchical, even when you might expect it is “flat”. And every XML file has to be initialized at the root record, or you will get a message such as the one shown here.



## Initializing an XML Document

In this example, for instance, there are three views into the Productions.xml file. All point to the same XML document, but a different part of the hierarchy is used in each view. The part that is used in the current view has green checkmarks on the fields involved, which are also shown on the right.

Now, if you want to write some titles to this XML file, you will have to create a root record in the Productions data source first. In practical terms, that means writing one record to data source #7 before adding any Titles to data source #8. You can use a Link Write or a small batch task to do create the root if the XML file does not already exist.



You can experiment with this by viewing the data in the Data Repository and creating records there (**Ctrl+G**). If you create a record at your root level (#7) then you can create records at the lower levels (#8 and #9) and they will be inserted correctly. But the reverse does not work.

## How do I Find an XML Schema?

In order to create an XML doc, you first need an XML *schema*. A schema is something like a database definition; it defines the format of each item in the database, whether it is required, unique, and has sub-items. If you have not dealt with XML before, it helps to learn something about it before you start.

If you are being asked to read an XML document, the schema will already exist, and you will just use it. If you are creating an XML document to send someone else, then you can either design the schema manually or use an external product, such as XML Spy. The schema will be in a text file that ends in *.xsd*.

In either case, once the schema exists, it is a simple matter to create the XML definitions in eDeveloper.

The schema describes complex data relationships. In essence, it can describe an entire database, consisting of multiple separate and nested tables. In order to use these as data sources, you need to select parts of the schema and describe them as individual flat data sources, called *XML Views*.

### The XML Schema

<b>Productions</b>	<b>Complex type</b>	
<b>StudioList</b>	<b>Complex type</b>	
<b>Studio</b>	<b>Complex type</b>	
Name	Simple Type	xs:string
Code	Simple Type	StudioCodeType
<b>TitleList</b>	<b>Complex type</b>	
<b>Title</b>	<b>Complex type</b>	
name	Simple Type	xs:string
listPrice	Simple Type	xs:float
releaseDate	Simple Type	xs:date
<b>Starring</b>	<b>Complex type</b>	
starName	Simple Type	xs:string
studioCode	Simple Type	StudioCodeType
sn	Simple Type	xs:string
version	Simple Type	xs:int
lastUpdate	Simple Type	xs:date
StudioCodeType	Simple Type	

This is a representation of the schema we are using. The schema consists of nested data. Some of the items are *complex types*, others are *simple types*. The *complex types* correspond roughly to rows in a table, or records in an ISAM file. The *simple types* correspond to columns in a table, or fields in an ISAM file.

So, you will create individual data source definitions for the complex types. In our example we will create one for the complex type “Studio”.

The simple types will usually contain a *type=* definition which will describe the data, as string, float, integer, or as a type in another schema element (as the StudioCodeType). eDeveloper will use these to create the default data types for each item in the data source.

**See also:** Chapter 14, “How do I Handle an XML Document with No Schema?” on page 369.

## How do I Create an XML View?

The XML view is the key to working with XML in eDeveloper. Once the XML view is created, you can work with the XML document much as you would any other Data Source in eDeveloper.

**Prerequisite:** You need to have:

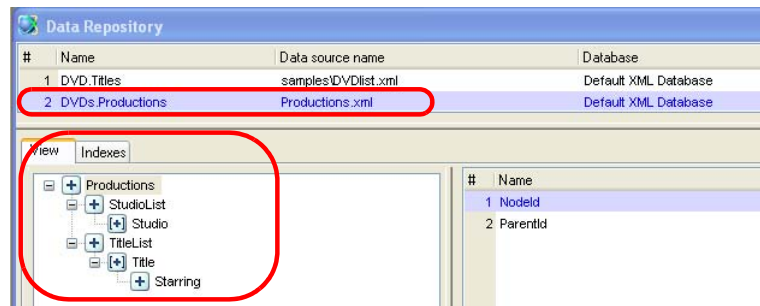
- A database type defined with Data Source type *XML File*.
- An XML Schema (See Chapter 14, “How do I Find an XML Schema?” on page 343).

### Creating an XML View

1. Open up a line in the Data Source Repository (**F4**, or **Edit->Create Line**).
2. Set the Database column to a database which has a Data Source type of *XML File*.

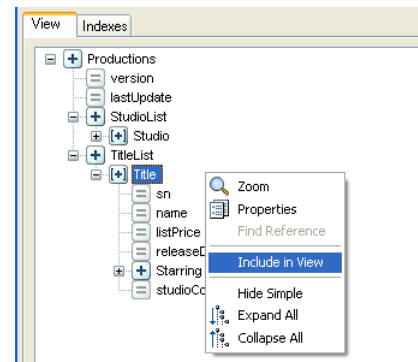
3. Get the schema information by selecting **Options->Get Definition (F9)**. A file selection dialog will open, which allows you to select the schema *.xsd* file

When you select the *.xsd* file, the *Name*, and *Data source name* fields will default to the first element in the schema, which in this example is “Productions”.

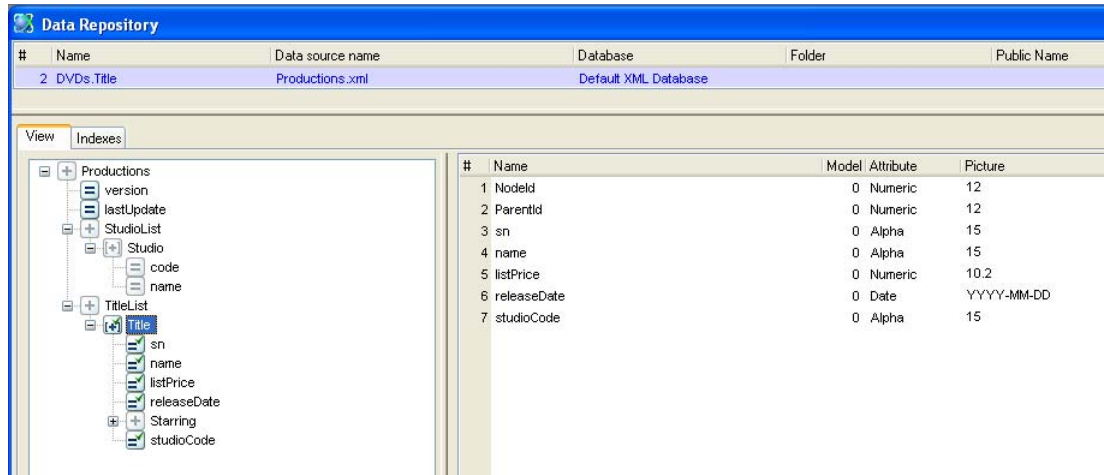


4. The *View* tab will be filled in automatically from the information in the *.xsd*.

You might see only the complex elements (The ones with the ‘+’ icon). To show the simple elements also, select **Right-Click->Show Simple**. In the picture to the right, we are showing the simple elements (and the menu changes to **Hide Simple**)



5. Since we want to create a view into the Title element, we move to the Title element, and select **Right-click->Include in View**.



Now, you have a sub-element that is linked to the root element by the NodeId and ParentId.

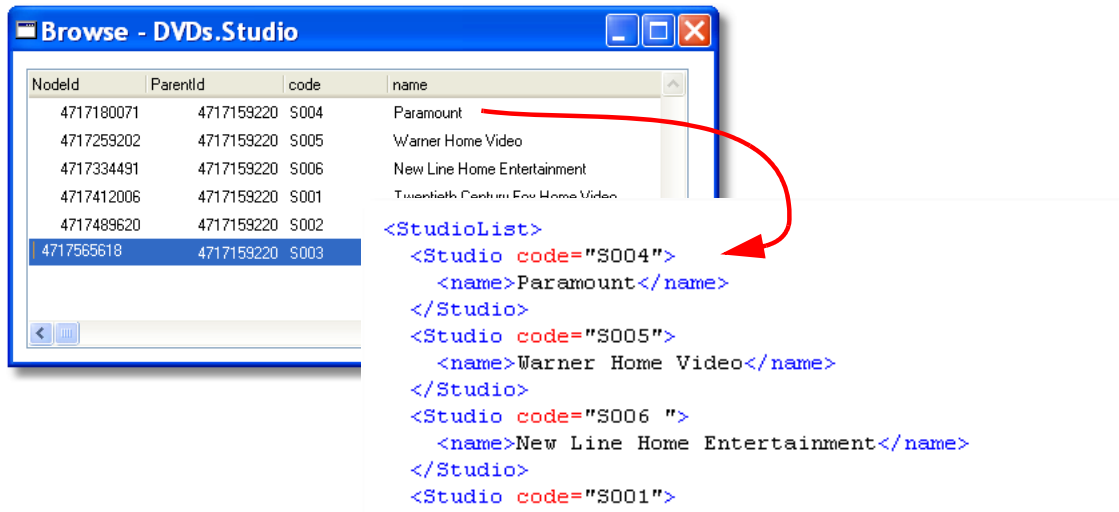
6. Change the *Data Source Name* to reflect the name of the actual XML document that will hold the data. It's a good idea to use logical names or relative paths rather than hard-coding the path name.
7. Change the *Name* to anything you like, to reflect what you want to call this XML view.

Now you can access the TitleList much as you would any other data source.

**Note:** You will need to create one XML view for each repeating element in the XML document (see Chapter 14, "How do I Access a Certain Compound in an XML File?" on page 348). Also, you need to initialize the root element before you can write to the XML document (see Chapter 14, "How do I Create an XML Doc from Scratch?" on page 341).

## How do I Update NodeId and ParentId?

XML



The screenshot shows a window titled "Browse - DVDs.Studio" with a table of DVD studios. The table has columns: NodeId, ParentId, code, and name. The data is as follows:

NodeId	ParentId	code	name
4717180071	4717159220	S004	Paramount
4717259202	4717159220	S005	Warner Home Video
4717334491	4717159220	S006	New Line Home Entertainment
4717412006	4717159220	S001	Twentieth Century Fox Home Video
4717489620	4717159220	S002	
4717565618	4717159220	S003	

Below the table, the XML code for the studios is displayed. A red arrow points from the "Paramount" row in the table to the corresponding XML code:

```
<StudioList>
  <Studio code="S004">
    <name>Paramount</name>
  </Studio>
  <Studio code="S005">
    <name>Warner Home Video</name>
  </Studio>
  <Studio code="S006 ">
    <name>New Line Home Entertainment</name>
  </Studio>
  <Studio code="S001">
```

You don't. NodeId and ParentId are used internally by eDeveloper, and if you do in fact update them in your program, the updates are ignored.

Each record in a repeating element will have a unique NodeId when the XML document is opened, although, as you can see in this example, it is not explicitly stored in the actual document. The ParentId refers to the parent element in the XML document. In this instance, that is the root node.

However, you don't have to know any of this to write your XML programs. You can ignore the NodeId and ParentId fields.

## How do I Access a Certain Compound in an XML File?

When you are working with an XML document, you are by definition working with a hierarchical database. Typically an XML document will have several levels. Even though these are stored together, you will work with them as though they were hierarchical data within SQL or ISAM tables.

### Repeating elements

The screenshot shows the 'Data Repository' window. At the top is a table with columns: #, Name, Data source name, Database, Folder, and Public Name. The table contains four rows, with the fourth row, '10 DVD.Starring', highlighted in blue and circled in red. Below this table are two tabs: 'View' and 'Indexes'. The 'View' tab is active, showing a hierarchical tree on the left and a table on the right. The tree on the left shows a hierarchy starting with 'Productions', which has children 'version', 'lastUpdate', 'StudioList', and 'TitleList'. 'StudioList' has a child 'Studio', which has children 'code' and 'name'. 'TitleList' has a child 'Title', which has children 'sn', 'name', 'listPrice', 'releaseDate', 'Starring', and 'studioCode'. The 'Starring' node is circled in red, and its child 'starName' is also circled in red. The table on the right has columns: #, Name, Model, Attribute, and Picture. It contains three rows, with the third row, '3 starName', highlighted in blue and circled in red.

#	Name	Data source name	Database	Folder	Public Name
7	DVD.Productions	Data\Productions.xml	Default XML Database		
8	DVD.Titles	Data\Productions.xml	Default XML Database		
9	DVDs.Studio	Data\Productions.xml	Default XML Database		
10	DVD.Starring	Data\Productions.xml	Default XML Database		

#	Name	Model	Attribute	Picture
1	NodeId	0	Numeric	12
2	ParentId	0	Numeric	12
3	starName	0	Alpha	15

In our example, the “Productions” xml document has several repeating elements:

- The root level: Productions, which will have exactly one record.
- Under Productions, the StudioList, which can have any number of elements.
- Under Productions, the TitleList, which can have any number of elements.
- Under TitleList, the Starring list, which can have any number of elements.

You will need one XML view for each level. After that, eDeveloper will do most of the work to keep the heirarchy correct when records are written.

### Selecting one compound

1. Select the node that is one level above the simple elements you want to include.



2. From the right-click menu, select *Include in View*.
3. For “child” records, you should also include a linking field. See Chapter 14, “How do I Modify an Existing XML Document?” on page 350.

Now, the simple elements will appear as fields in the XML view, and you can access them from an eDeveloper program. See Chapter 14, “How do I Create an XML View?” on page 345 for more details.

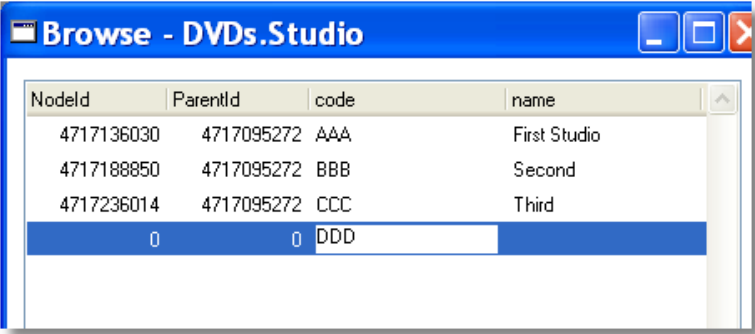
**Note:** You can tell the simple elements from the complex elements because the simple elements have an “=” icon in front of them, while the complex elements have a “+”. If the simple elements are not showing on the view, select **Show Simple** from the right-click menu.

**See also:** Chapter 14, “How do I Modify an Existing XML Document?” on page 350.

## How do I Modify an Existing XML Document?

Once an XML document has been created, and the XML views are set up, you can access it much as you would ISAM or SQL tables. Most of the underlying record handling is handled automatically.

### Accessing a parent record

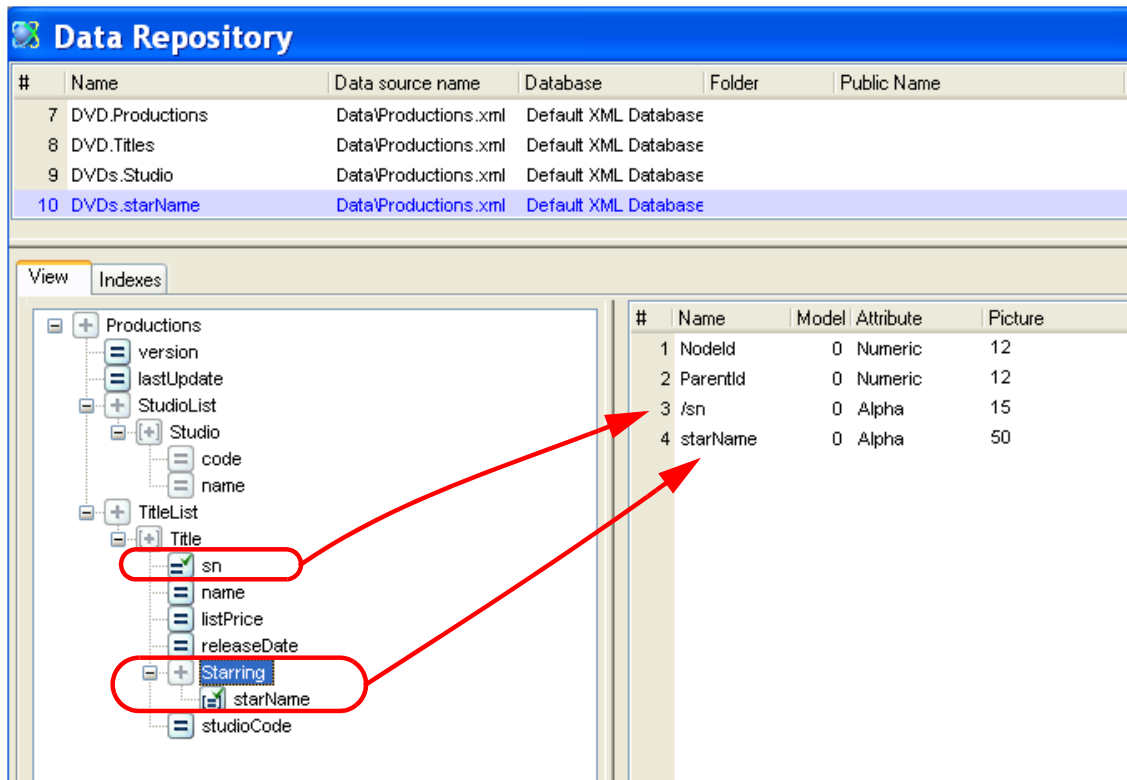


NodeId	ParentId	code	name
4717136030	4717095272	AAA	First Studio
4717188850	4717095272	BBB	Second
4717236014	4717095272	CCC	Third
0	0	DDD	

You can work with parent records just as you would any other table. The usual Create, Modify, and Delete modes work as they would with any data source. In this example, we just used **Ctrl+G** to create a Browse program and added a few test records. We don't do anything with the NodeId and ParentId fields (See Chapter 14, "How do I Update NodeId and ParentId?" on page 347).

Note that these "top level" records, however, are not the root record for an XML document. In our example, we are accessing the Studio List, which in an ISAM or SQL system would not be considered a child record, but in our XML document, it is just one element of the Productions XML document. The root record, "Productions" must be written before we can add record. See Chapter 14, "How do I Create an XML Doc from Scratch?" on page 341.

## Accessing a child record



Now, when you are creating child records, how do you link them to the parent? There are two steps to this.

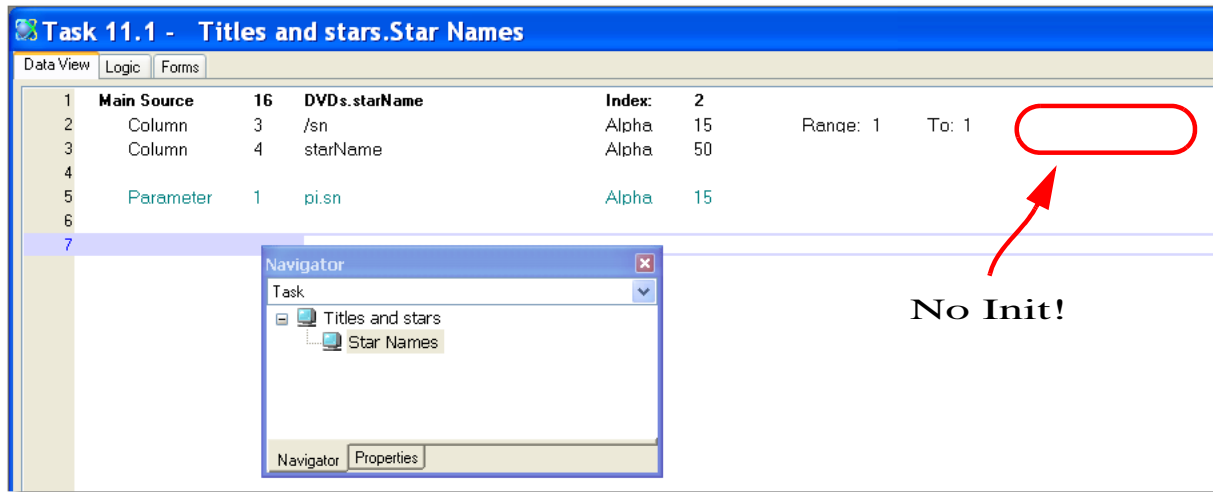
First, when you are creating the XML View, include the “linking field” in the view for the child record..

You select the linking field you want to use by:

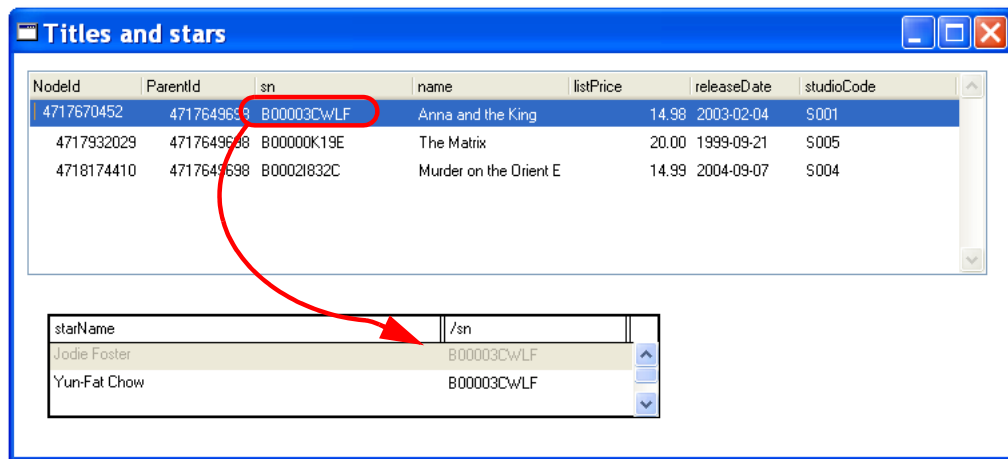
1. Positioning the cursor on the element in the parent level you want to use for linking.
2. From the right-click menu, select *Include in View*.

Now the linking field will be included in the view. In this example, the “sn” (serial number) field of the the title is selected in the child record. eDeveloper adds a forward slash in front of the name to indicate that it is a linking field.

Now, when the record is used in a subtask, the linking field will be automatically updated by eDeveloper.



In this example, we use a parameter for a range, to show only the stars that relate to this movie, which is how most child task views work. But, there *is no init* on the /sn field. Nonetheless, when the program runs, the /sn field is initialized to the parent sn that was used for the range.



## How do I Determine the eDeveloper Datatypes Corresponding to XML Datatypes?

XML

View		Indexes			
Productions		#	Name	Model	Attribute
version		1	Nodeid	0	Numeric
lastUpdate		2	Parentid	0	Numeric
StudioList		3	sn	0	Alpha
TitleList		4	name	0	Alpha
Title		5	listPrice	0	Numeric
sn		6	releaseDate	0	Date
name		7	studioCode	0	Alpha
listPrice					
releaseDate					
Starring					
studioCode					

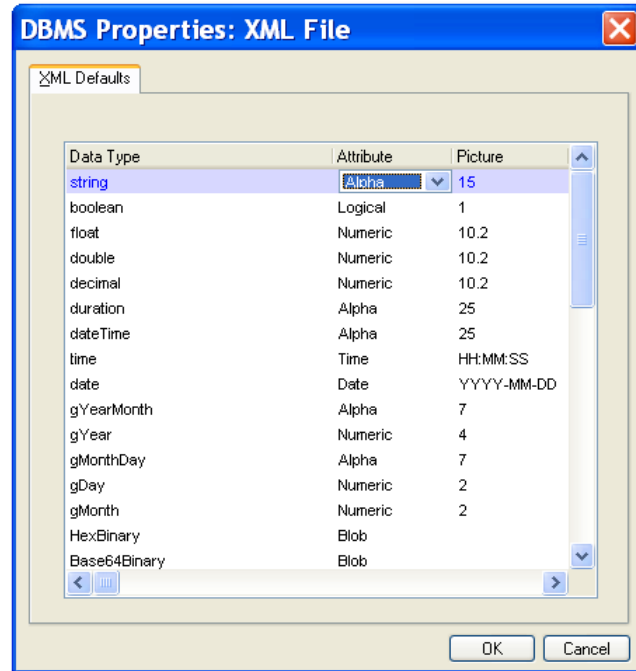
```

<xs:element name="Title" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="listPrice" type="xs:float"/>
      <xs:element name="releaseDate" type="xs:date" minOccurs="0"/>
      <xs:element name="Starring">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="starName" type="xs:string" maxOccurs="unbound
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="studioCode" type="StudioCodeType"/>

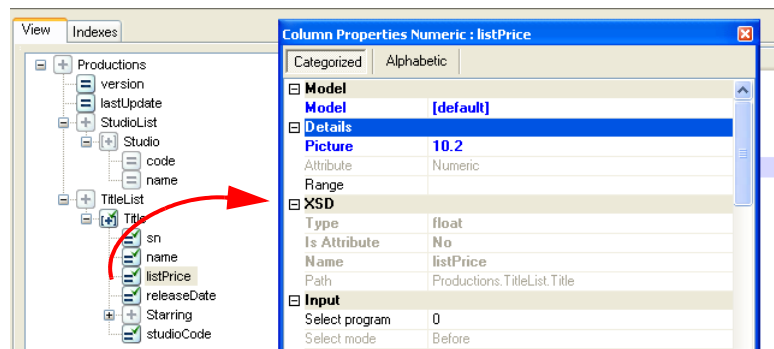
```

When you create an XML View, eDeveloper will look at the XML schema and convert the data according to the mapping in **Settings->DBMS->Properties** (XML Defaults). So for instance, in this example, the XML data type “float” will translate into the eDeveloper picture of “10.2”.

Once you have a real XML document from the other party, you can change the XML View to reflect the real data by simply changing the XML View field definitions. For instance, we could change the name from 15 characters to 40, if we saw that the actual XML document has longer title names.



## Viewing the schema setting



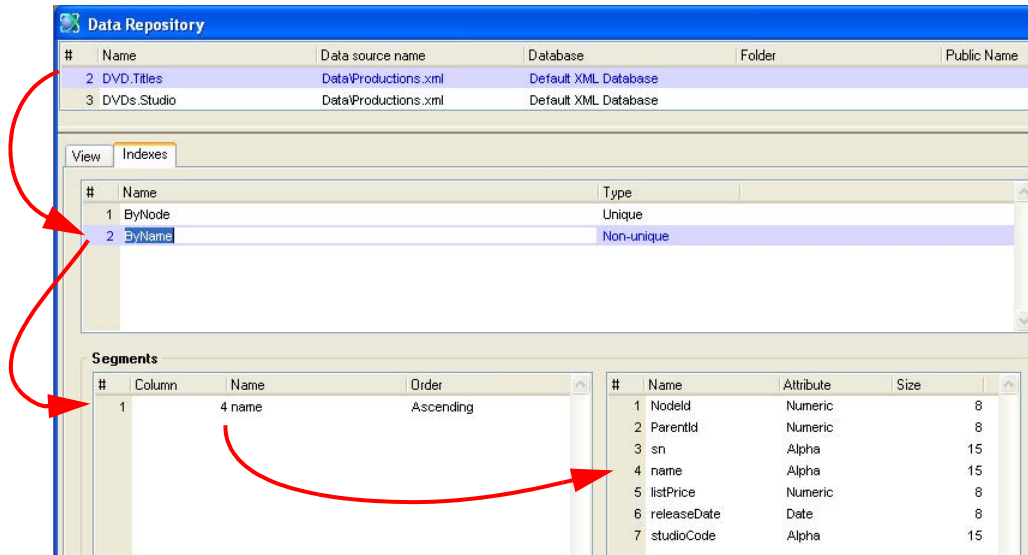
When an element is included in the XML view, you don't have to view the schema directly to see how it was defined in the schema. The XSD section of the property sheet shows the schema settings.

## How do I Retrieve Data from an XML Doc in a Preferred Order?

It is important to remember that the XML view simply creates a temporary table in which to store the XML data. Since it is a temporary table, you can add indexes if you wish, and use those indexes as you would for any data source.

XML

### Creating an alternate index for an XML view



1. In the Data Repository, select the XML view you want to work with.
2. Click on the *Indexes* tab.
3. Press **F4** to create a new index line.
4. Type in a name for your index. Tab to the right.
5. In the Type column, select Non-unique if it is not a unique index.
6. Click on the *Segments* area (lower left part of the screen). Press **F4** to open up a line.
7. Press **F5** to **zoom** to the list of fields. Select the field you want to sort on.
8. Repeat steps 6 and 7 for more index segments, if desired.

Now, when you use your new index in a program, the data will be presented in the order according to that index.

## How do I Handle Multi-Occurrence Elements in an XML Doc?

NodeId	ParentId	sn	name	listPrice	releaseDate	studioCode
4540617612	4540596860	B00003CWLF	Anna and the King	14.98	2003-02-04	S001
4540879290	4540596860	B00000K19E	The Matrix	20.00	1999-09-21	S005
4541121671	4540596860	B00021832C	Murder on the Orient E	14.99	2004-09-07	S004

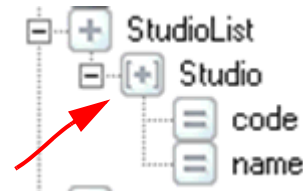
  

starName	/sn
Nodie Foster	B00003CWLF
Yun-Fat Chow	B00003CWLF

Multi-Occurrence elements in an XML document are handled the same way that “child” records are handled in ISAM or SQL data sources. The child records are displayed in a subtask, in a child form or sub-form, and are displayed using a range so only the records that relate to the parent are shown.

Below are details for how to create programs using recurring XML elements.

**Hint:** You can tell the repeatable elements in a schema from within eDeveloper by looking at the icon. Repeatable elements have square brackets around them. You can tell the repeatable elements in the schema because they have `maxOccurs="unbounded"` or `maxOccurs=>1`.



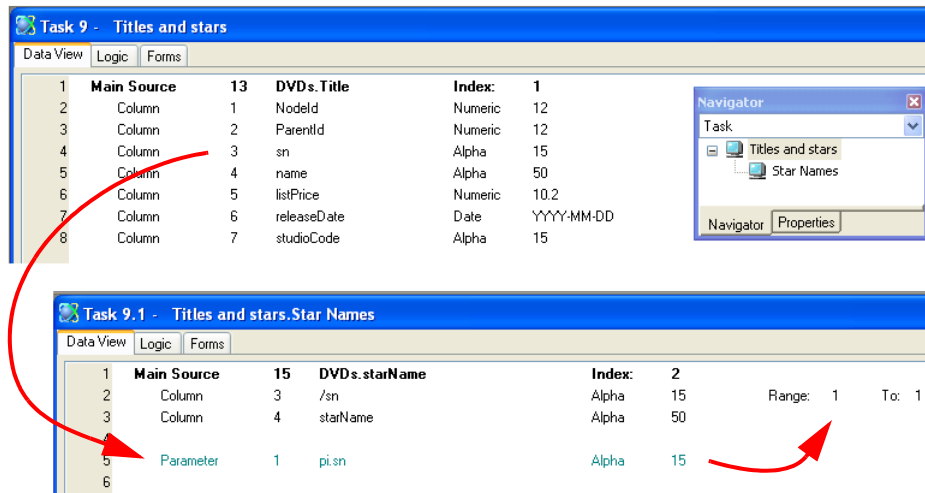
### Displaying repeatable elements

#	Name	Model	Attribute	Picture
1	NodeId	0	Numeric	12
2	ParentId	0	Numeric	12
3	/sn	0	Alpha	15
4	starName	0	Alpha	50

1. First, you need to have an XML view for the repeatable element. In this example, we include the “sn” field, which is the linking field, and the repeatable element “starName”.



2. On this XML view, we also create an index so we can access the elements by “sn”.



3. We create task that displays the parent element, **Title**. In this task we select the **/sn** field. We pass the **/sn** field as a parameter to the subtask.
4. We also create a subtask, which accesses the **starName** XML view. The index used here is the **/sn** field, and we use that same field as the parameter, to range the view.
5. A *Subform* is used on the parent form to display the subtask (See Chapter 8, “Subforms” on page 197).

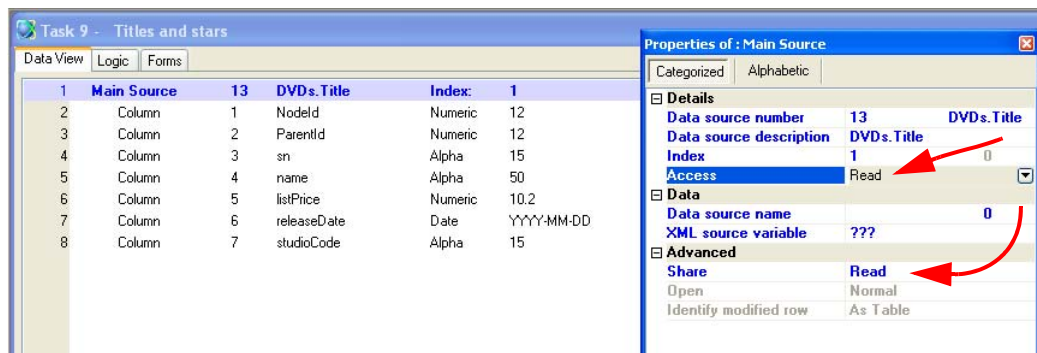
Now the repeating elements will display in the subform.

## How do I Allow Different Users Access to the Same XML Document?

If more than one user needs to access an XML document, you need to make sure that the document is opened in read mode.

Because XML documents are not managed by a DBMS, you cannot allow more than one user to update the XML document at one time. If you need to produce an XML document that will be updated by multiple users concurrently, you should store the data as SQL or ISAM data source, and produce the XML document when needed from the stored data.

### Setting the access mode for an XML document



1. Go to *Main Source* or *Link* for the data source you want to work with.
2. Press **Alt+Enter** to go to the *Properties* pane.
3. Set the *Access* property to *Read*, so this task will only access the XML document in read mode. Now, two different users use this task at the same time.
4. Set the *Share* property to either *Read* or *Write*. Set it to *Write* if you want other tasks (or external products) to be able to update the XML document while this task is reading it. Do not set it to *None*, however, because then only one user can view the data at a time.

## How do I Create Different XML Docs Based on the Same Schema?

XML

Data Repository			
#	Name	Data source name	Database
1	DVD.Productions	Data\Productions.xml	Default XML Database
2	DVD.Titles	Data\Productions.xml	Default XML Database
3	DVDs.Studio	Data\Productions.xml	Default XML Database
4			Default Database
5	DVD.Productions	Data\Productions2.xml	Default XML Database
6	DVD.Titles	Data\Productions2.xml	Default XML Database
7	DVDs.Studio	Data\Productions2.xml	Default XML Database

When you create an XML view, you create it according to a schema you set when do **Options->Get Definition (F9)**. However, the data itself is stored in a file whose name is determined by the name entered in the *Data source name* column.

**Note:** However, you can override the XML document name within the programs that access the document. This is done in the same way you override the data source name for an SQL or ISAM file, by creating an expression for the Data source name property of the source. Note that while in our example we hardcoded the name for clarity, it would be better to use logical names or variables that are set at runtime.

## How do I Access an XML Document Stored in an eDeveloper Data Variable?

You can use the XML view to access an XML document directly from a BLOB, without converting that BLOB to a file first.

### Using a BLOB as a data source

**Task 8 - View Studio**

Data View | Logic | Forms

Main Source	Index	Data Type	Index
11	1	DataType2.Studio	1
2	1	NodelId	Numeric 12
3	2	ParentId	Numeric 12
4	3	code	Alpha 15
5	5	RTF	Blob

**Properties of : Main Source**

Categorized | Alphabetic

**Details**

Data source number	11	DataType2.Studio
Data source description	DataType2.Studio	
Index	1	
Access	Write	

**Data**

Data source name

**XML source variable**

**Advanced**

Share | Read

**Variable List**

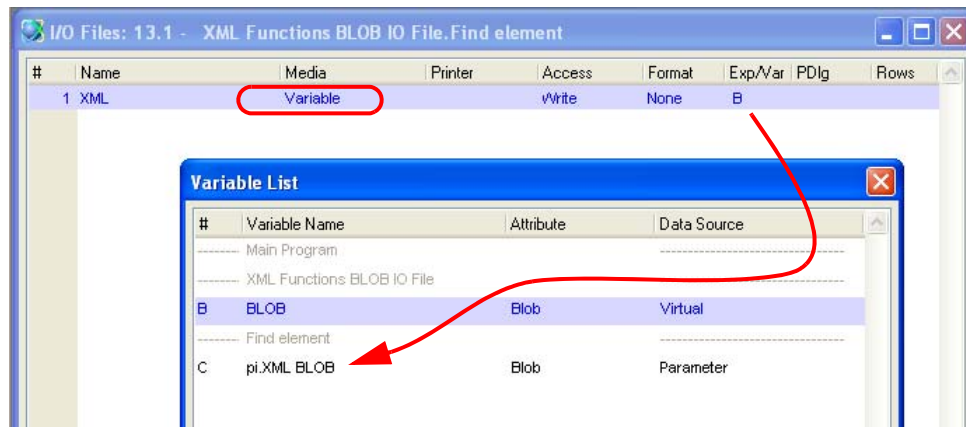
#	Variable Name	Attribute	Data Source
	Main Program		
	View Studio		
E	RTF	Blob	DataType2.Studio
F	v.StudioXMLBlob	Blob	Virtual

1. Specify your Main Source as your XML view.
2. Go to the Properties of the Main Source (**Alt + Enter**).
3. Go to the XML Source variable, zoom (**F5**) to the Variable list.
4. Select the BLOB that contains your XML document.

Now you can use the XML document in the same way you would as if it were in a file.

**See also:** Chapter 14, “How do I Create Different XML Docs Based on the Same Schema?” on page 359.

## Using a BLOB as in IO file



If you are using the XML functions rather than the XML data views, you can use a BLOB in the Exp/Var column of the IO File name.

1. Press **Ctrl+I** to access the IO Files.
2. Press **F4** to open up a line.
3. Select *Variable* in the *Media* column.
4. In the *Exp/Var* column, select the BLOB you want to access.

Now, you can use this IO File in the XML functions. You will refer to it by the generation (0 for the current task, 1 for the parent, etc.) and the sequence number of the file (here, it is 1, but you may have more IO files in your task).

**See also:** Chapter 14, “How do I Handle an XML Document with No Schema?” on page 369.

## How do I Validate an XML Document?

It is recommended that you validate an XML document before using it. If the XML is not properly formatted, or does not match the schema definitions, you may get erratic results.

eDeveloper has a built-in function to validate the XML document, **XMLValidate()**. This function will compare the XML document with its schema, and prepare a list of errors.

The screenshot shows the eDeveloper IDE with a task named "Task 12 - XMLValidate()". The task is configured with the following logic:

Line	Event	Validate	Variable	Value	With	Scope
1	Event	Validate	Variable	F	ErrorCount	Task
2	Update	Variable	F	ErrorCount	With: 1	0
3						
4						
5						
6						
7	Update	Variable	G	Errors found?	With: 4	XMLValidate (File2B1b(XM Cnd. Yes
8	Update	Variable	E	XMLErrorVector	With: 5	XMLValidationError()
9	Update	Variable	F	ErrorCount	With: 6	VecSize(XMLErrorVector)
10						
11	Block	While	7	{LoopCounter ()<=ErrorCount		
12				Display error to the user		
13	Verify	Warning	8	VecGet(XMLErrorVector,LoopC		Display in Box
14	Block	End		}		
15						
16						

The 'Expression Rules' dialog box is open, showing the following expressions:

#	Expression
3	'C:\eDeveloper10_Project\XML_Demo\samples\.
4	XMLValidate (File2B1b(B),C)
5	XMLValidationError()

### Using XMLValidate()

The basic steps to using XMLValidate are:

1. Call **XMLValidate()**, storing the result in a logical. This logical will be FALSE if there are errors; however, if there are only warnings or no schema was specified, it can be TRUE and there will still be messages produced.
2. Call **XMLValidationError()** to store the errors and warnings in a vector.
3. Use **VecSize()** to find how many errors/warnings are in the vector
4. Use a BlockWhile loop to read the vector. Here you can give the messages to the user one at a time in a warning box, as in our example, or store them in a table to view all at once.

You can also create a global function to handle validation errors, as shown in Chapter 14, "How do I Retrieve Validation Errors of an XML Document?" on page 364.

### XMLValidate() syntax

The syntax of **XMLValidate()** is:

**XMLValidate**(*XMLBLOB*, [*XSDSchema*])

- *XMLBLOB* is a BLOB containing the XML.
- *XSDSchema* is the location of the schema, in URL format

**Returns:** It returns 'FALSE'LOG if errors were found. It returns 'TRUE'LOG if there were no errors, or if only warnings were found.

**See also:** The eDeveloper help for XMLValidate.  
Chapter 14, "How do I Retrieve Validation Errors of an XML Document?" on page 364.

XML

# How do I Retrieve Validation Errors of an XML Document?

After you do an `XMLValidate()`, the errors, if any, can be retrieved with the `XMLValidationError()` function. This function returns a vector of text messages (Unicode). Using `VecSize()` on the vector gives you the number of error messages, which you can then save to a table, print, or display.

Data View	Logic	Forms
1	Event	Validate
2	Evaluate	Expression
3	Update	Variable
4	Update	Variable
5		
6		*** Errors found? ***
7	Block	If
8	Update	Variable
9	Update	Variable
10	Block	While
11	Verify	Warning
12	Call	SubTask
13	Block	End
14		
15		*** Possibly warnings found ***
16	Block	Else
17	Update	Variable
18	Update	Variable
19		
20	Block	While
21	Verify	Warning
22	Call	SubTask
23	Block	End
24	Block	End

**See also:** Chapter 14, “How do I Handle Errors Encountered During XML Access?” on page 365.



## How do I Handle Errors Encountered During XML Access?

Some of the XML functions give return codes which can be used to determine that an error occurred. For instance, **XMLSetEncoding()** returns a 0 if it worked properly, a negative value otherwise.

**Note:** Since the return code can be negative, be sure your variable allows negative values (that is, that the picture contains an “N”, “5N” for example).

### Creating a global error handler

All the error codes are negative integers. So, you can create a handler that will be triggered whenever your return code becomes negative, as shown here.



If you use a return code in the Main Program, you can capture the error globally for all your XML functions.

The same return code values are used for all the XML functions, so you can also create user-friendly error messages. The error codes are listed in the eDeveloper Help file for the function.

## How do I Determine the Encoding for an XML Document?

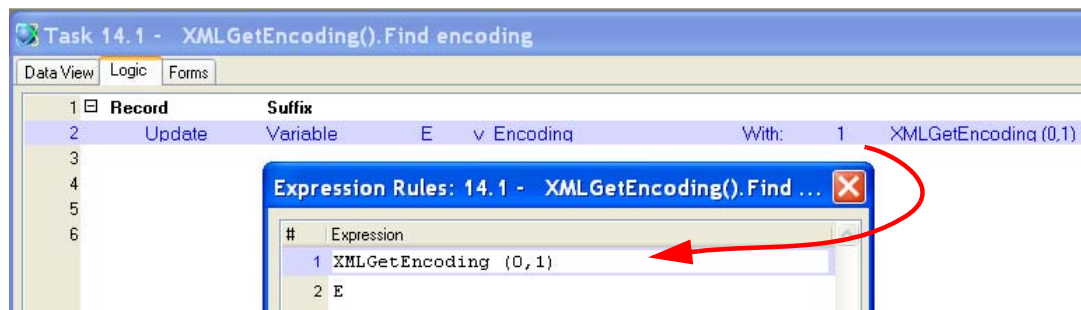
The encoding of an XML document can be given in the XML document header (the first line of the XML document), as shown below:

```
<xml version="1.0" encoding="iso-8859-1"?>
```

If no encoding is given, utf-8 is assumed. The encoding is used by the parser to determine how to parse the XML; some characters are legal under one encoding but are considered invalid in another.

You don't need to parse the XML from within eDeveloper to find the encoding, however. There is a function, **XMLGetEncoding()**, that will fetch it for you.

### Using XMLGetEncoding()



The syntax of **XMLGetEncoding()** is:

**XMLGetEncoding(*generation*, *I/O entry*)**

- *generation* is the task generation, 0 for the current task, 1 for the parent, and so on.
- *I/O entry* is the sequence number of the IO file that has the XML document.

It returns a text field, which is the value of the "encoding" attribute on the XML document.

**See also:** The eDeveloper Help utility for XMLGetEncoding and XMLSetEncoding.

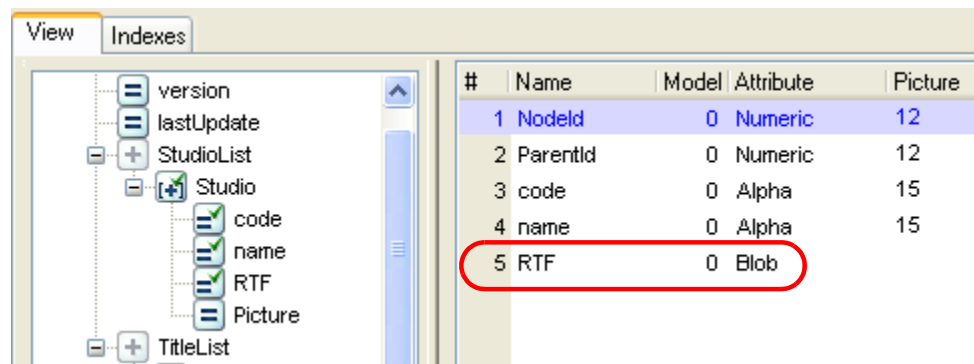
## How do I Handle XML Data Which is Base64 Encoded?

If you are using the XML Views, then eDeveloper will automatically convert Base64 data to/from the binary values to text, when the field is defined in the schema as “base64Binary”.

For example, here we added two base64 elements to our sample .xsd:

```
<xs:element name="Studio" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="RTF" type="xs:base64Binary"/>
    /xs:sequence>
    <xs:attribute name="code" type="StudioCodeType" use="required"/>
  </xs:element>
```

When we use this to create an XML View, we get:



In this example, we are using the BLOB to store RTF text, but it could contain an image or anything else.

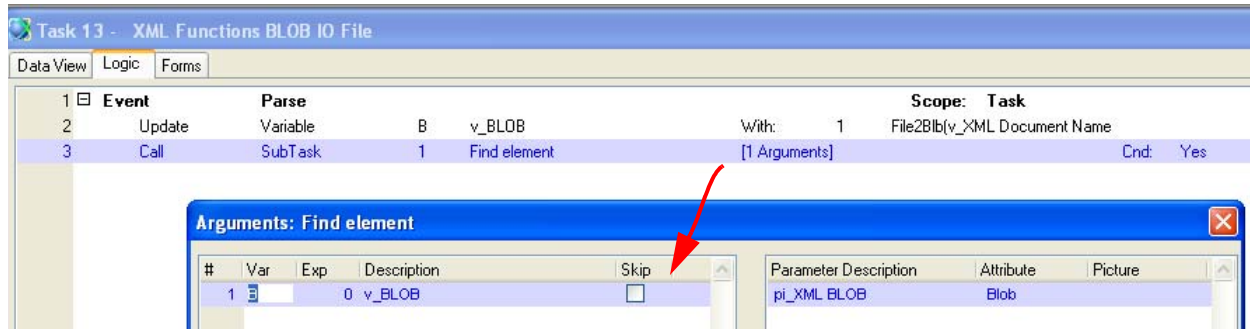
When we write to the RTF field and then store the XML, it looks something like this:

```
<Studio code="S001">
  <name>Universal Studios</name>
  <RTF>c2QgIGFzZGYgYWRmIGQKDFkZiBhc2RmCg1hZHNmYXNkZmFkc2YKDFkZCBmNZg==</RTF>
</Studio>
```

eDeveloper did all the work.

**Note:** Only one BLOB is allowed per XML View.

## How do I Pass an XML Document as a Parameter?



If you want to pass an XML document as a parameter, the easiest way to do it is to store the XML in a BLOB (if it isn't already) and then access that BLOB directly. See Chapter 14, "How do I Access an XML Document Stored in an eDeveloper Data Variable?" on page 360.

To convert a file to a BLOB, use the **File2BLB()** function. See the eDeveloper help files for the syntax on how to do this.

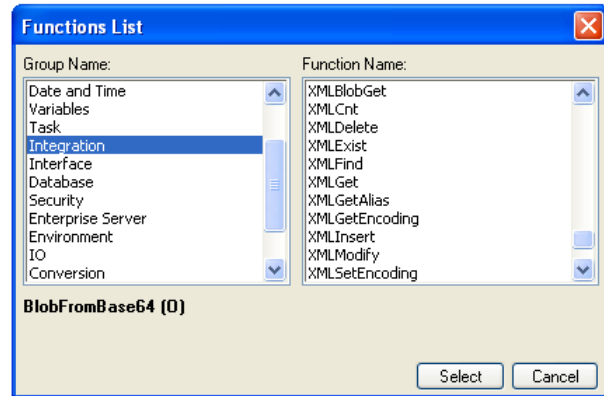
## How do I Handle an XML Document with No Schema?

In order to use the XML views in the Data repository, you must have a schema. If you have a document with no schema, you can create one with a 3rd party product, or write one manually.

Alternatively, you can use the built-in eDeveloper functions to access the document. These functions require you to know the internal format of the XML file rather than using a schema to do the work for you. Using these functions is explained in:

Chapter 14, “How do I Retrieve / Update /Insert Data According to a Certain Path in an XML Document?” on page 370

Chapter 14, “How do I Uniquely Identify a Data Element and Its Hierarchy Within an XML Document?” on page 372



## How do I Retrieve / Update /Insert Data According to a Certain Path in an XML Document?

If you are using the XML views to access an XML file, then you will work with the XML data source much as you would any other data source in an eDeveloper task. Using XML views is explained in Chapter 14, “How do I Create an XML Doc from Scratch?” on page 341

However, you can also use XML functions to manipulate an XML IO File. This is more work than using an XML view; the functions mainly exist for backward compatibility to eDeveloper version 9.4.

**Prerequisite:** If you are going to update or insert data you have to be sure the file is open in write (not append) mode. Also, be aware that the syntax is slightly different if the data is an attribute or not.

For more information about formatting element path and attribute name, see Chapter 14, “How do I Uniquely Identify a Data Element and Its Hierarchy Within an XML Document?” on page 372. Also, these are fairly complex functions; see the eDeveloper help for each function for more information.

### Retrieving XML Data

**XMLGet**(*generation*, *file*, *element path*, *attribute name*)

- *generation* is the task generation, 0 for the current task, 1 for the parent, and so on.
- *file* is the sequence number of the IO file that has the XML document.
- *element path* is a string that uniquely defines one element in the path, as described below
- *attribute name* is an attribute of the element, if any

**Returns:** If the function was successful, the function returns the data, as an alpha string. Otherwise, it returns an empty string.

### Updating XML Data

The **XMLModify()** function allows you to modify elements in an XML file. The syntax is:

**XMLModify**(*generation*, *I/O entry*, *element path*, *attribute*, *value* [, *auto convert*])

- *generation* is the task generation, 0 for the current task, 1 for the parent, and so on.
- *I/O entry* is the sequence number of the IO file that has the XML document.
- *element path* is a string that uniquely defines one element in the path, as described below.
- *attribute* is an attribute of the element, if any.
- *value* is a string containing the data you want to change the element or attribute to.
- *auto convert* is optional. If TRUE, then any characters that are invalid in XML are converted to placeholders.

**Returns:** Zero if the function was successful, an negative number otherwise. The error codes are listed in the eDeveloper help utility.

## Inserting XML Data

The **XMLInsert()** function allows you to add elements to an XML file. The syntax is:

**XMLInsert** (*generation*, *I/O entry*, *element path*, *attribute*, *value* [, *before/after flag*, *reference element*, *auto convert*])

- *generation* is the task generation, 0 for the current task, 1 for the parent, and so on.
- *I/O entry* is the sequence number of the IO file that has the XML document.
- *element path* is a string that uniquely defines one element in the path, as described below.
- *attribute* is an attribute of the element, if any.
- *value* is a string containing the data you want to add, or a Rich Edit BLOB.
- *before/after flag* is optional, It is used with the reference element, to determine the placement of the new element.
- *reference element* is optional. It is used with the before/after flag to determine the placement of the new element.
- *auto convert* is optional. If TRUE, then any characters that are invalid in XML are converted to placeholders.

**Returns:** Zero if the function was successful, an negative number otherwise. The error codes are listed in the eDeveloper help utility.

## Deleting XML Data

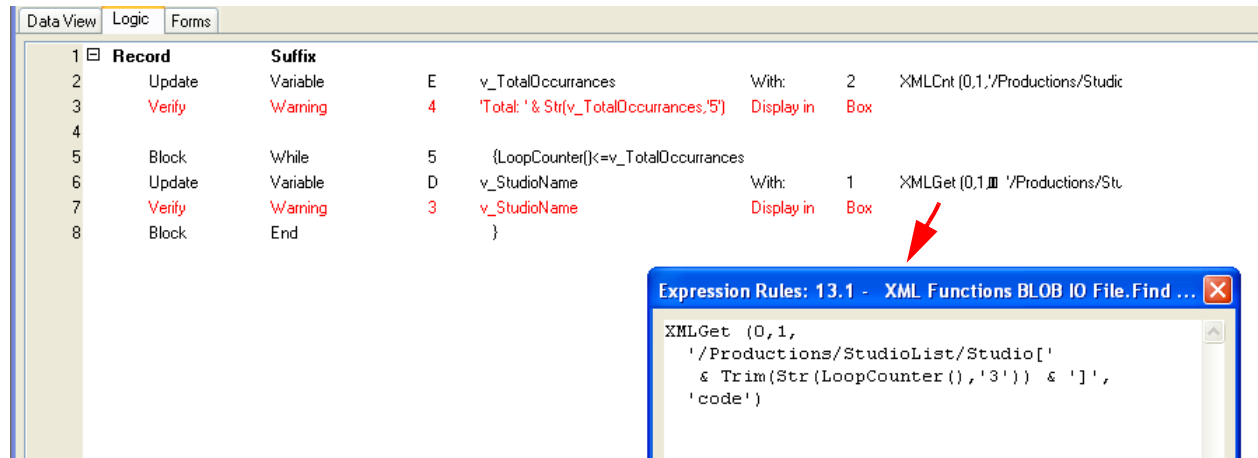
The **XMLDelete()** function allows you to delete elements from an XML file. The syntax is:

**XMLDelete** (*generation*, *I/O entry*, *element path*, *attribute*)

- *generation* is the task generation, 0 for the current task, 1 for the parent, and so on.
- *I/O entry* is the sequence number of the IO file that has the XML document.
- *element path* is a string that uniquely defines one element in the path, as described below.
- *attribute* is an attribute of the element, if any.

**Returns:** Zero if the function was successful, an negative number otherwise. The error codes are listed in the eDeveloper help utility.

## How do I Uniquely Identify a Data Element and Its Hierarchy Within an XML Document?



If you are using an XML View to access an XML data source, you can access the elements just as you would columns in any other data source. You can view the hierarchy easily in the Data Repository: see Chapter 14, “How do I Create an XML Doc from Scratch?” on page 341.

However, if you are using the XML functions, you need to specify the hierarchical path into the XML file.

For more information on these functions, see Chapter 14, “How do I Retrieve / Update / Insert Data According to a Certain Path in an XML Document?” on page 370.

Let’s take one the **XMLGet()** function as an example.

**XMLGet(generation, file, element path, attribute name)**

- **generation** is the task generation, 0 for the current task, 1 for the parent, and so on.
- **file** is the sequence number of the IO file that has the XML document.
- **element path** is a string that uniquely defines one element in the path, as described below
- **attribute name** is an attribute of the element, if any

Each element along the path is identified by its name, and an index (if it has multiple occurrences), separated by forward slashes. The last parameter is used for the attribute, if any.

Let’s take this XML snippet for an example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

```
<Productions lastUpdate="2006-01-01" version="1" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
  <StudioList>
    <Studio code="S004">
      <name>Paramount</name>
    </Studio>
    <Studio code="S005">
```



```
<name>Warner Home Video</name>
</Studio>
<Studio code="S006 ">
  <name>New Line Home Entertainment</name>
```

XML

So for instance, to access the 3rd occurrence of the Studio code *attribute*, you would enter

```
XMLGet (0,1,
        '/Productions/StudioList/Studio[3]','code')
```

Which would return 'S006'

If you wanted to get the studio name *element*, the syntax would be:

```
XMLGet (0,1,
        '/Productions/StudioList/Studio[3]/name', '')
```

Which would return 'New Line Home Entertainment'.

Usually, of course, the index would not be hardcoded, it would be a variable or some function such as **LoopCounter()**.

**See also:** Chapter 14, "How do I Retrieve / Update /Insert Data According to a Certain Path in an XML Document?" on page 370.

## How do I Handle Mixed-Content in an XML Document?

A mixed-content element is one that contains both elements and text. For instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<notice>
  To
  <name>Fred Flicker</name>
  Your DVDs are are overdue. They were due on
  <duedate>07/21/2003</duedate>
  Please bring them in ASAP.
  We really appreciate it. Thanks!
  <greeting>Sincerely,</greeting>
  <outlet>AAA Rentals</outlet>
</notice>
```

Here we have 4 text items (regular text) and 4 element items (in bold).

First, if you receive an actual XML document with multiple roots. For instance:

When using XML views, for handling mixed content we have two functions:  
DbXmlMixedGet and DbXmlMixedSet (explained with an example in the eDev help)

## Chapter 15: COM

---

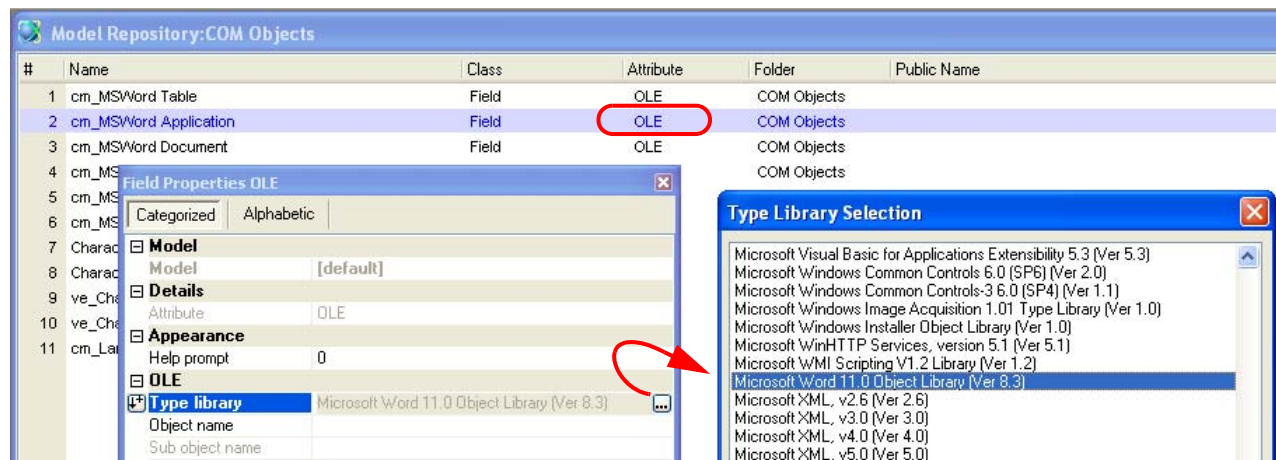
### How do I Define the COM Object That I Want to Use?

Before you can use a COM object, you have to declare it. A COM object is defined in eDeveloper like any other variable, in the Data section of your task.

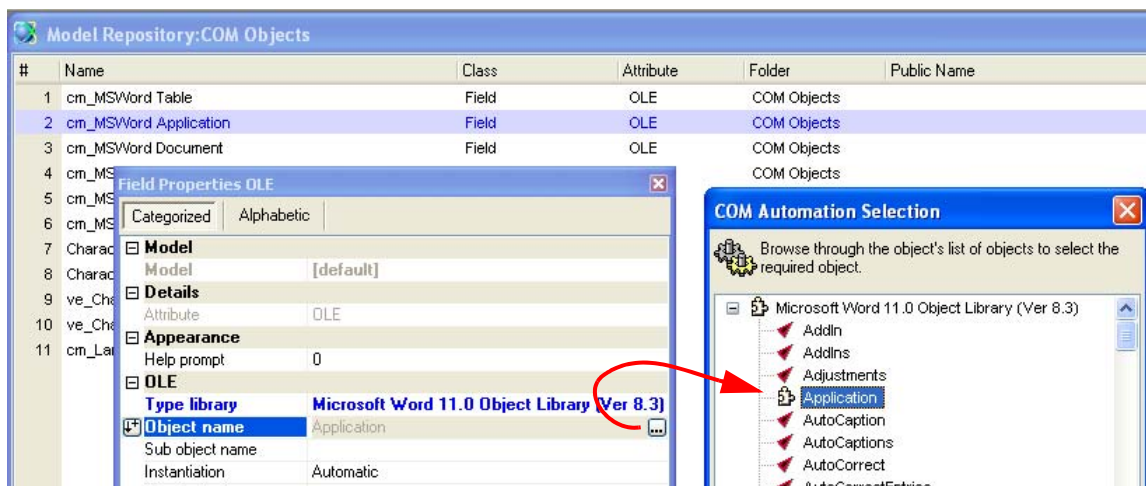
Defining a COM object is a little more work than say, defining a date variable or a text field. You have to select from a list of libraries, and within each library, you have to select from what can be a very large list of objects. Also, COM objects are often used to link to other products, such as Word or Excel, and you may have to modify the COM definition when new versions of those products are installed.

For these reasons, it is recommended that you declare your COM definitions as Models and reuse them as needed. See Chapter 15, “How do I Re-use COM Object Definitions?” on page 401.

## Defining a COM object



1. Open up a line in the *Model* repository (**F4** or **Edit->Create Line**).
2. Type anything you like in the *Name* column.
3. Leave the default *Class* (Field).
4. Set the *Attribute* to OLE (or ActiveX if it is a visible object).
5. From the *Type Library* property, **zoom** to select the library you want to use. The library list can be very large; all the libraries installed on your computer are listed. You can type the first letter to get to the section you want. Press **Enter** to select the library you want to use.



6. From the *Object name* property, zoom to select the object you want from this library.
7. Select the *Sub object* in the same way, if needed.

Now, you can use this model to declare COM objects in your tasks. Just select it as you would any other model when declaring a variable.

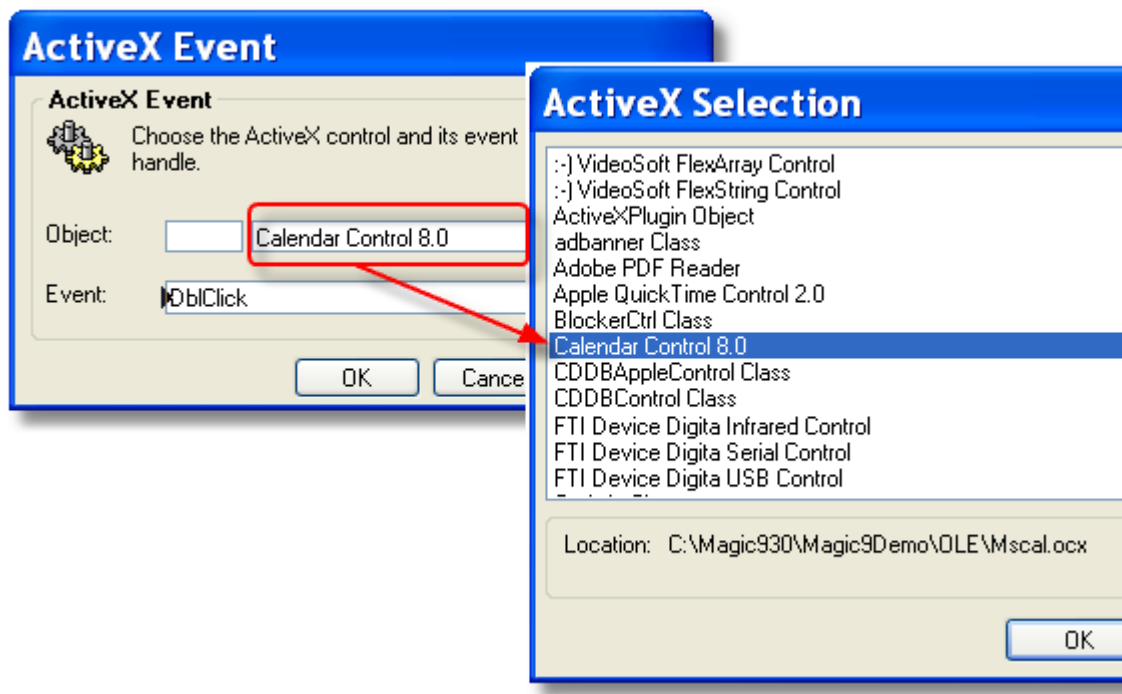
**Note:** A person might reasonably ask, seeing all the choices from within a COM library, how one knows which object to pick. Basically, a COM library is like a huge function list. You don't really know which functions you need unless you read the documentation that is produced by whoever wrote the library (the Microsoft developer documentation for Word, for instance). Nevertheless, eDeveloper makes it rather easy to just experiment with the objects and often you can just figure them out by working with them.

## How do I Set a Single Event Handler for Several COM Objects of the Same Type?

14		
15	Event	Calendar Control 8.0.DbClick
16		

If you have several COM objects of the same type, you can set one ActiveX event handler that will respond to any of those COM objects. For instance, in this example, we have an event handler that will respond to a double-click on any *Calendar Control 8.0*.

### Setting a handler by class



1. Press **Ctrl+H** to create your header line.
2. Type **E** to choose *Event*. The *Event dialog* will appear.
3. Choose *Event Type: Activex*. Tab to the next field. The *ActiveX Event dialog* will appear.
4. Usually, you zoom from the first field after Object, to select your ActiveX variable. This time, however, zoom from the field after that, as shown in the screenshot above. This will allow you to select the COM object directly, without tying it to one variable.
5. Last, zoom from the *Event:* field to select the ActiveX event you want to respond to. Here we selected *DbClick*.

## How do I Enable Event Handling for a COM Object While in a Descendant Task?

If you have several COM objects that do similar things, you will find it convenient to do create event handlers that work in one of the parent tasks. For instance, it is often useful to create error handlers that work globally, in the Main program, to handle errors in all the COM objects of one type.

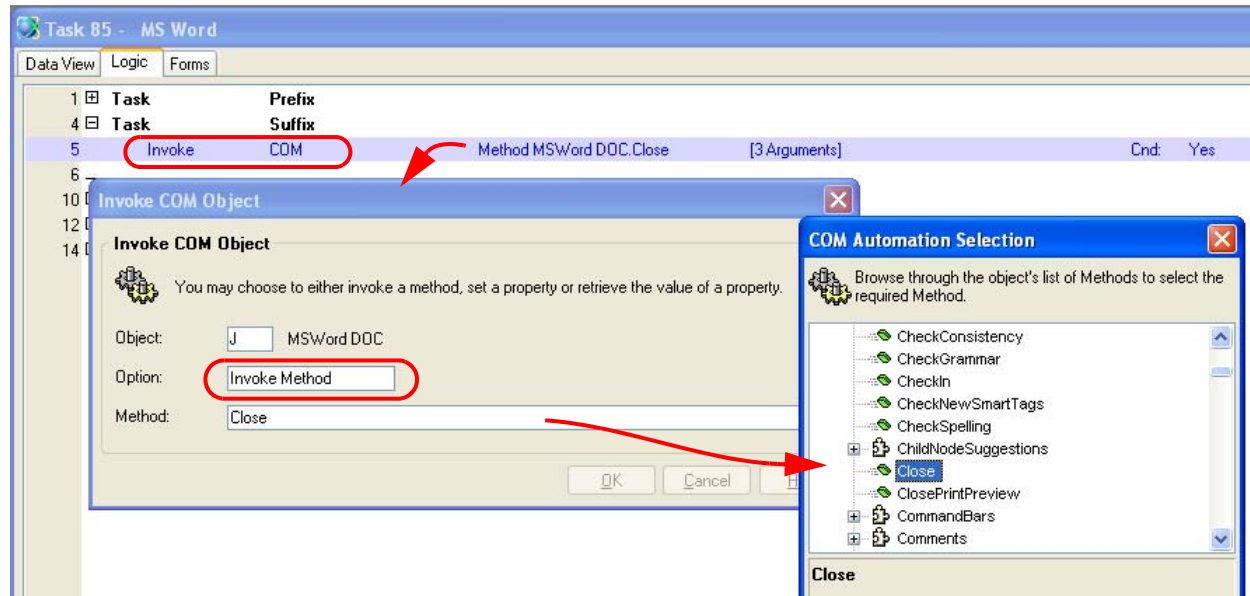
Here is how to do it.

COM

27									
28	Event	ChadoSpellText.SpellText.SpellingError2				Scope:	Global	Cnd:	Yes
29	Variable	Virtual	9	BadWord	Unicode	40	Task		
30	Variable	Virtual	10	Suggestions	Unicode	40	SubTree		
31	Variable	Virtual	11	OffsetOfWord	Numeric	N10	Global		
32	Variable	Virtual	12	WhatToDo	Numeric	N10			
33									

1. Create an event that responds to any COM object of a specific type. In this example, we have a global handler for a spelling error, using a 3rd party spell-checking tool. See Chapter 15, “How do I Set a Single Event Handler for Several COM Objects of the Same Type?” on page 378 to see how to set an Event handler like this.
2. Set the Scope appropriate to what you are trying to do. For example, to set a handler in the Main program that will respond to any program, set *Scope*: to **Global**. If you want a parent task to respond to a subtask, set the *Scope* to **SubTree**.

## How do I Call a COM Object Method?



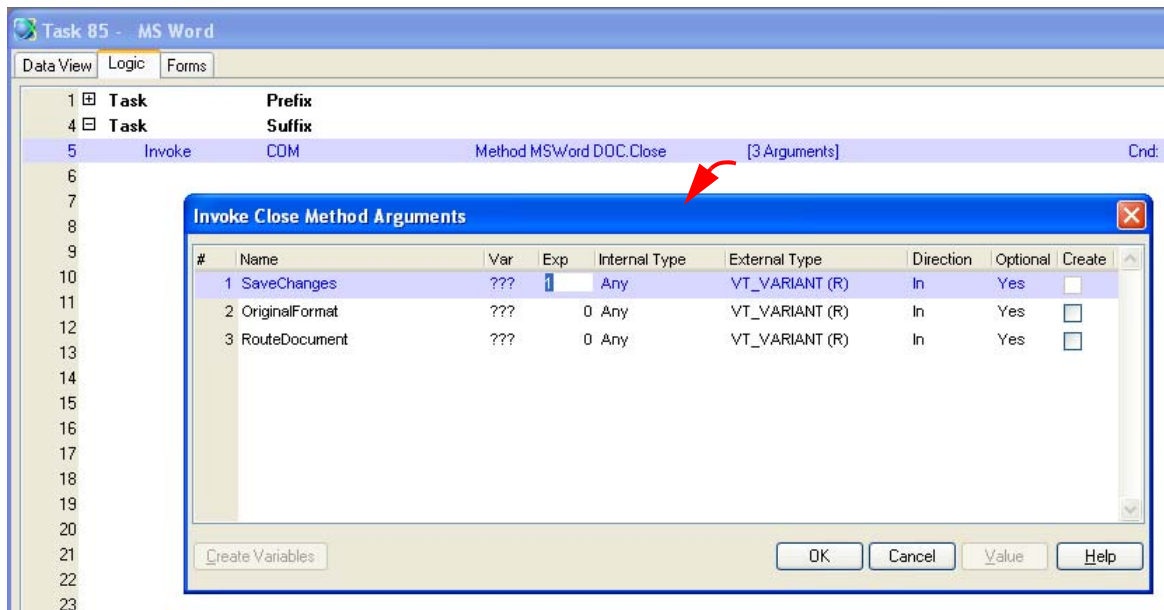
Once your COM object is declared, you can call it much as you would call a program or a subtask. However, a different operation is used, *Invoke*, to differentiate the call from a call within eDeveloper.

### Calling a COM Method

1. Open up a line in the appropriate logic unit.
2. Set the operation to invoke by typing **I** or selecting from the drop-down list.
3. Set the invoke type to **COM** by typing **C** or selecting from the drop-down list. Tab.
4. Zoom to bring up the *COM object dialog*.
5. Zoom from the *Object* field to select your COM object.
6. Select *Invoke Method* from the drop-down list on the Option field.
7. Zoom from the *Method* field to select the method you want to invoke.



8. Press Escape to close the COM Object dialog. Tab to the arguments field, and zoom.



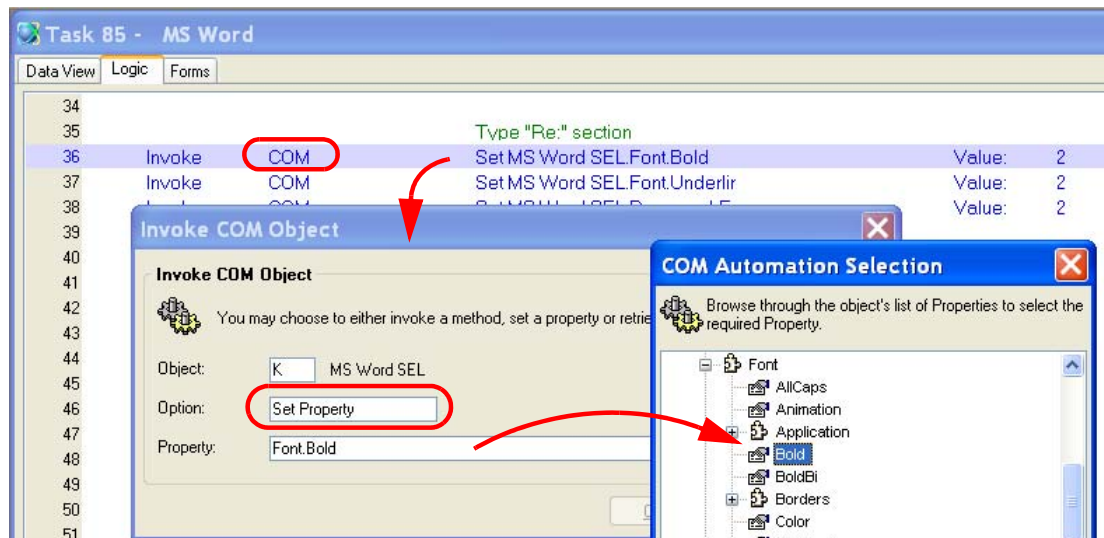
9. Now you will be presented with a list of arguments for this method. Note that you get a lot of information about these arguments in the list, such as whether it is an input or output parameter, whether it is optional or not, and the data type.

If you click on the box in the Create column, eDeveloper will create a variable that matches the argument data type. However, eDeveloper also does a lot of the data conversion for you, so you don't have to be as exact about the data types as you would if you were using other programming tools. For instance, this example uses the VT\_VARIANT type, which, if you study COM objects, means you need to pass in a pointer. But we just pass it an expression (which happens to be the number zero in this case) and eDeveloper does the rest of the work. Similarly, if you are passing in a numeric value, you needn't worry about issues such as whether the number is float or long or packed integer or whatever.

Now you have created a call to a method in your COM object.

**Hint:** To save time, you can copy Invoke COM operations to start the next one. This is particularly useful if you are invoking the same method over and over, as will happen when, for instance, you are formatting a Word document using COM objects and need to add paragraph breaks. Use **Ctrl+Shift+R** to copy an existing operation, or **Ctrl+C** Copy and **Ctrl+V** Paste.

## How do I Set or Get a Property of a COM Object?



Once your COM object is declared, you can change its properties by using Set Property, or fetch properties by using Get Property.

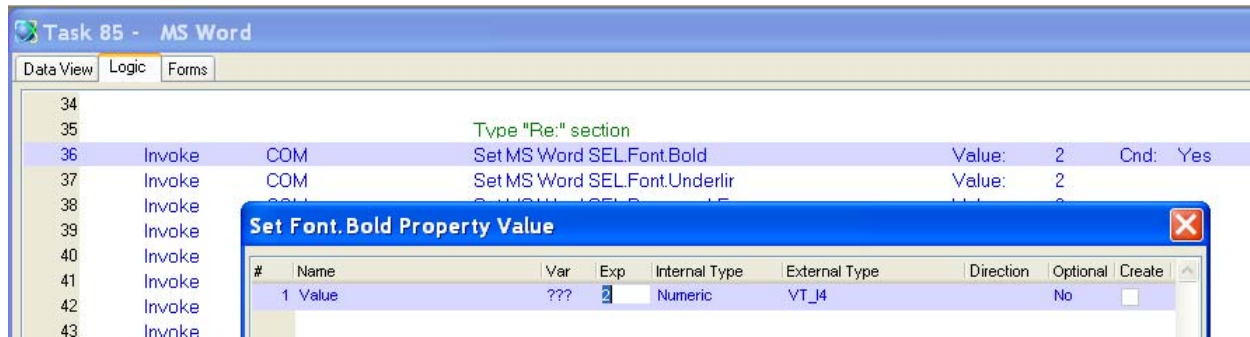
Set Property is used to change the object. For instance, with an ActiveX object, you might use Set Property to change the color of the object or how it is displayed. In our example here, which is writing to a Word document, Set Property is used to change the current font style to bold. Get property is used to query the settings of those properties.

Get and Set Property are also used to change and retrieve the value of an object. For instance, in the Calendar object, Set Property is used to set the default date, and Get Property is used to retrieve the date the user selected.

### Using Get and Set Property with a COM Object

1. Open up a line in the appropriate logic unit.
2. Set the operation to invoke by typing **I** or selecting from the drop-down list.
3. Set the invoke type to **COM** by typing **C** or selecting from the drop-down list. Tab.
4. Zoom to bring up the *COM object dialog*.
5. Zoom from the *Object* field to select your COM object.
6. Select *Get* or *Set Property* from the drop-down list on the *Option* field.
7. Zoom from the *Method* field to select the method you want to invoke.

8. Press Escape to close the COM Object dialog. Tab to the arguments field, and zoom.



9. Now you will be presented with a list of arguments for this Get or Set. Note that you get a lot of information about these arguments in the list, such as whether it is an input or output parameter, whether it is optional or not, and the data type.

If you click on the box in the Create column, eDeveloper will create a variable that matches the argument data type. However, eDeveloper also does a lot of the data conversion for you, so you don't have to be as exact about the data types as you would if you were using other programming tools. For instance, this example uses the VT\_I4 type, which, if you study COM objects, means you need to pass in "signed long integer". But we just pass it an expression (which happens to be the number one in this case) and eDeveloper does the rest of the work.

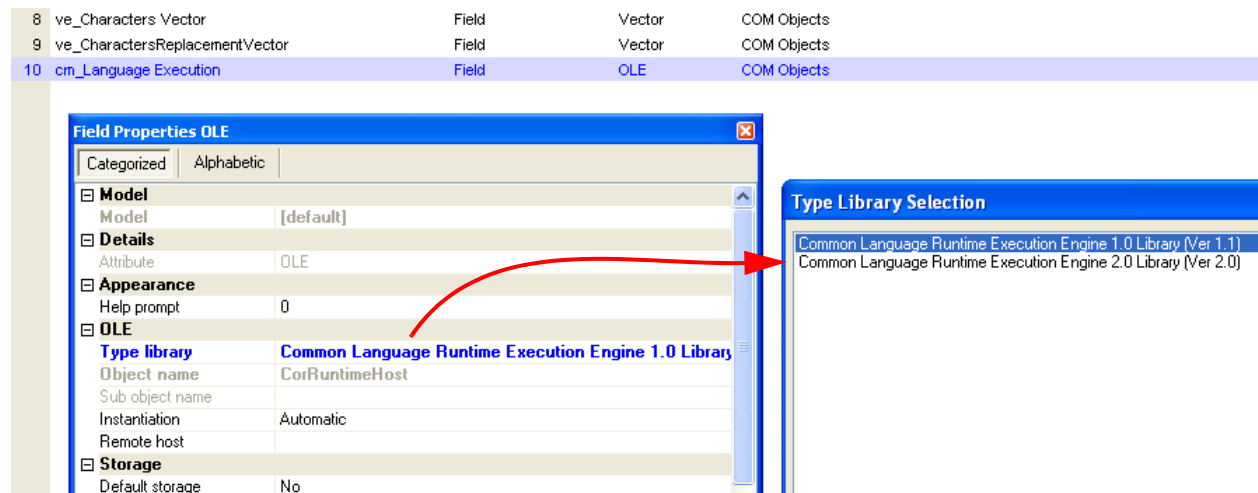
Now you have created a call to Get or Set a property in your COM object.

**Hint:** To save time, you can copy *Invoke COM* operations to start the next one. This is particularly useful if you are invoking the same method over and over, as will happen when, for instance, you are formatting a Word document using COM objects and need to add paragraph breaks. Use **Ctrl+Shift+R** to copy an existing operation, or **Ctrl+C** Copy and **Ctrl+V** Paste.

## How do I Change a Reference to a Certain COM Object?

One very nice thing about the COM standards is that they are designed to be forward compatible. That is, if the COM object conforms to standards, then when you install a new version of the COM object, the old methods should still work. The standards cover a lot of details about this, and when you write your own COM objects, you should conform to that standard.

However, the library names do change. For instance, if you have upgraded Microsoft Word on your computer year after year, you may have several versions of the Microsoft COM libraries all registered at once. If your COM object is pointing to an older version, it won't automatically upgrade when you install the new version. Worse, when you install your COM object on another computer that doesn't have the old libraries, the calls to the object will fail.



For example, here is a COM object which is a “Ver. 1.0” library. “Ver.2.0” also happens to exist on this computer. zooming from the Type library, eDeveloper recognizes that these two libraries are for the same object, and allows us to choose the new one.

If we were running on a computer that did not contain the old library at all, eDeveloper would still show the old library name, and allow us to zoom and select the new replacement library.

Note that eDeveloper does not allow us to change the Object name, nor does it allow us to change the Type library to anything other than an upgraded version of this object. This protects you from inadvertently changing in object in ways that will cause all references to it to fail.

### Changing a COM Library Reference

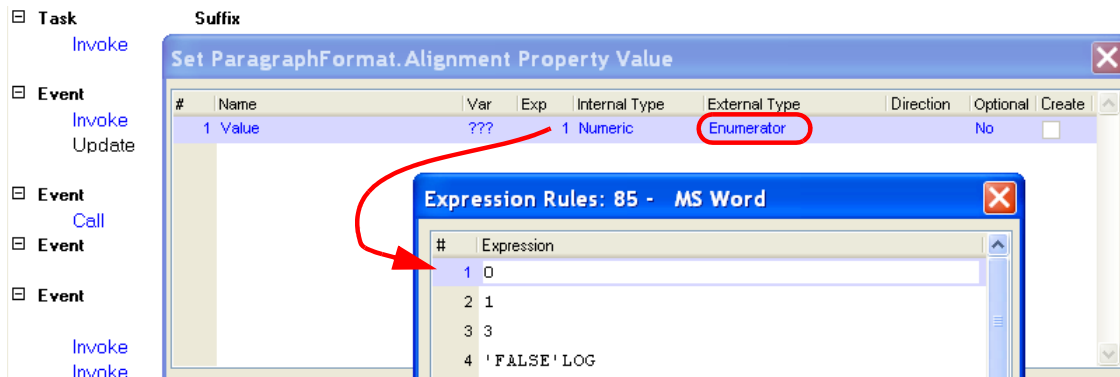
1. Go to the definition of the COM object in question (in the data section of the task, or in the Model repository).
2. On the property sheet (**Alt+F1**), go to the *Type Library* property.
3. **Zoom** (**F5**, or double click) to bring up a *Type Library Selection* box. You may see only one entry, or you may see more, depending on how many revisions exist for this object.

4. Press Enter to select the upgraded object.

Now, all calls to the object will use the upgraded library.

COM

## How do I Set a Value of an Enumerated Type Parameter of a COM Object?

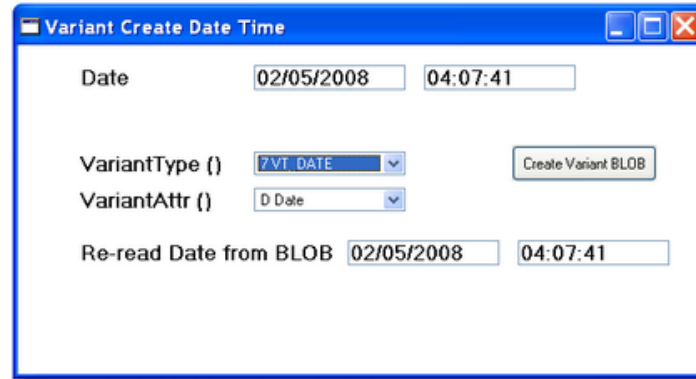


An enumerated type is a type whose legal values consist of a fixed set of constants. Examples would be days of the week, months of the year, or something like types of paragraph alignments. Often in COM objects, when you have a choice between several things, you pass an integer to make a choice.

Exactly what integer stands for which choice is something you need to look up in the documentation for the COM object, or you can experiment and see what happens.

## How do I Extract/Set Data From/To a Variant Type of a COM Object?

COM



Many COM objects use the VT\_VARIANT data type. The definition of the VT\_VARIANT, is, in essence, that it can hold any data type. However, the object you are using will be expecting data of a certain format for a specific function.

Usually, eDeveloper will handle the conversions automatically. If you know that a certain VT\_VARIANT parameter is expecting an array, for instance, and you set up a vector, you just pass the vector in and eDeveloper does the rest (see Chapter 15, “How do I Send and Retrieve Array Values from COM Objects?” on page 394 for an example). Similarly, if the object is sending back a negative integer in a VT\_VARIANT, and you accept the parameter in a numeric variable that allows negatives, eDeveloper will handle the underlying conversion correctly.

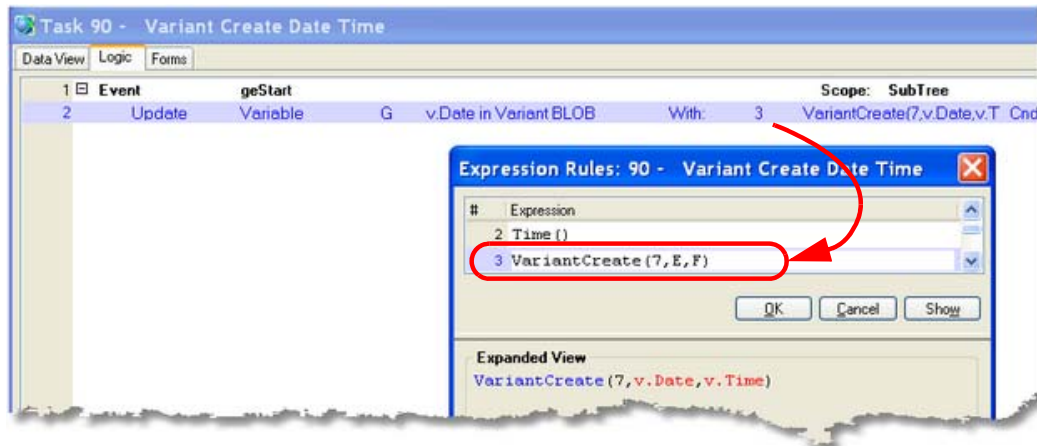
However, if you need a more precise level of control, you can use a BLOB data type to pass data back and forth, and use the Variant functions to set up exactly the type of variant you want to send. You can also use the same functions to extract data from a variant that is sent back from the object.

Here is a summary of the functions used.

- **VariantCreate()**: Copies data into a BLOB variant. See Chapter 15, “Creating a Variant with VariantCreate()” on page 388.
- **VariantGet()**: Copies data from a BLOB variant into a variable. See Chapter 15, “Extracting data from a Variant using VariantGet()” on page 389.
- **VariantGetVector()**: Copies data from a BLOB variant into a vector. See the eDeveloper Help files).
- **VariantAttr()**: Returns the eDeveloper data type (Alpha, Numeric, Date, etc.) of the variant. See Chapter 15, “Extracting the attribute type using VariantAttr()” on page 392.
- **VariantType()**: Returns the data type (a number representing the standard variant types, such as VT\_DATE, VT\_I4, etc.). See Chapter 15, “Extracting the data type using VariantType()” on page 393.

## Creating a Variant with VariantCreate()

You can use the **VariantCreate()** function to populate the variant BLOB. In this example, we used **VariantCreate()** to move a Date and a Time into the VT\_DATE variant, which can actually hold a date/time stamp.



The syntax of VariantCreate() is:

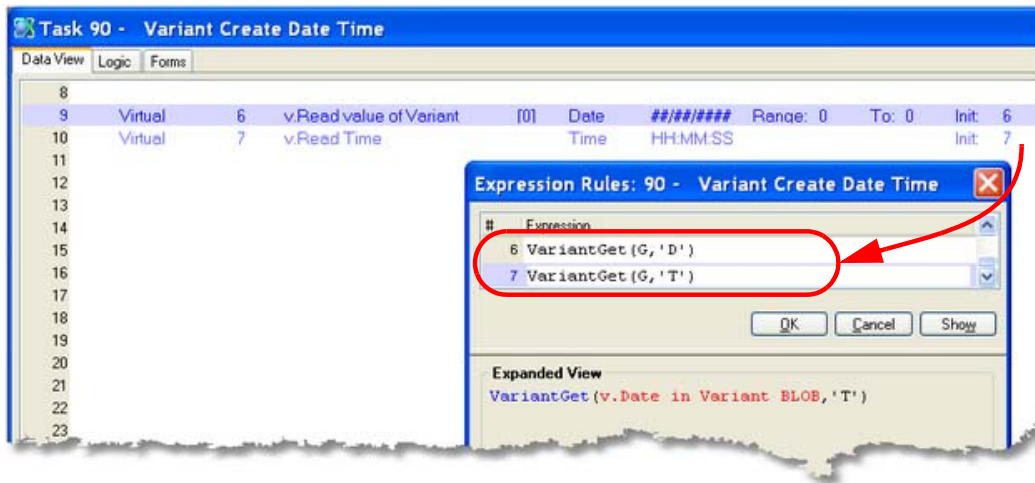
```
VariantCreate(VT Type, Value, Time value)
```

where:

- **VT Type:** A number representing the data type. For instance, in our example, 7 represents the date type. See Chapter 15, "Variant Data Types" on page 390 for a list of the types.
- **Value:** The value you are putting into the variant. This could be any data type, a date, string, number, or BLOB.
- **Time value:** (optional) This allows you to move a time into a VT\_Date type. In this example, we pass in a Time as the third parameter.



## Extracting data from a Variant using VariantGet()



The syntax of **VariantGet()** is:

`VariantGet (Variant Value, Attribute)`

where:

- **Variant Value:** The BLOB that has your Variant.
- **Attribute:** A letter representing the type you are extracting. See Chapter 15, “Variant Data Attributes” on page 390 for a list of the codes.

In this example, the variant is a VT\_DATE type, which holds both a Date and a Time, so we can extract both from the same variant using **VariantGet()**.

## Variant Data Attributes

Attribute Letter	Developer Data Attribute
A	Alpha
N	Numeric
L	Logical
D	Date
T	Time
B	Boolean
U	Unicode

## Variant Data Types

Here is a list of the data type that are used in the Variant functions.

VarType returns	Enumeration symbol	VariantAttr returns	
0	VT_EMPTY	No value specified	
1	VT_NULL	SQL-style Null	
2	VT_I2	Signed 2-byte integer	–32,768 to 32,767
3	VT_I4	Signed 4-byte integer	–2,147,483,648 to 2,147,483,647
4	VT_R4	Signed 4-byte real	1.1E -38 to 3.4E +38 (7 digits)
5	VT_R8	Signed 8-byte real	2.2E -308 to 1.7 E +308 (15 digits)
6	VT_CY	Currency	
7	VT_DATE	Date	
8	VT_BSTR	Automation string	
9	VT_DISPATCH	A pointer to an object that implements <b>IDispatch</b>	
10	VT_ERROR	SCODE	
11	VT_BOOL	Boolean	
12	VT_VARIANT		
13	VT_UNKNOWN	A pointer to an object that implements <b>IUnknown</b>	
14	VT_DECIMAL	Decimal	
16	VT_I1	1-byte character	–128 to 127
17	VT_UI1	Unsigned 1-byte character	0 to 255
18	VT_UI2	Unsigned 2-byte integer	0 to 65,535

VarType e returns	Enumeration symbol	VariantAttr returns	
19	VT_UI4	Unsigned 4-byte integer	0 to 4,294,967,295
22	VT_INT	Signed machine integer	
23	VT_UINT	Unsigned machine integer	
36	VT_RECORD	User defined type	
8192	VT_ARRAY		An array of data type
16384	VT_BYREF		A reference to data type

## How do I Determine the Type and Corresponding eDeveloper Attribute of a Variant Value Belonging to a COM Object?

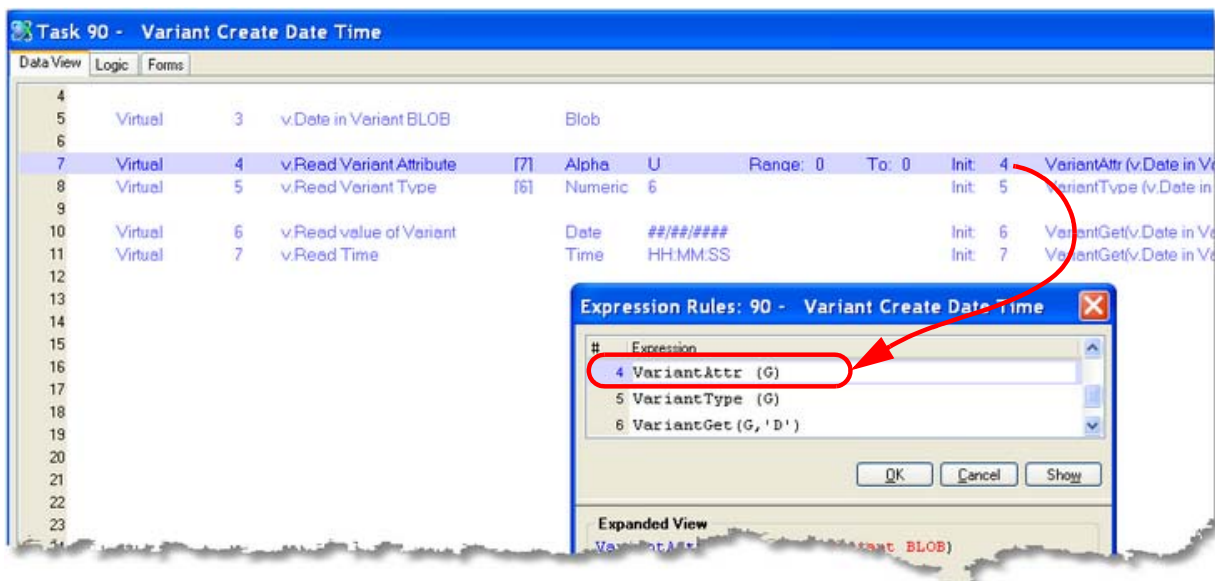
Many COM objects use the VT\_VARIANT data type. The definition of the VT\_VARIANT, is, in essence, that it can hold any data type. So when you see a parameter of this type, you don't necessarily know what is being passed.

In practice, you will usually know what the object is expecting or sending because you are copying an example, or you have the documentation for the object. eDeveloper is very good at converting the data for you, so you don't usually need to do the conversions manually. For instance, if you accept a VT\_VARIANT type in a numeric variable, and the object was in fact passing back a number, then the work is done for you.

However, it could be the case that one object sends back a numeric in a variant for one call, and an alpha for another call, in which case you would have to accept the data in a BLOB and then figure out what it contains.

- One way to do this is with the **VariantAttr()** function. This function accepts a BLOB as a parameter, and returns the data type of the variant as a one character code. That code can then be used with the **VariantGet()** function to extract the data. You can also use the **VariantType()** function, which returns a code specifying the underlying data type (such as VT\_I8).

### Extracting the attribute type using VariantAttr()



The syntax of **VariantAttr()** is:

```
VariantAttr(Variant)
```

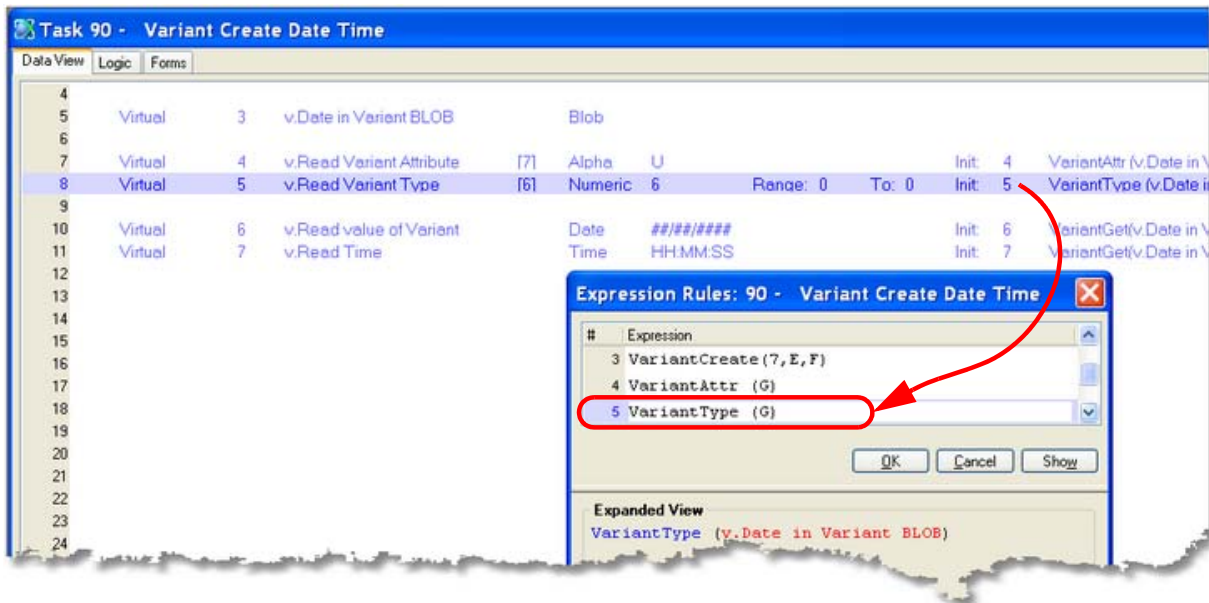
where:

- **Variant:** The BLOB that has your Variant.

This returns a letter that represents the eDeveloper attribute type. See Chapter 15, “Variant Data Attributes” on page 390 for a list of those types.

In this example, the variant is a VT\_DATE type, so the letter ‘D’ is returned, corresponding to the eDeveloper “Date” attribute. Now that we have the attribute type, we can use that with the **VariantGet()** function to fetch the data out of the variant into the proper variable.

### Extracting the data type using VariantType()



The syntax of **VariantType()** is:

`VariantType(Variant)`

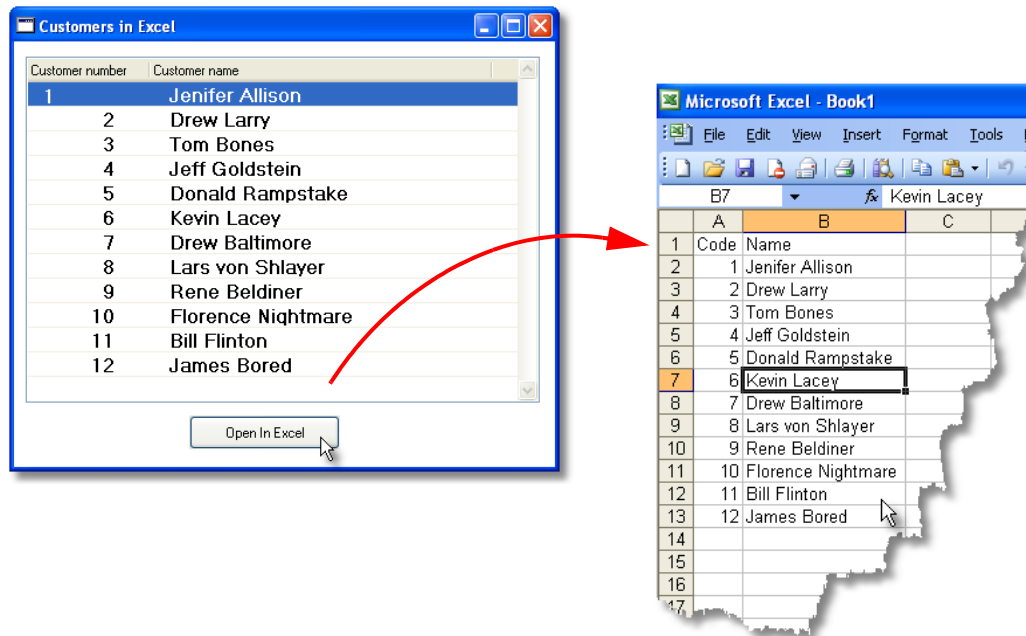
where:

- **Variant:** The BLOB that has your Variant.

This returns a letter that represents the eDeveloper attribute type. See Chapter 15, “Variant Data Types” on page 390 for a list of those types.

In this example, the variant is a VT\_DATE type, so the number 7 is returned.

## How do I Send and Retrieve Array Values from COM Objects?



You can see some of the real power of eDeveloper and COM objects when you start working with arrays and COM objects. In the example shown above, we processed a table and moved it into an Excel spreadsheet, using only a few lines of code.

You pass a vector to and from a COM object just as you would pass any other parameter. You do need to know approximately what the COM object is expecting or sending in terms of whether the data is numeric or alpha, but you don't need to worry about the details (whether the number is Float or Long, for example).

You also may need to make allowances for how many items there are in the vector. In this example, for instance, we used **Counter(0)** to keep track of how many customers were in the table, so we could allocate the correct number of cells in the Excel worksheet.

Passing an array to a COM object

- 1. First, set up your vector. In this example, we are using a two-dimensional array, where each line is one-dimensional array containing the customer number and customer name.

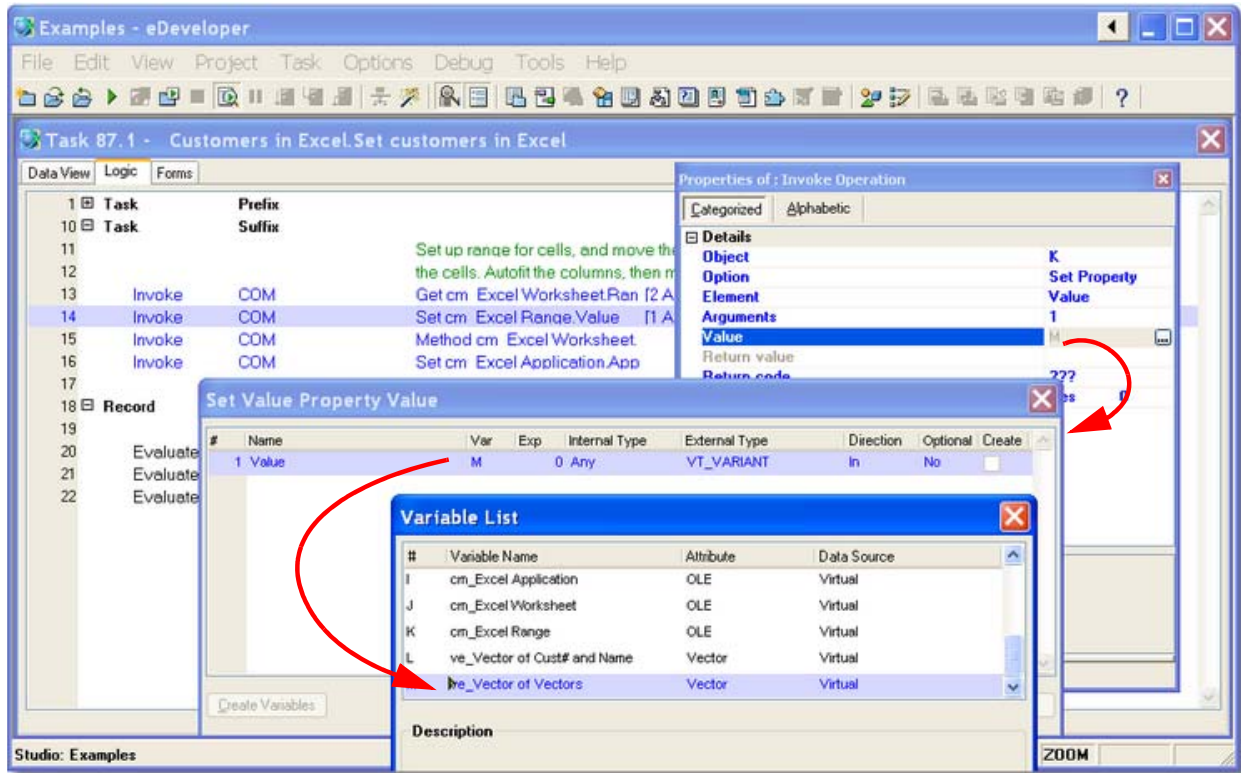
The first vector is just a string array. When it is full, it contains two elements: the customer number (as a string) and the customer name (also a string). This corresponds to one horizontal row in the spreadsheet.

The second vector is an a vector where each row is using the model of the first vector. This corresponds to the entire spreadsheet.

Task 87.1 - Customers in Excel.Set customers in Excel				
Data View				
	Main Source	Customers	Index	
1	Column	1	Customer number	Numeric 3
2	Column	2	Customer name	Alpha 30
3				
4	Virtual	1	cm ExcelApplication	[14] OLE
5	Virtual	2	cm ExcelWorksheet	[15] OLE
6	Virtual	3	cm ExcelRange	[16] OLE
7				
8	Virtual	4	ve Vector of Cust# and Nam	[18] Vector
9	Virtual	5	ve Vector of Vectors	[19] Vector
10				
11				

Task 87.1 - Customers in Excel.Set customers in Excel				
Logic				
1	Task	Prefix		
10	Task	Suffix		
18	Record	Suffix		
19				Create a vector line for each row
20	Evaluate	Expression	7	VecSet("L\VAR,1.Str(Customer number,'6L')")
21	Evaluate	Expression	8	VecSet("L\VAR,2.Customer name")
22	Evaluate	Expression	9	VecSet("M\VAR,Counter(0)+1,ve Vector of Cust# and N

2. Next, since we are passing a vector into the COM object, we fill the vector up with data, using the **Vec-Set()** function.



3. After the vector is set up, passing it in is easy. The vector is simply passed in as any variable would be.



## How do I Handle a Collection in a COM Object?



Index	Key (ID)	Name
1	1024	(no proofing)
2	1025	Arabic (Saudi Arabia)
3	1026	Bulgarian
4	1027	Catalan
5	1028	Chinese (Taiwan)
6	1029	Czech
7	1030	Danish
8	1031	German (Germany)

A *collection* object is a list of items, such as “recently used files”, “Zip codes”, or in this instance, “languages”.

The list must contain a unique identifier, which can either be:

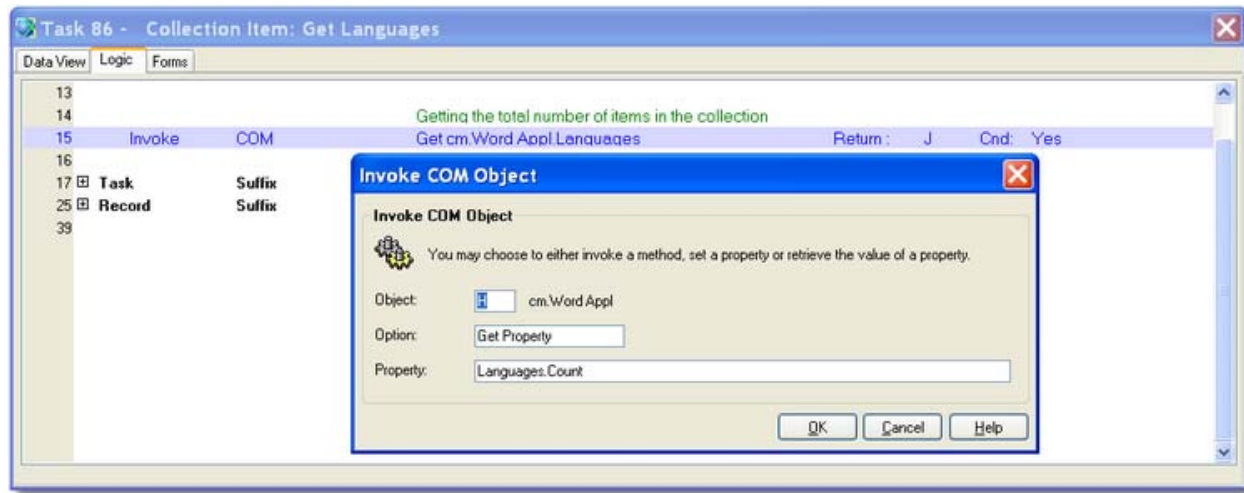
- an *Index*, which is a sequential number from 1 to the number of items in the list.
- a *Key*, which is any string or number that uniquely identifies this item in the collection.

Each collection will contain either an index or a key, but not both. In the example above, the languages list from Microsoft Word contains only a key type of ID. This makes it difficult to extract the entire list, because you don’t know in advance what the IDs are.

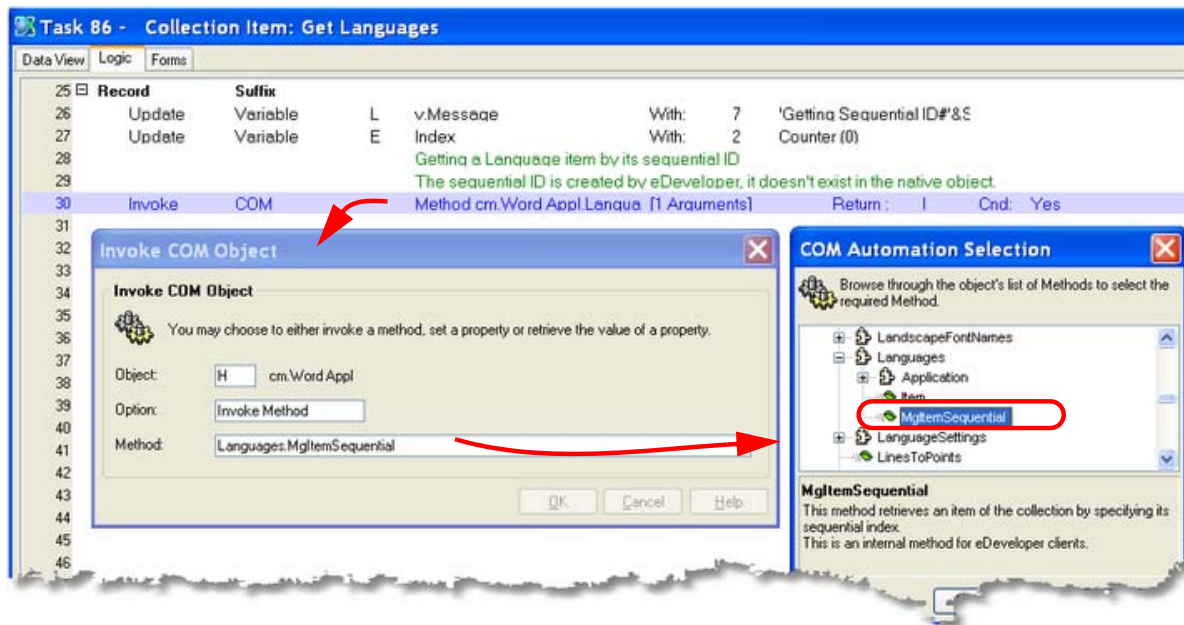
Fortunately, eDeveloper solves this problem for you by providing a special method for collections called *MgItemSequential*. This method isn’t part of the native object, but appears on the method list with all the other methods. It allows you to fetch each item in the list using an index from 1 to the number of items. Once the items are in a table, as shown above, you can use them as you would any table in eDeveloper.

The example below fetches the list of supported languages from Microsoft Word.

## Fetching a Collection

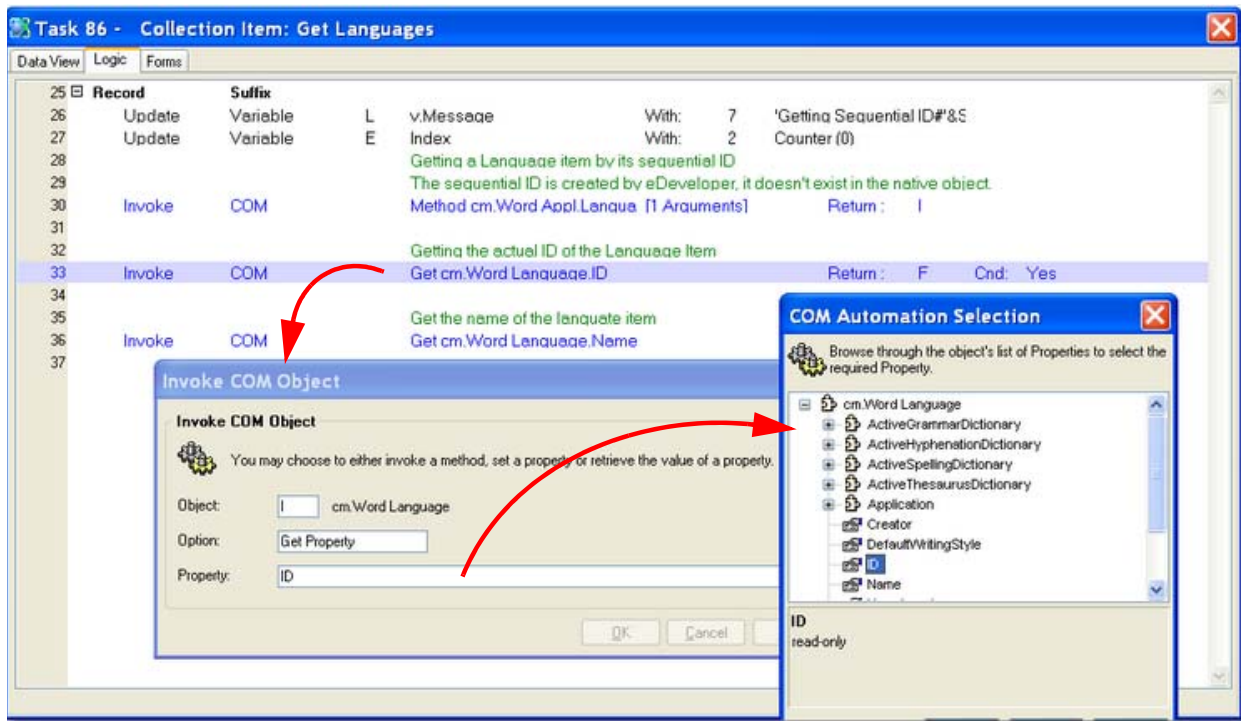


1. First, determine the size of the entire collection. In this example, there is a method in Word Application.Languages object called Count, which gives us the total number of languages. This is used in the *End Task Condition* so we loop until we have fetched the entire list of languages.



2. Then, for each item in the list, use the *MgItemSequential* method to fetch the item. *MgItemSequential* does not exist in the native object, but you will see it on the COM Automation Selection list.

The return parameter returns an object, which in this case is the Word Application.Language object.



- Now, use the object returned by *MgItemSequential* to fetch the properties of the object. In this case we fetch the ID and the Name, which we store in our table.

Once the task is done, we will have the collection stored in an eDeveloper table, ready for use.

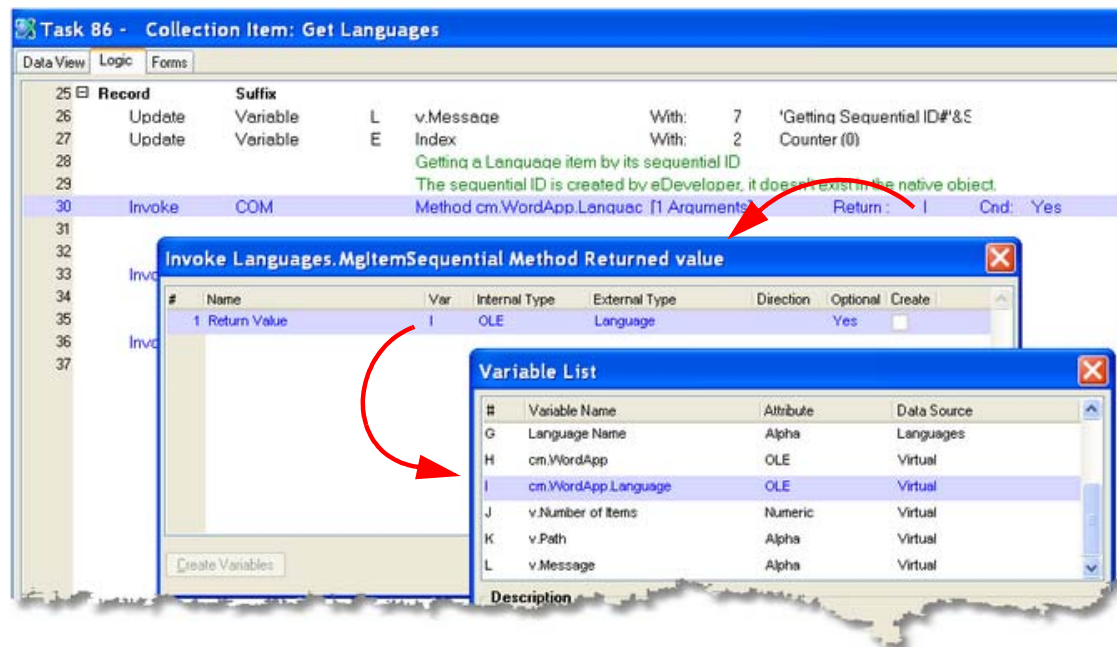
## How do I Pass/Receive a COM Object as a Parameter To/From a COM Object?

COM objects frequently pass and receive other COM objects as parameters. While this may seem a little odd when you first work with COM objects, it is very easy to do in eDeveloper. Since COM objects are just variables, you pass them as you would any other variable.

### Receiving a COM Object

This example uses the `MgItemSequential` method that was discussed in Chapter 15, “Fetching a Collection” on page 398.

**Prerequisite:** You must have the COM object declared within the scope of your program.



1. Go to place where you will be sending or receiving the parameter. In this case, we are receiving a COM object as the returned value.
2. Zoom from the Var column as you would for any other variable. Our COM object is listed, and we select it by pressing Enter.

Now, when the method is invoked, the `WordApp.Language` COM object will be ready for use by the next Invoke operation.

## How do I Re-use COM Object Definitions?

Part of having a good, solid, maintainable application is to re-use items as much as possible. This not only reduces errors and makes the application easier to maintain, it also makes coding faster. eDeveloper provides a very easy way to re-use data definitions and control definitions in the *Model Repository*.

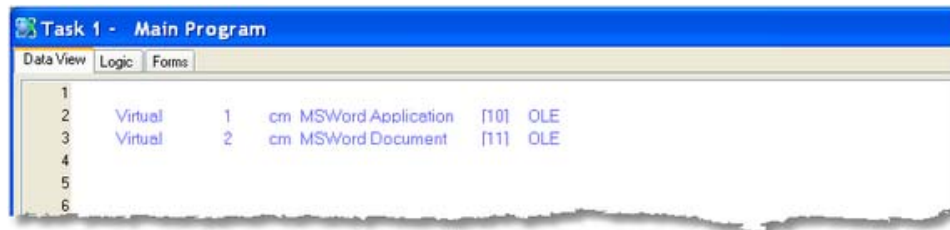
A COM object definition is defined in the Model repository exactly as it would be if you defined it directly inside your task. The only difference is, you can define it once and re-use it as much as you like. Also, when the object is defined in the Model repository, you can use **Find Reference (Ctrl+F)** to easily get a list of everywhere that object is used.

Details for how to set up a COM Object in the Model Repository are found in Chapter 15, “How do I Define the COM Object That I Want to Use?” on page 375.

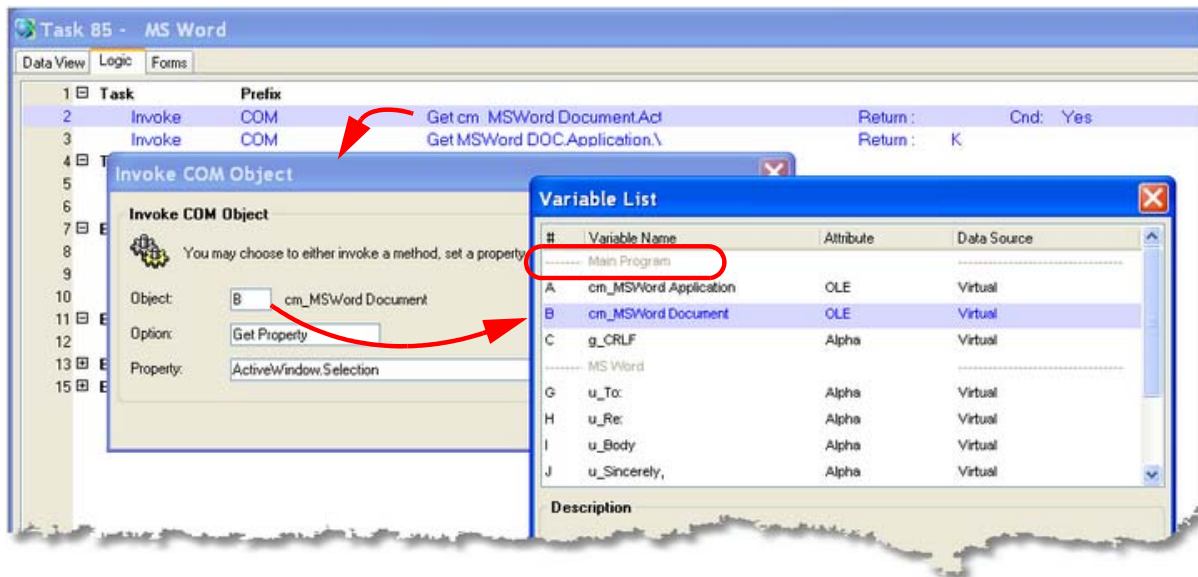
## How do I Keep an Instance of a COM Object Available Across Programs?

You might have an occasion where you want to instantiate a COM object and keep it open while different programs run. To do this, define the COM object in the Main Program. Then, it will be available for any program in the Program repository. You can use Task Prefix of the Main Program to initialize the object, if needed, so it is ready for use when the programs first run.

### Defining a COM object in the Main Program



1. Define the COM objects in the *Data View* of the *Main Program* just as you would if you were defining them in your task.



Now, when you are selecting a COM object, you will see the COM objects listed on the variables list at the very top, under the “Main Program” header.

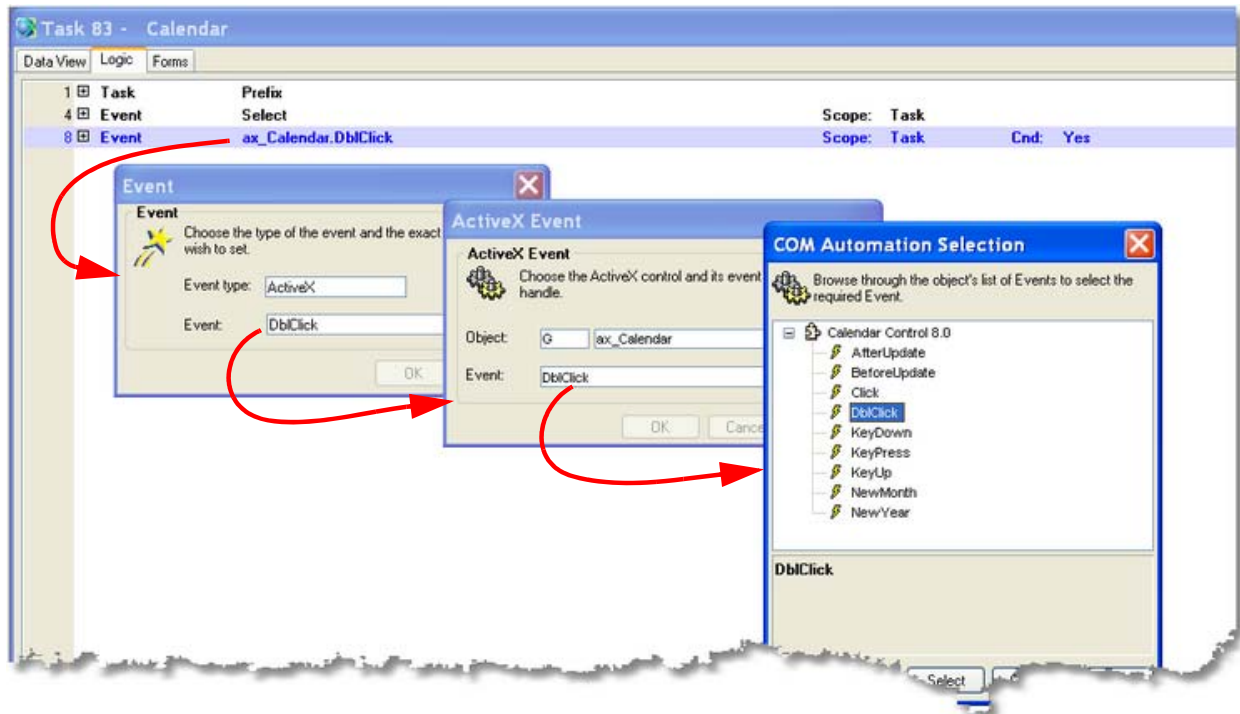
**Note:** COM objects can also be shared by passing them between programs as parameters.

## How do I Trap Events Triggered by a COM Object?

COM objects generate their own set of events. These events are automatically detected by eDeveloper, and all you need to do is decide which ones you want to use.

COM

### Creating an Event Handler for a COM event

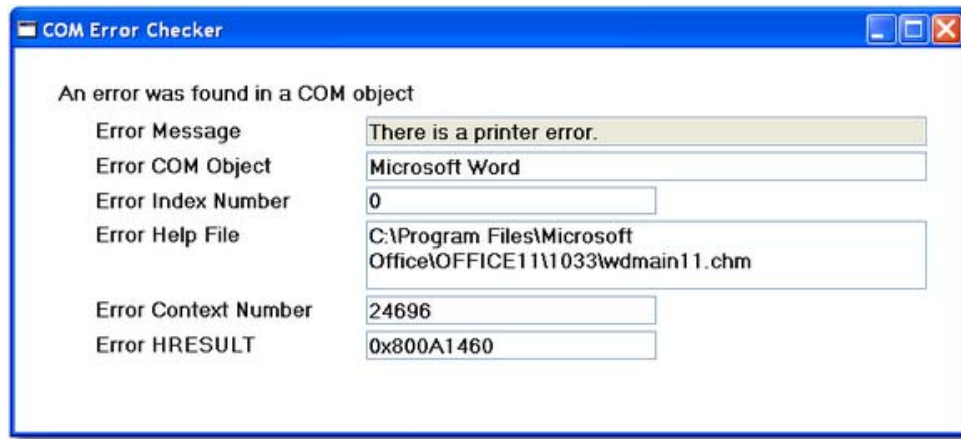


1. Open up a header line (**Ctrl+H**) in the Logic section.
2. Type **E** to select the Event.
3. A dialog box titled “Event” will open. For the *Event Type*, select *ActiveX*.
4. Zoom from the *Event* field. A dialog box titled “ActiveX Event” will open.
5. Zoom from the *Object* field. A list of all the available ActiveX objects will appear. Select the object whose event you want to trap.
6. Zoom from the *Event* field. A *COM Automation Selection* dialog box will appear. It will show your object, and a list of all the events raised by that object. Select the event you want to trap.
7. Press OK to close all the dialog boxes.

You now have an event that will be triggered whenever the object raises the selected event.



## How do I Handle an Error Triggered by a COM Object?



You can capture errors generated by a COM object by using the eDeveloper **COMError()** function. COM-Error returns the last COM generated error. The syntax is:

```
COMError( number )
```

where *number* determines what kind of information is passed back. The value returned is always an alpha string.

- 1 returns the error description
- 2 returns the name of the COM object
- 3 returns the index number
- 4 returns the help file for that error
- 5 returns the context number
- 0 returns the HRESULT code

**Hint:** You can encapsulate this function in one program or global function, which you can call to check for errors and give a message to the user if an error is found, or to log the error.

Also note that while you are debugging COM objects this information is always found in the eDeveloper debugger activity log.



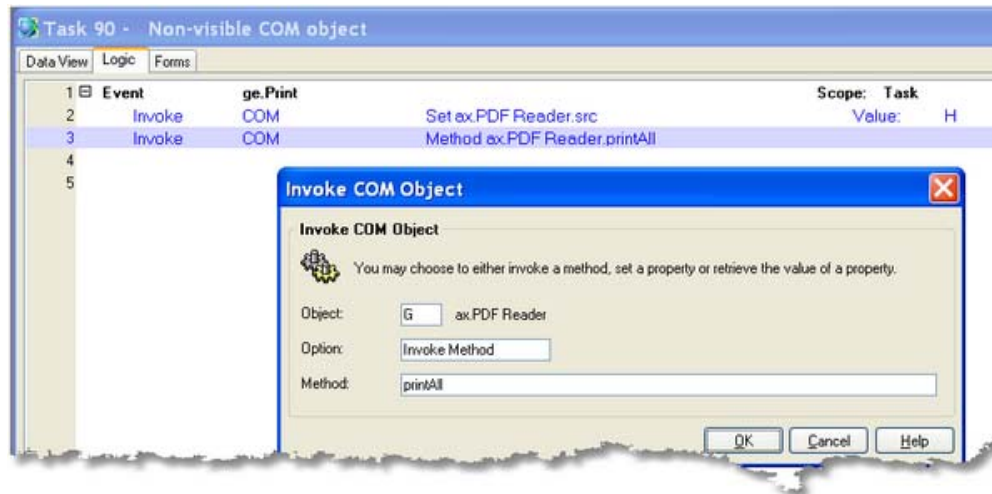
## How do I Handle an ActiveX Control Without Displaying it on the Form?

COM



By definition, ActiveX objects are designed to be viewed on a form. However, sometimes it is handy to use them without displaying them. In this example, we wanted to use Acrobat Reader to print a PDF, but do not want to allow the user to control how it is printed. The Acrobat Reader ActiveX object has the functions we need. How we did it is explained below.

### Using an invisible ActiveX Object



1. Declare your ActiveX object just as you ordinarily would (See Chapter 15, “How do I Define the COM Object That I Want to Use?” on page 375 for details).
2. Set up the logic you want to use with the ActiveX object. In this case, we use Set Property “src” to set the file name, then call the “printAll” method to do the printing.
3. Place the ActiveX object on the form, but make it very small.
4. In the *Control Properties* for the ActiveX control, set the *Visible* property to ‘FALSE’LOG.

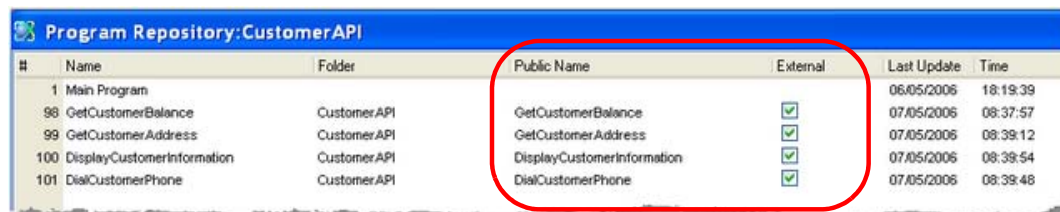
Now, your logic will work with the ActiveX control in the same way it would if it were visible.

## How do I Expose eDeveloper Logic as COM Methods?

In eDeveloper, you can write your own COM objects to act as an API into your application. That way, you can access your eDeveloper code from programs written using other tools. This is done using the COM interface builder, which does most of the work for you.

The interface builder is quite powerful, and gives you a lot of control over how your COM object will work. Here is a summary of the steps involved.

### Creating a COM interface

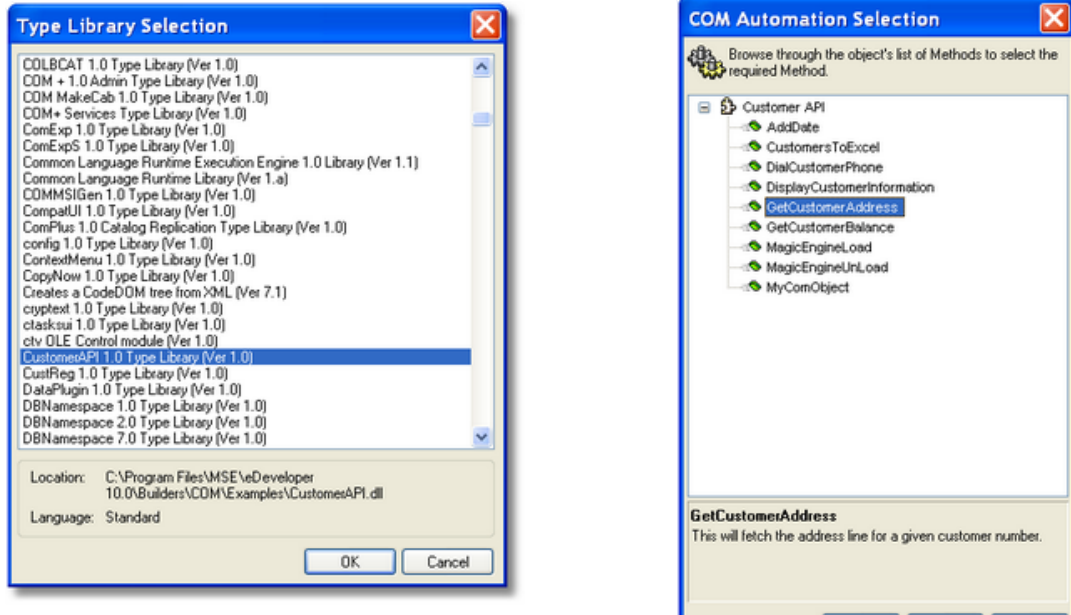


#	Name	Folder	Public Name	External	Last Update	Time
1	Main Program				06/05/2006	18:19:39
98	GetCustomerBalance	CustomerAPI	GetCustomerBalance	<input checked="" type="checkbox"/>	07/05/2006	08:37:57
99	GetCustomerAddress	CustomerAPI	GetCustomerAddress	<input checked="" type="checkbox"/>	07/05/2006	08:39:12
100	DisplayCustomerInformation	CustomerAPI	DisplayCustomerInformation	<input checked="" type="checkbox"/>	07/05/2006	08:39:54
101	DialCustomerPhone	CustomerAPI	DialCustomerPhone	<input checked="" type="checkbox"/>	07/05/2006	08:39:48

1. Decide what programs you want exposed in the COM object. Give those programs public names, and check the *External* box.
2. Select **Options->Interface Builder->COM**. The COM Interface Builder Wizard will appear. Press **Next**.
3. The next screen allows you to modify an existing component, or to create a new one. Press the New button to create a new component.
4. You will now be on the *Interface Settings* dialog. Type in the name of your COM object. Then select Remote or Local engine. Press **Next**.
5. Now you will be on the *COM Object Properties* dialog. Here you can set a number of properties for your COM object, including the application name, messaging server, user name, and password. You can also set a help key for links into a help file, and some help text that will be displayed to the programmer using the COM object. Fill these out as required by your application, then press **Next**.
6. Next, you will see the *Add Programs* dialog. Here you will see a list of all the programs that are marked External. Press the **Add>>** or **Add all** button to move the items you want in your COM object to the Selected column.
7. For each program selected, you can use the *Arguments* and *Details* buttons to configure details about how the arguments are passed, and to attach a Help file and help text. You can also rename the method here, so the method name does not have to be the same as the program name in eDeveloper. Configure your methods as needed, then press **Next**.
8. Now you will be on the *COM Object Information* screen. Here you can set the Version number of your COM object, the location of the Help library, the Class ID, and you can type in a general information bit of documentation. Fill the in as necessary, and press **Next**.
9. Next you will be presented with the *COM Object Path* dialog. Here you simply specify the path where you want your DLL to be generated. eDeveloper provides a default though, which you can use. Then press **Next**.

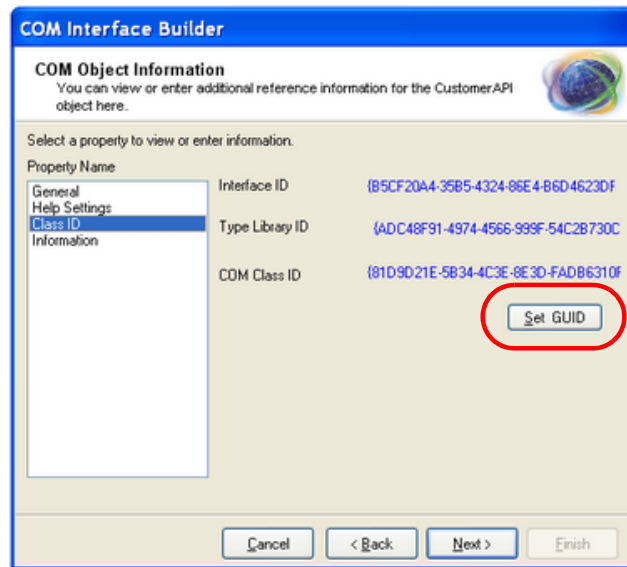
10. Finally, you will be presented with the Generate Component dialog. Press Finish and your component will be generated.

Last you will see the *SUCCESS!* dialog, which means your DLL exists. Before you can use it, however, you need to register it on your computer. For this you use [regsvr32.exe](#). You can run it from the **Start->Run** menu, typing in the name of your DLL, or create an installation script. But an easier method to use while testing is to create a shortcut to regsvr32.exe somewhere handy (you will find it in the WIN-DOWS\System32 directory, most likely), and just drag your DLL into the shortcut, which will automatically register it.



You can test your COM object by selecting it in eDeveloper just as you would any other COM object. On the left you can see that it is registered and so shows on the list with all the other COM objects registered on this system. On the right, you can see the list of methods that appears when we invoke the object.

## How do I set a Class ID When Exposing eDeveloper Logic as a COM Server?



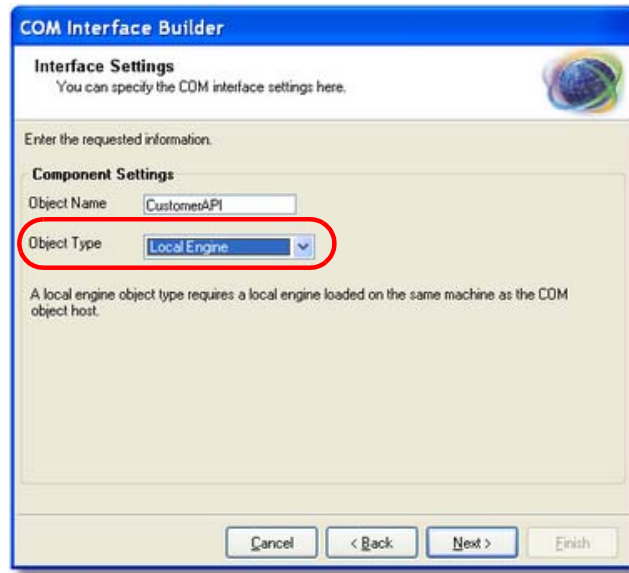
When you are creating a COM object, eDeveloper will automatically create Class IDs for you. However, if you need to enter your own, you can do so on the *COM Object Information* dialog while you are using the *COM Interface Builder*. Just press the Set GUID button and you can type in your IDs.

**See also:** Chapter 15, “How do I Define the COM Object That I Want to Use?” on page 375.

## How do I Configure a COM Client Locally Accessing eDeveloper as a COM Server?

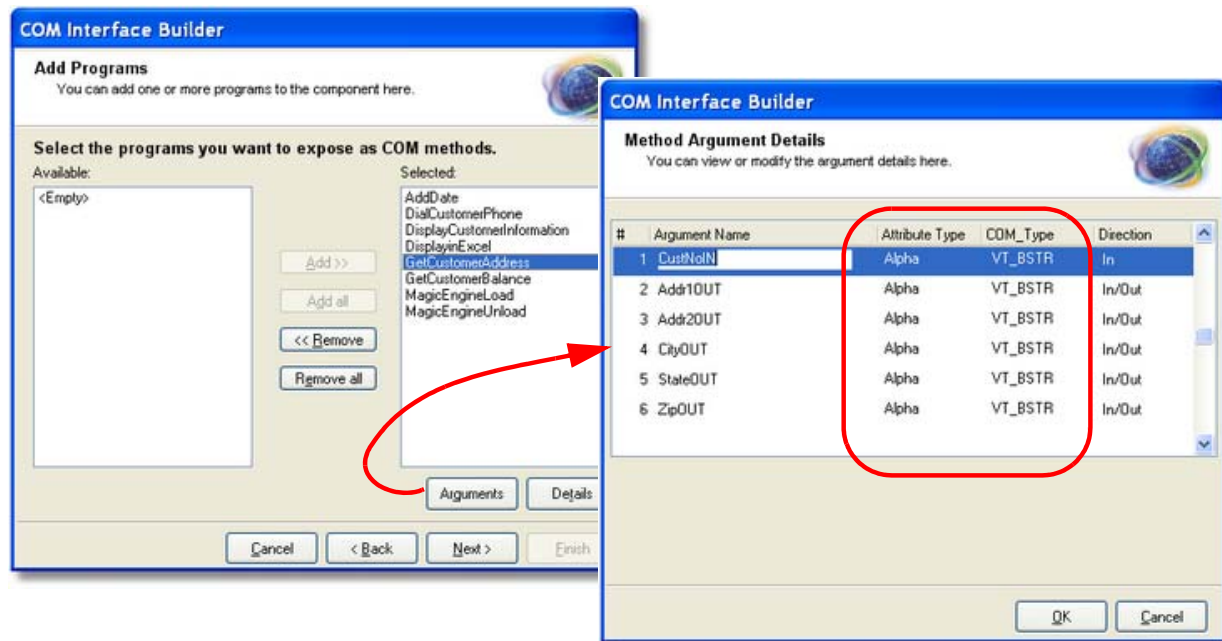
COM Objects can be created to run on the local machine, or they can be configured to be accessed remotely.

COM



1. When you are creating a COM object, you will be prompted for the Object Type. If the COM Object is designed to be run on the local machine, you must select the *Local Engine* as the Object Type.
2. Also, the DLL has to be registered on the machine it is running. You can do that using RegSvr32. If you are installing on multiple machines, you will want to automate that process using a batch file or script or installation utility.
3. The eDeveloper runtime engine must be available also. You can specify the location of the engine when you create your COM object, but otherwise the system will look in the registry and use the eDeveloper engine registered there.

## How do I Determine the COM Datatypes When Exposing eDeveloper Logic as COM Methods?



When you are creating your COM object using the COM Interface Builder, you can set up the parameter definitions.

### Viewing and Changing Method Argument Details

1. Select **Options->Interface Builder->COM**. The COM Interface Builder Wizard will appear. Press **Next**.
2. The next screen allows you to modify an existing component, or to create a new one. Select the component whose methods you want to view.
3. Press Next until you reach the Add Programs dialog. Here you will see a list of the methods that are in this COM object. Select the one you want to view, then press the Arguments button.
4. Now you will see a list of the arguments for this method. The com datatypes are displayed in the *COM\_Type* column. You cannot change them however, as eDeveloper automatically assigns them based on the definitions in eDeveloper.

For more information on COM datatypes see Chapter 15, “Variant Data Types” on page 390.

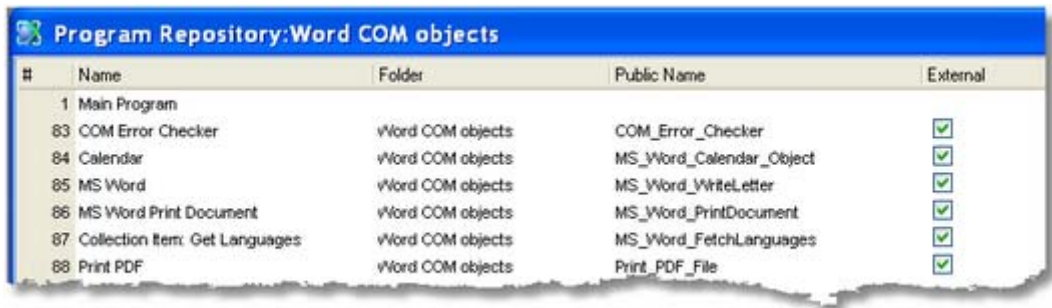
# Chapter 16: Components

## How do I Reuse eDeveloper Objects Across Projects?

The eDeveloper studio provides a very good ability to reuse objects within one project. By defining your models and data sources in their repositories, you can save time when writing your programs. However, you can take this a step further by creating a library of models, data sources, and programs that are reusable across many projects. Not only that, but you can upgrade the library without changing the project that uses it.

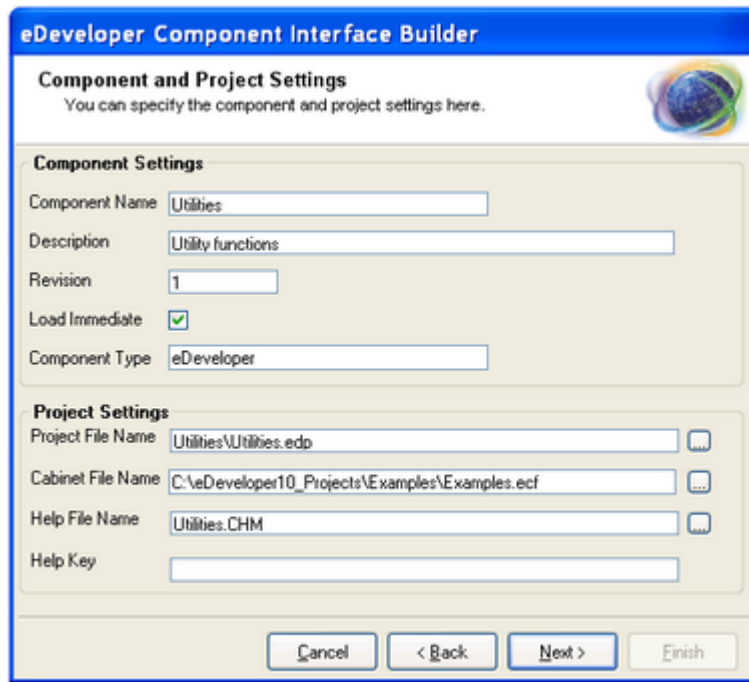
These libraries are called eDeveloper *components*. You can create a component from any eDeveloper project, and deploy the component as either an *.edp* or an *.ecf* file. You have total control over what parts of the application go into which component, so you can create different components with different abilities. This is particularly useful for commercial applications, where some features may be turned off for cheaper versions of the product.

### Creating an eDeveloper Component



1. Decide what objects you want exposed in the component. Give those objects public names.

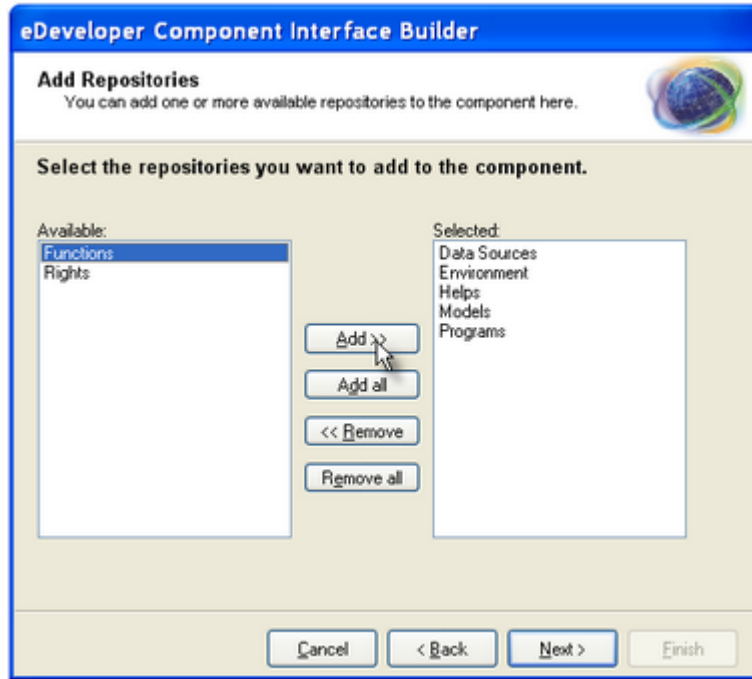
2. Select **Options->Interface Builder->eDeveloper**. The *eDeveloper Component Interface Builder* Wizard will appear. Press **Next**.
3. The next screen allows you to modify an existing component, or to create a new one. Press the **New** button to create a new component.



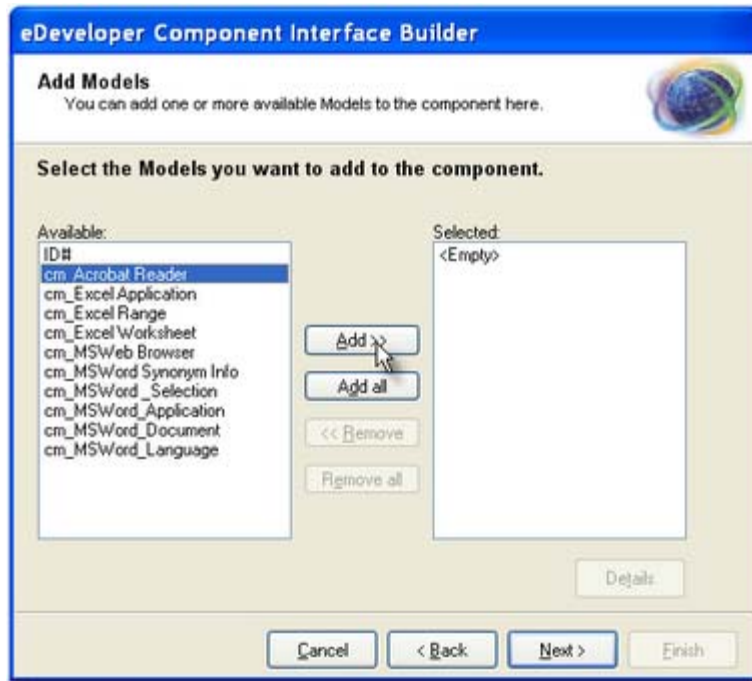
The image shows a screenshot of the 'eDeveloper Component Interface Builder' dialog box. The title bar is blue with the text 'eDeveloper Component Interface Builder'. Below the title bar, there is a section titled 'Component and Project Settings' with a subtitle 'You can specify the component and project settings here.' and a small globe icon. The dialog is divided into two main sections: 'Component Settings' and 'Project Settings'. In the 'Component Settings' section, there are fields for 'Component Name' (containing 'Utilities'), 'Description' (containing 'Utility functions'), 'Revision' (containing '1'), 'Load Immediate' (checked), and 'Component Type' (containing 'eDeveloper'). In the 'Project Settings' section, there are fields for 'Project File Name' (containing 'Utilities\Utilities.edp'), 'Cabinet File Name' (containing 'C:\eDeveloper10\_Projects\Examples\Examples.ecf'), 'Help File Name' (containing 'Utilities.CHM'), and 'Help Key' (empty). At the bottom of the dialog, there are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

4. You will now be on the *Component and Project Settings* dialog. There are a number of items you can specify here; press **F1** to get a more detailed description of them. You must, however, specify the *Component Name* and the *Project File name*. Press **Next**.

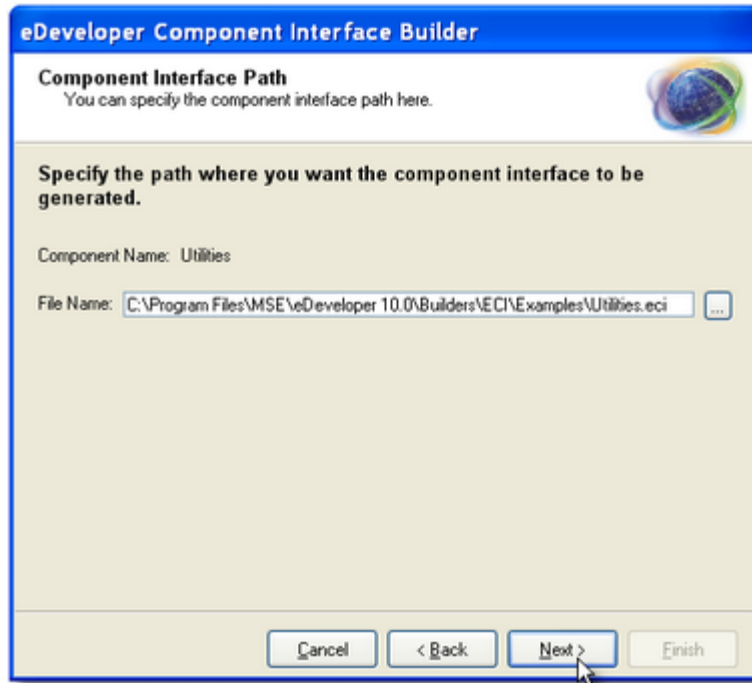




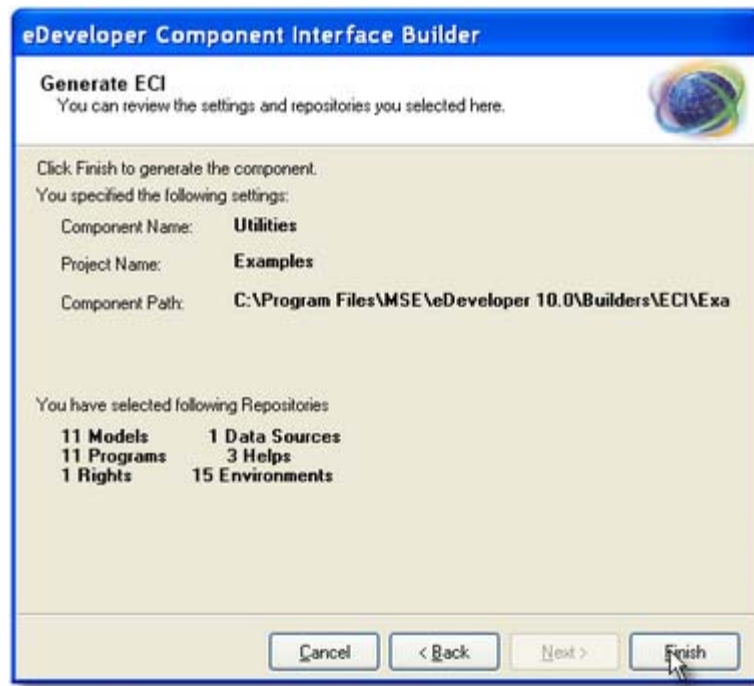
5. Now you will be on the *Add Repositories* dialog. Here you can decide which repositories you want to have visible in your component. Use the buttons to move items from the “Available” to the “Selected” column or the reverse. Then press **Next**



6. You will then be presented with a series of dialogs that allows you to select items from each of the repositories you selected. Select only the items you want to be visible in your component. Here we are adding the *Models* to the component.



7. Next you will be prompted for the path for the *.eci*, or eDeveloper Component Interface file. This is the file that will be used to bring the component in to the project that uses it. Press **Next**.



8. Finally, you will be presented with a screen that summarizes your choices. Press **Finish** to create your component, or use the **Back** button to make changes.

When the *.eci* file has been generated, you will get a “SUCCESS” message and your component is ready to use.

**Note:** The *.eci* file is a text file, and has a syntax similar to the *magic.ini* file. You can edit it manually without Interface builder, if you like.

Now, your component is ready for use.

**See also:** Chapter 16, “How do I Load a Component Into My Project?” on page 418.

## How do I Determine Which Data is Used by Your Component Builder?

When you are building a component, you have total control over what is and what is not revealed in the component. In order to be revealed in the component, an object has to meet the following criteria:

1. The object must have a *Public Name*.
2. If the object is a program, it must have the *External* column box checked.
3. You must select the item while you are building the component (as explained in Chapter 16, “Creating an eDeveloper Component” on page 411).

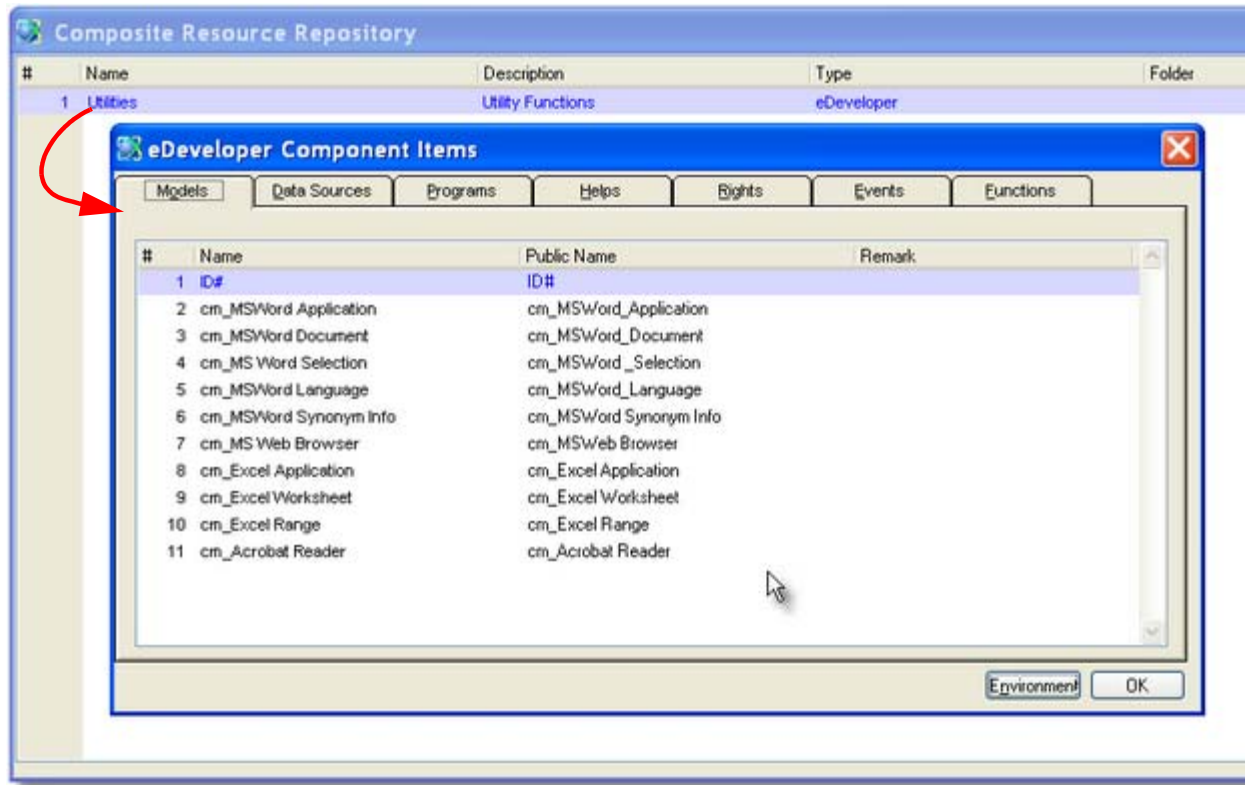
**See also:** Chapter 16, “How do I Reuse eDeveloper Objects Across Projects?” on page 411.

## How do I Load a Component Into My Project?

Before you can load a component into a project, you need to have created an eDeveloper Component Interface file, or *.eci*. This is a text file that is used to load the component into a project. You can see how to create the *.eci* file in Chapter 16, “Creating an eDeveloper Component” on page 411.

Once the *.eci* file is created, do the following steps to make the component available to your project.

### Using an eDeveloper Component



1. Select **Project->CRR (Shift+F7)**. The Composite Resource Repository will open.
2. Press **F4 (Edit->Create Line)** to open up a line. Type in a name for the component you are going to use. This name doesn't have to be the same as the name of the component: it is for documentation only.
3. **Zoom (F5 or double-click)** to select the *.eci* file that describes this component.
4. Now, when you **zoom** again, you will see a list of all the component items. There is one tab for each of the items in the component, plus a button at the bottom that will bring up the environment setup for the component.
5. You can **zoom** on each of the items in these lists to get more information about them, such as the Model properties or the arguments for a program. However, you cannot change any of the objects from within this project; you can only use them. All changes are done in the component.

Now, you can use any of these items inside your project just as you would use any other object. When you select, for instance, a **Model**, you will see the Models on this component list on the same list as the Models that are in that project.

**Note:** You don't need the *.eci* file at runtime. You do have to make sure the component is in the same relative location, but once the component is brought into the project, the *.eci* file is not used.

## How do I Implement Changes Done in Existing Component, Into a Host Application Using This Component?

Once you have created a component, and loaded it into your project, it is easy to implement changes to the component. There are two aspects to implementing changes. First, there is the case where you are moving a component into a runtime environment, where the host application is not being changed. Second, there is the case where you are using the component in the Studio, and will want to use whatever new items are in the component. Both cases are discussed below.

### Changing a runtime component

When you are making changes to a component that is being installed in a runtime environment, implementation is straightforward: just replace the old component with the new one. This kind of installation would happen when, for instance, you are fixing a bug in a component or making a routine run faster, or doing some other internal fix. As long as the names of the objects and their arguments don't change, you don't need to change the host.

If you add items to the component, the host won't pick up the new objects -- it didn't use the objects before, so it doesn't know about them now, so nothing will break. However, if you change the public name of an existing object, or change the parameters of a program, then that can cause a fatal error.

### Changing a component in the Studio

When changes are made to a component that you are working with in the Studio, however, you need to change the **.eci** file to reflect the changes, or you will not see the new objects. In this case, you need to do the following steps:

1. If the component is one you created, then generate a new **.eci** file. This is similar to the procedure described in Chapter 16, "Creating an eDeveloper Component" on page 411, except you will modify an existing component rather than creating it from scratch.

If you received the component from another party, then you should receive a new **.eci** file along with the new component.

2. Go to **Project->CRR (Shift+F7)** and position the cursor on the component you want to update.
3. Select **Options->Load/Reload Components**. You will be prompted for the name of the **.eci** file to use. Select the **.eci** file and press Open.

This will refresh the components list, and you can work with the new components.

**Hint:** Do not be tempted to delete the old component and reload it. If you do that, all references to the component objects will be lost. Load/Reload will refresh the references correctly.



### **Renaming objects in a component**

Within the eDeveloper Studio, the “name” of an object is not fixed, because the Studio references most items using an internal reference system. However, once you start making objects available to other programs, the references are done based on the actual text name, as is done on other programming tools.

So, there really is no graceful way to rename a component object once it is in use. If you rename an object and try to reload the component library, you will receive an “Item not available” message in the object name column, and all links to it will be broken.

Therefore it is recommended that when you design your components, you follow a strategy similar to that used in the development of COM libraries. Do not change the public names of your objects, do not change the arguments passed to a program, and do not delete objects. If you want to implement a new improved `Calendar` object, for instance, call the new one `Calendar2` and leave the original `Calendar` in place.

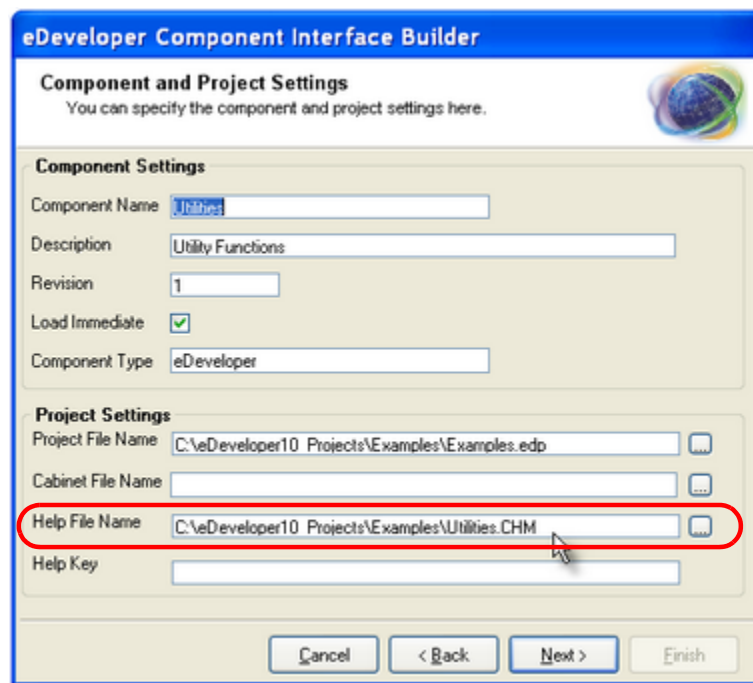
## How Can I Provide My Own Help File for a Component?

Providing your own help file for a component makes the component look professional, and also makes it easier to use.

Creating a Windows Help File requires getting a 3rd-party Help authoring tool. There are some good ones on the market for a very reasonable price. Once you have entered all your Help information, and compiled it, you will end up with a file ending in **.chm**, which is your Help file. Internally, that file has a numeric index to each Help item.

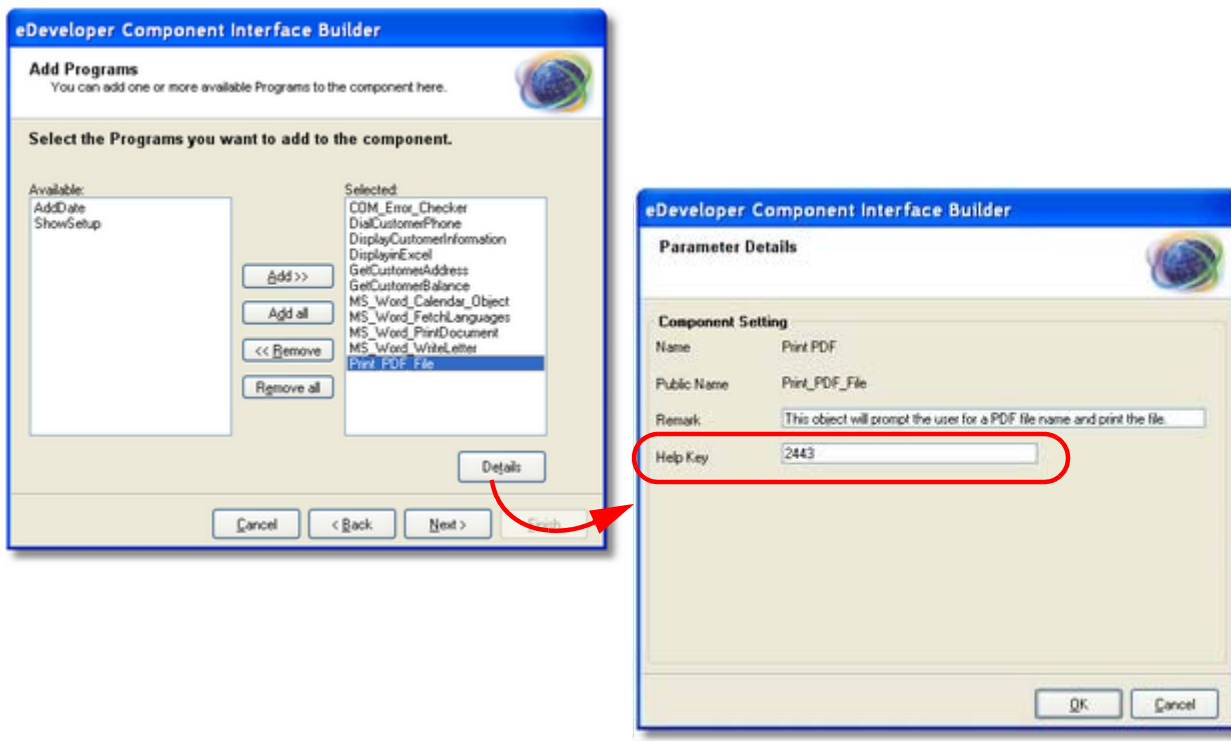
When you create your component, then, all that remains is to tell the component where that Help file resides, and which index to call under what circumstance.

### Implementing a Help file



1. On the Component and Project Settings screen, specify the name and location of the Help File that will be used.

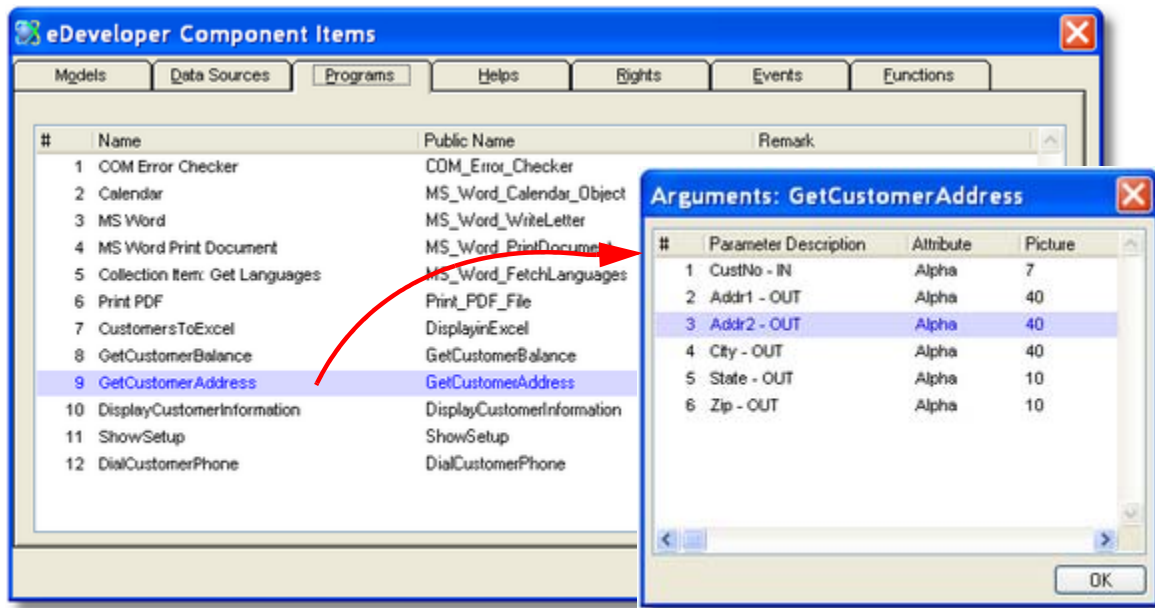
It is better to use a logical name here, such as **%WorkingDir%**, than to have the path hard-coded.



2. Continue through the wizard, until you reach the object for which you want to add a help reference.
3. Click the Details button for the object. Now you will see the parameter details, and a spot to add your *Help Key*. Enter the number of the Help item that pertains to this object.

Now, when this component is implemented, the end-user will see your customized help file when they press **F1** while using that object.

## How do I See Detailed Information About a Component's Objects?



When you are using a component in a project, you can view detailed information about the objects by zooming (**F5** or double-click) on each object. For instance, in this example, zooming from “Get Customer Address” brings up the list of arguments for that object.

Other details about how “Get Customer Address” works are not visible to the user of the component. A component is considered a “black box” and is designed to be invisible except for the pieces needed to be visible to implement it.

## How do I Access the Directory in which the Component Resides?

Task 6 - ProjectDir

Data ViewLogicForms

1	Main Source	0	No Main Source	Index:	0	
2	Virtual	1	ProjectDir()	Alpha	255	Init: 1 ProjectDir()
3	Virtual	2	%WorkingDir%	Alpha	255	Init: 2 Translate("%WorkingDir%")
4	Virtual	3	%TempDir%	Alpha	255	Init: 3 Translate("%TempDir%")
5	Virtual	4	%EngineDir%	Alpha	255	Init: 4 Translate("%EngineDir%")
6						

While a component is running, you can find out which directory it is running in by using the **ProjectDir()** function.

Note that this is a function, not a logical name. The logical names that give directory information, such as **%WorkingDir%** and **%TempDir%**, will be the same across all the loaded components, but the **ProjectDir()** is local to one component.

## How do I Determine if a Currently Running Application is a Component?

An eDeveloper project can be run as a host or as a component, depending on how it is implemented. So one project file can do double-duty. However, you can determine in which mode it is running by using the **IsComponent()** function.

Syntax: **IsComponent()**

Returns: TRUE if the task that is running is running as a component, FALSE otherwise.

## How do I Dynamically Call a Program Within Another Application not Defined as a Component?

While components are easy to use, you can call programs in another application without using components. There are several ways to do this, including implementing your called programs as COM objects or SOAP services. But the two most direct methods are using a *Call Remote*, or *Call by Name*.

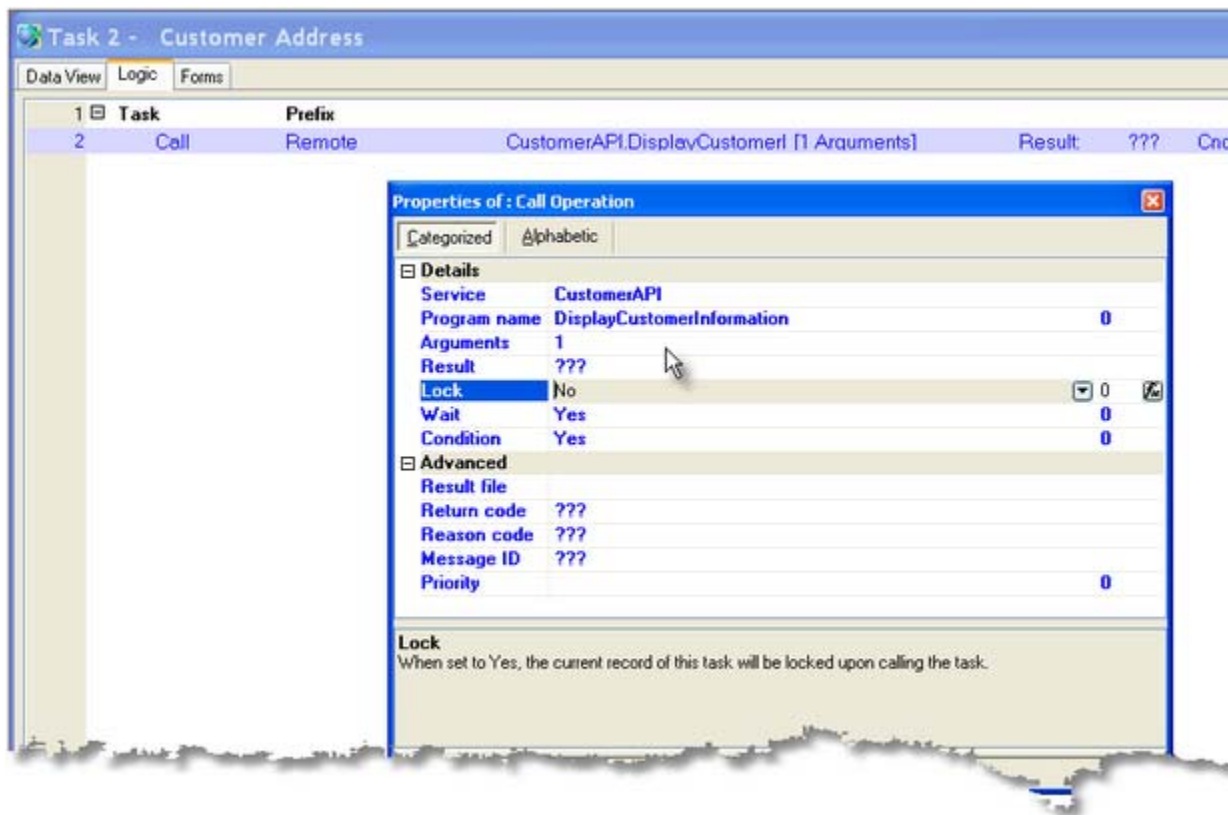
Note that when you use these types of calls, the eDeveloper Studio has no information about the object you are calling until runtime, so it is up to you to code the name and arguments correctly.

### Using Call Remote



1. First, set up a Service that points to the other application. That is done in **Options->Settings->Services**. Zoom from the *Endpoint* column to select the application you want to use.
2. Next, code your *Call Remote* operation.
  - Open up a line by pressing **F4**.
  - Type **C** for *Call*

- Type **R** for *Remote*

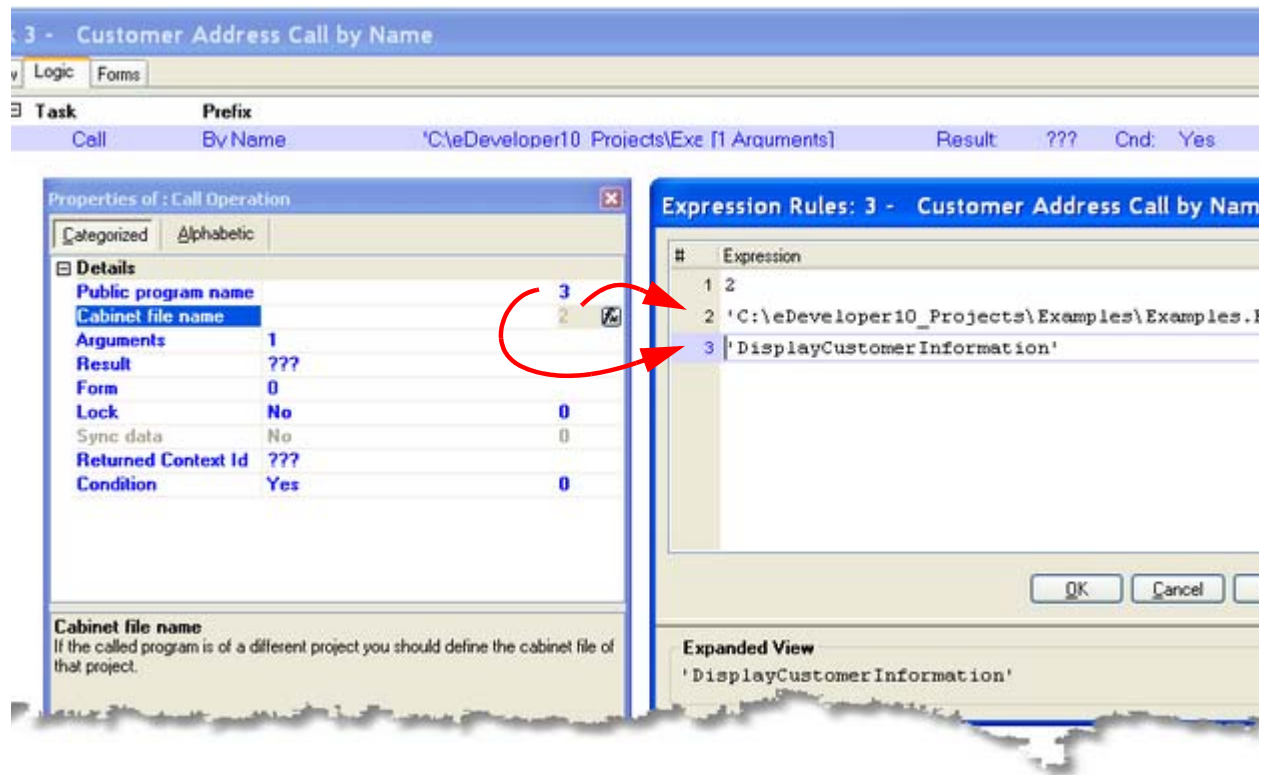


3. Go to the *Operation Properties* to enter the rest of the operation (**Alt+Enter**, or click on the *Properties* pane if it is open).
4. From the *Service* field, zoom to select the service that points to the application you want.
5. Type in the *Program name*. This should be the public name of the program within the application you are calling. It is up to you to type it correctly.
6. Zoom from the *Arguments* field to enter whatever arguments the program requires. Again, it is up to you to set them correctly, there is no automatic match-up as there is with a component.

Now, when the program runs, it will call the program from the other application.



## Using Call by Name



1. Enter the *Call By Name* operation:
  - Open up a line by pressing **F4**.
  - Type **C** for *Call*
  - Type **N** for *by Name*
2. Go to the *Operation Properties* (**Alt+Enter**, or click on the Properties pane if it is open).
3. Zoom from the *Public program name* to get to Expression rules. Type in the public name of the program you want to call.
4. Zoom from the *Cabinet file name* field to enter the name of the cabinet file that has the program.
5. Zoom from the *Arguments* field to enter the arguments you want to send and receive from this program.

That's all there is to it. The program will be called from the cabinet file you specified.

**Hint:** Although we show the path name typed in for clarity, it would be better to use a variable in the Main Program to hold the path name for all the components being used, and to use Logical Names.

## How do I Handle Recursive Calls Between Applications?

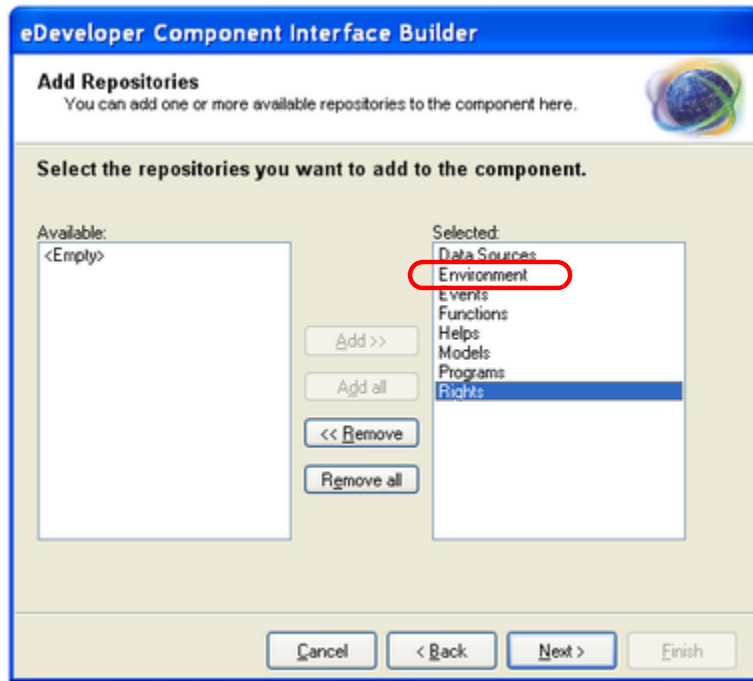


Recursive calls between components are not allowed. If you try to include a component object that makes a call into your current application, you will get a message “Item Not Available” on the component item. In our example, the program “Recursive Call” is a program that calls the Calendar program from the current application, so it cannot be included.

In general, it is a good idea to structure your applications so they don’t require this kind of recursive calling. For instance, you can have a library of utilities that are used between many applications, but the utilities would never call anything in those applications.

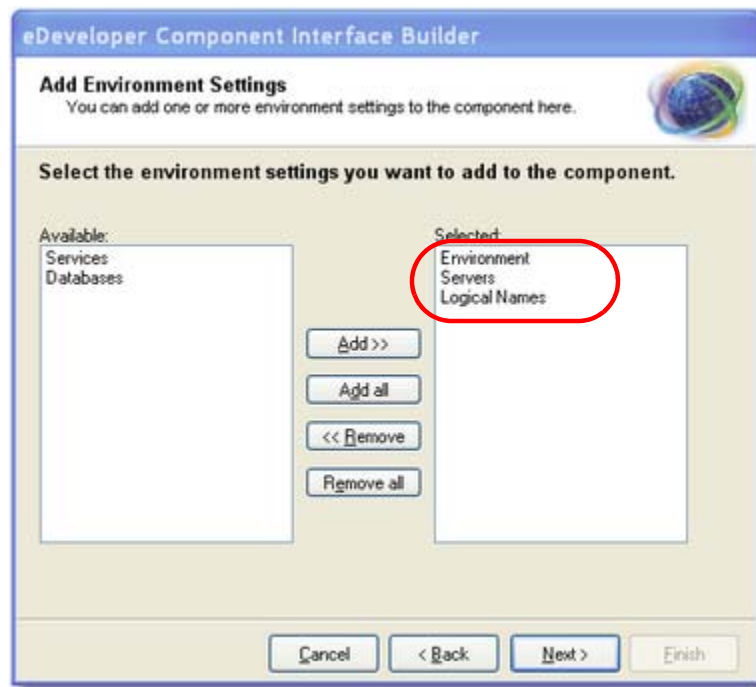
However, if you do need recursive calling, you can in fact implement it using *Call by Name* or *Call Remote*, as explained in Chapter 16, “How do I Dynamically Call a Program Within Another Application not Defined as a Component?” on page 427.

## How do I Implement Environmental Requirements for a Component?

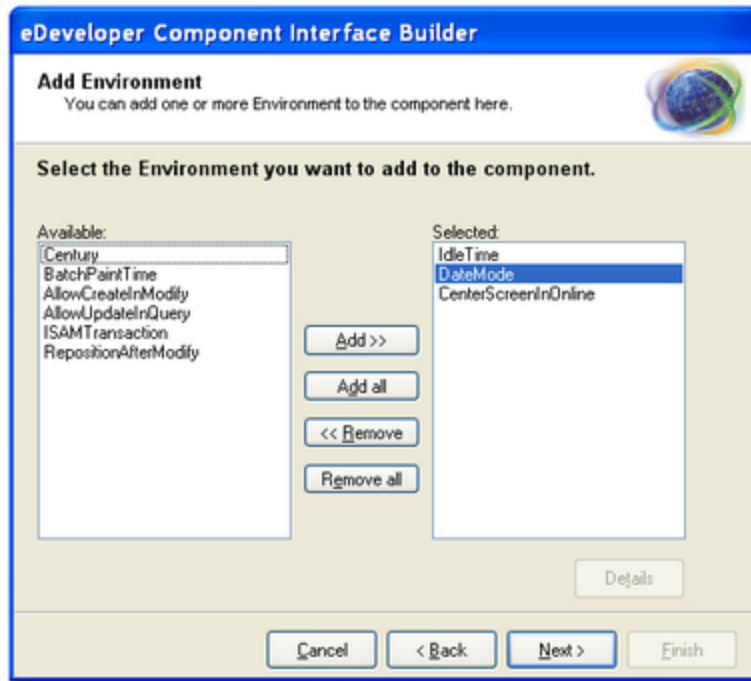


When you are creating a component, you can specify whether or not you want to specify the environment for that component. If you select the Environment repository, then you will see a series of screens that

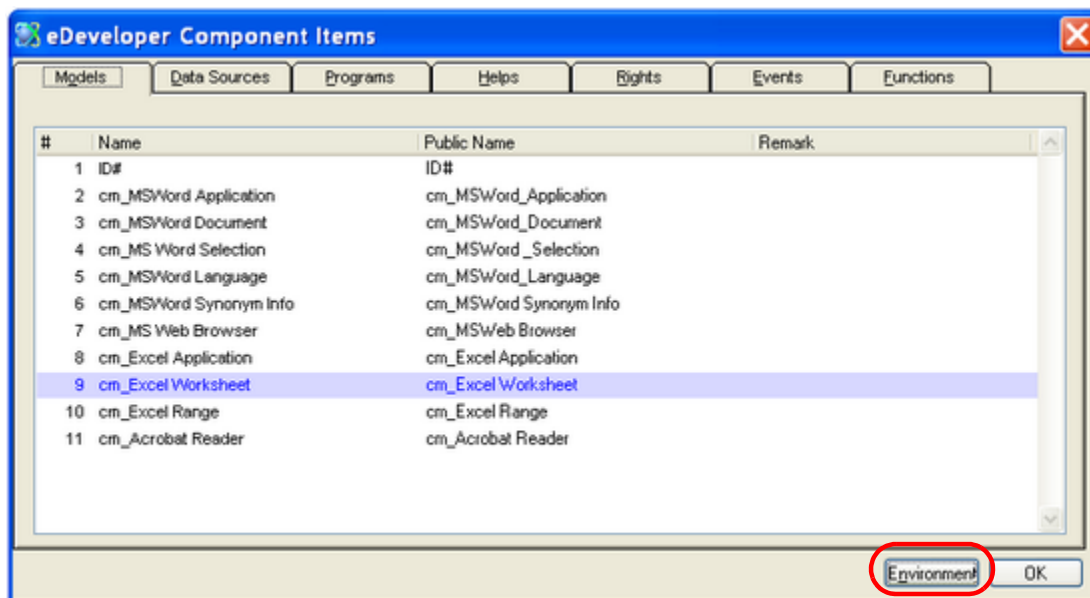
allow you to specify that individual aspects of the environment become part of the component.



In this instance, we are choosing to make the Environment, Servers, and Logical Names parts of the component be part of the component implementation. For each of these items, more dialog boxes will open,

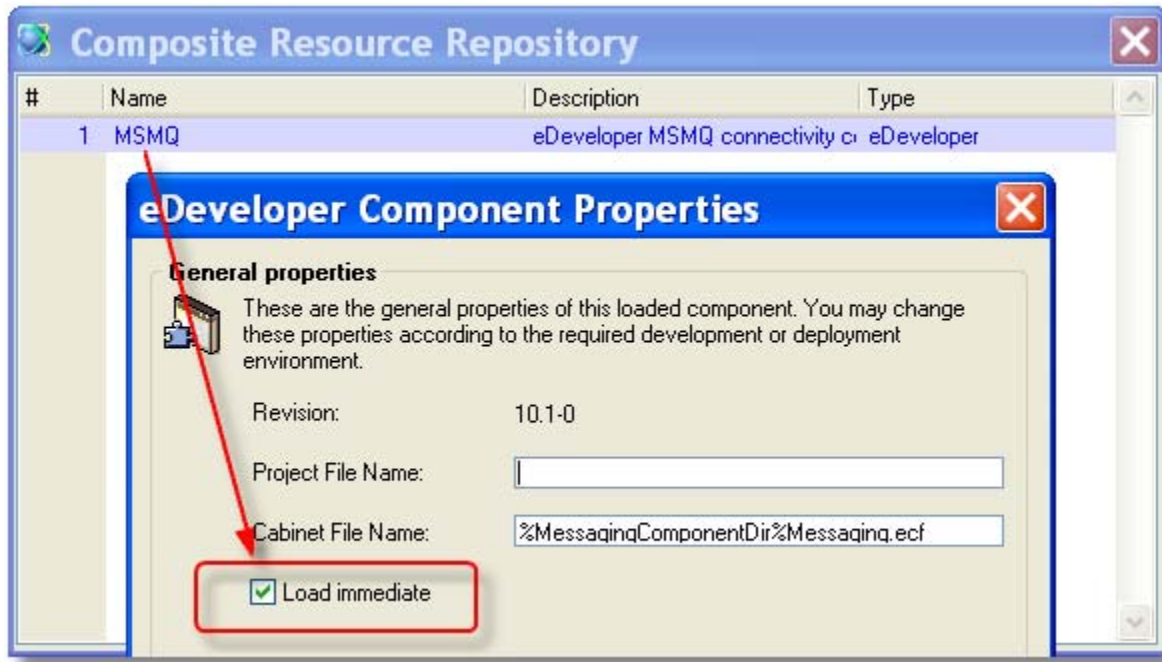


prompting us to select which items we want visible in the component.



When the component is used, these items will be visible when you press the Environment button.

## How do I Optimize Access to a Component?



Before a component can be run for the first time, it has to be loaded into memory. In order to ensure there is no delay the first time the component is run, you can specify to eDeveloper that the component be loaded when the project is loaded.

This is specified in two places. When a component is generated, the *Load immediate* flag can be specified in the *.eci*. This then becomes the default setting for the flag.

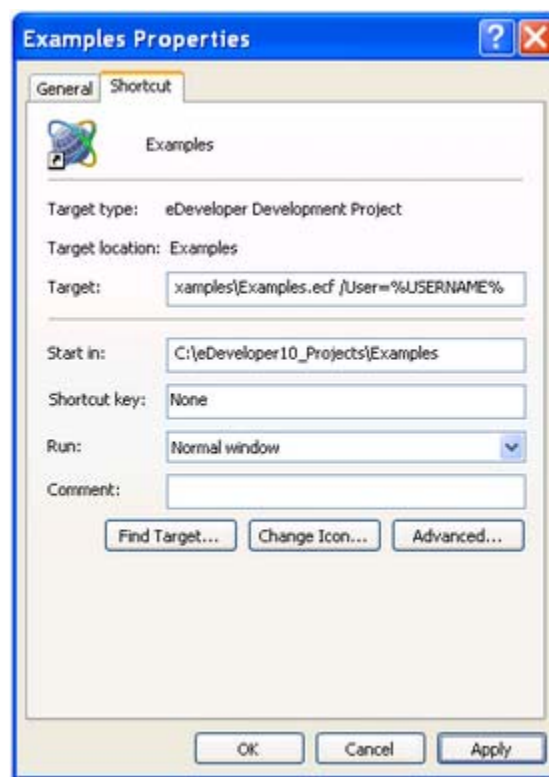
Then, in the Composite resource repository, you can override the default settings for each component, in the *Component properties* (**Alt+Enter**).

## Chapter 17: Environment

### How do I Use My Windows Login to Log on to eDeveloper?

eDeveloper has its own login screen, but some companies prefer to bypass it, and to use the Windows login to automatically log in to the eDeveloper application. This is easy to do and saves time for the user.

Once a user is logged in to Windows, the system variable %USERNAME% contains the user's login ID. This can be passed to eDeveloper at runtime, as well as other Magic.ini overrides, on the Target line of the shortcut.



## Using the Windows userid to log in to eDeveloper

1. Create your shortcut in Windows as you usually would.
  - For the *Target:* field, point to your *.ecf* file.
  - For the *Start in:* field, point to the directory you want to start in (typically the same directory as the *.ecf* file is in).
2. After the Target directory, add /USER=%USERNAME%. So in our example, the entire line reads:

```
C:\eDeveloper10_Projects\Examples\Examples.edp /User=%USERNAME%
```

3. Set up the userids in eDeveloper to match the Windows login ids. Make the password field blank.
4. In **Options->Settings->Environment->System**, set **Input Password** to No.
5. In **Options->Settings->Environment->System**, set **Allow Access to Logon** to No.

Now, the user can log in to eDeveloper automatically without entering a userid or password.

## Considerations

You need to be careful when using this feature, because if the user does get access to the logon dialog, they can easily log in as any other user. This can be an issue in environments where, say, Administrators can view sensitive data.

There are several methods to avoiding difficulties with this:

- Disallow any kind of login to eDeveloper except through the icon (as described above). This works unless a user happens to gain access to an open Administrator computer.
- Fill in the userid by passing it in, but set **Input Password** to Yes so the user has to enter the password.
- Force password entry for those users who access sensitive data. Using this method, the administrators would have a password entered on the eDeveloper userid, and a slightly different shortcut:

```
C:\eDeveloper10_Projects\Examples\Examples.ECF /InputPassword=Y
```



## How do I Automatically Have a Project Opened When Invoking the Studio?

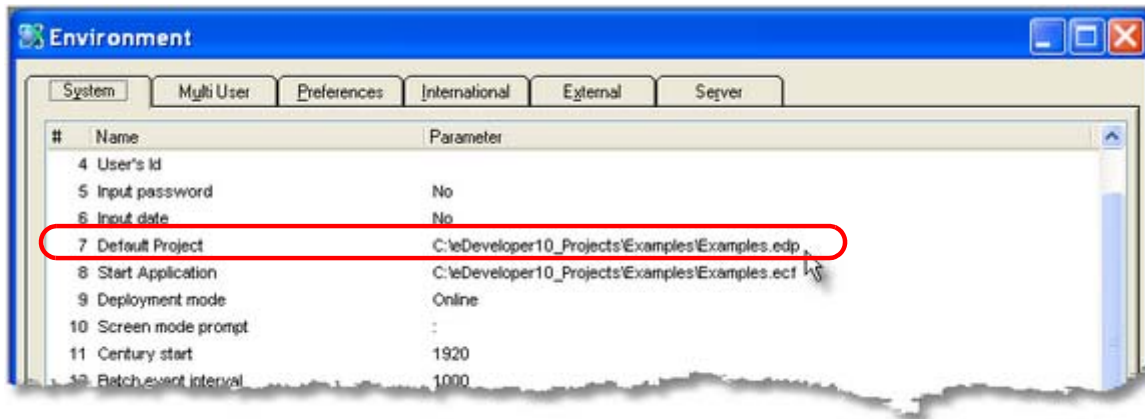
There are two basic ways to start the eDeveloper studio:

- By clicking on a project (**.edp** file), or a shortcut to that project,
- By invoking the studio directly, by selecting it from the Start menu or clicking on a shortcut to the Studio.

When you invoke the studio directly, it does not, by default, open any particular project. However, you can direct it to do so by changing the environment settings. The *Default Project* setting contains the full path name to the project that the Studio should open.

**Note:** The *Start Application* setting is very similar, except that it is used to invoke the *runtime* application (cabinet or **.ecf** file)

### Setting the Default Project



1. You can set the default project by typing in the path and filename in **Options->Environment->System->Default Project**, as shown above.
2. Alternatively, you can edit the Magic.ini in the eDev Studio directory, and change the `DefaultProject` line. So in this instance, we would change it to:

```
DefaultProject = C:\eDeveloper10_Projects\Examples\Examples.edp
```

Now, the next time the Studio is opened, it will open the specified project, which in this case is “Examples.edp”.

When you are developing for the Web, you do not need a website or any special software to test your application. The eDeveloper engine can act as a server engine. There are a few steps involved, which are outlined below. You need to have some basic idea of how web services work.

## Testing server applications

**Prerequisite:** Before you get started with working on web applications, you need to do the following:

1. Make sure you have IIS services running on the machine you are working on.
2. Make sure the scripts are installed (that eDeveloper was installed properly)
3. In the **Options->Environment->Server**, set *Activate as Enterprise Server* to Yes. Exit out of your project, then restart it, if you needed to change this flag.
4. Start the Magic Broker (**Start->Programs->eDeveloper->Broker->Start Broker**). (Depending on how you installed eDeveloper, however, it may start automatically when you start Windows.

Now, when you want to test a specific program, all you have to do is:

1. Position the cursor on the program you want to test.
2. Select **Debug->Run in Browser (Ctrl+Shift+F7)**.

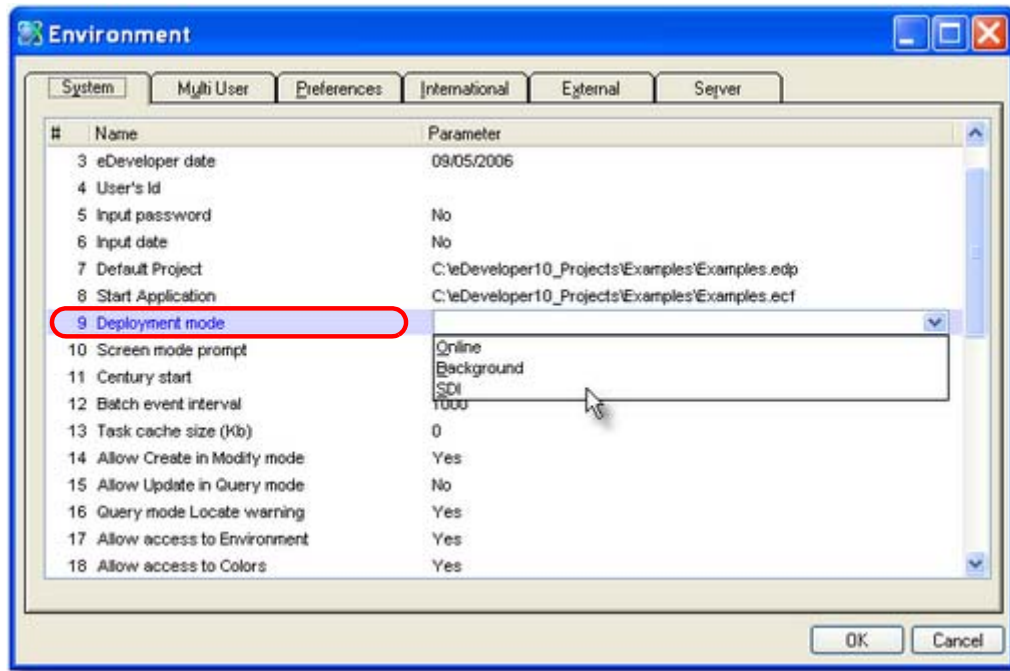
eDeveloper will automatically open up a Browser window and run your program in it. You don't need to set up a special link: it will be created automatically. Since you are running in Debug mode, you can have the activity monitor, variables, etc. open also, to help you in working on the application.

## How do I Force the Runtime Engine to Run its Application as an SDI Application?

A Single Document Interface application is one where each individual window has its own menu, task bar, and status bar. There is no “main” MDI background screen, and the application closes when the last SDI screen closes.

There is still a main context and a main program, but this is in the background and does not show to the user.

### Creating an SDI Application



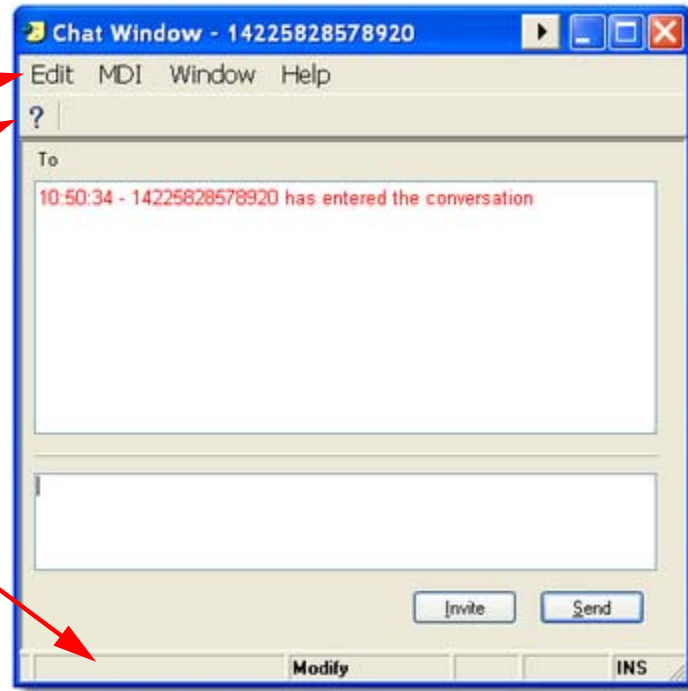
1. Set **Options->Environment->System->Deployment mode** to **SDI**.
2. Start the program in **Task Prefix**, depending on whether the context is Main or not. Otherwise it will be loaded multiple times.
3. Make sure that the Main Program has **Task Properties->Interface->Open Window** evaluated to No. Otherwise you will get an error.

**See also:** Chapter 17, “How do I run a program as an SDI program?” on page 440.

## How do I run a program as an SDI program?

An SDI, or Single Document Interface program is a program that has its own:

- Menu
- Toolbar
- Status bar
- Internet host bar (not shown)



It does not have a parent; its parent is the Desktop. It runs in its own context.

It is very easy to create SDI windows in eDeveloper. Just follow the steps below.

### Defining an SDI context

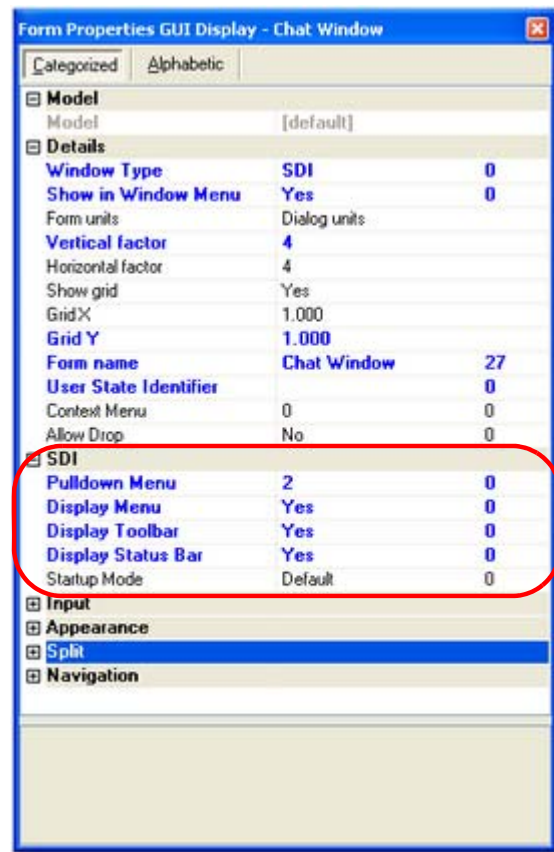
1. Set **Form Properties->Window Type=SDI**.
2. Check **Task Properties->Advanced->Parallel execution**.
3. Make sure that the Main Program has **Task Properties->Interface->Open Window** evaluated to No. Otherwise you will get an error.

That is all you need to do to define an SDI context.

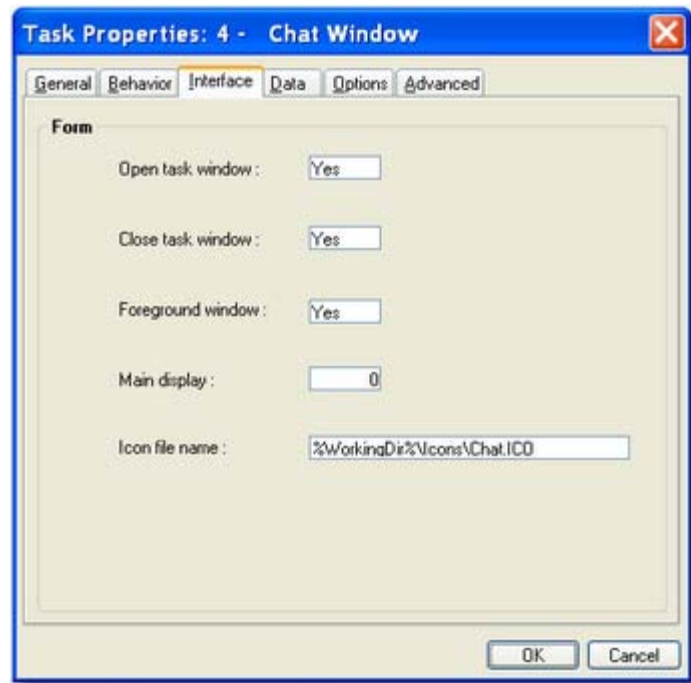
## Modifying the SDI context

1. In **Form Properties**: When the task is defined as an SDI, there is a new section in the Form Properties called **SDI**. Here you can define:
  - *Form name*: this becomes the frame title or caption.
  - *Pull down menu* and *Display menu* (Choose from the menus in the Menu Repository).
  - *Toolbar* (to display or not).
  - *Status bar* (to display or not).
  - *Startup mode* (Default, maximized, minimized). If the form is set to maximized, it will take up the entire desktop, because it is not confined to the eDeveloper frame.

Most of the rest of the Form Properties are still available and work as they would for any other form. The exception is the “Centered to MDI” startup position. This has no meaning, because there is no MDI, so it will be ignored.

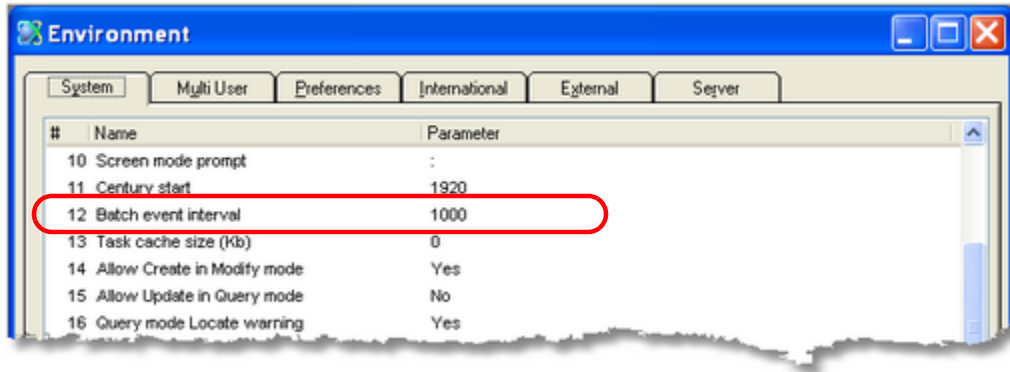


2. In **Task Properties** you can set the icon of the form. This shows in the upper left corner of the form, and also appears when using Windows **Alt+Tab**. If no value exists there, then the application icon will be used.



**See also:** Chapter 17, “How do I Force the Runtime Engine to Run its Application as an SDI Application?” on page 439

## How do I Specify the Screen Refresh Rate in a Batch Task?



When you are creating batch tasks, it is common to have a progress screen of some sort show to the user, so the user knows something is going on. This screen needs to refresh itself, so it can show records cycling or a progress bar moving. However, if the screen refreshes itself too often, this impacts the speed of the batch task. The ideal refresh rate often depends on the hardware that is running: newer, faster computers can handle more screen refreshes than older ones can.

So, you can set this feature at runtime using the **Batch event interval** environment setting. The Batch Event Interval is set in units of 1 millisecond, so a batch interval of 1000 means the screen will refresh once every second.

### Specifying the batch event interval

1. Go to **Options->Environment->System->Batch Interval**. Enter the number of milliseconds you want to use. Or,
2. Enter the desired number of milliseconds in the Magic.ini file, as the BatchPaintTime setting. Or,
3. Use ININPUT to change the BatchPaintTime setting temporarily. You can do this if you want to change the paint time for one batch program, for instance, and change it back when the program is finished.

## How do I Determine the Interval Being Used by the Engine to Poll Async Events?

When a batch task is running, the eDeveloper engine does not wait for input, it just processes records as fast as the system allows. It can poll for pending events though, and respond to those events when they happen. How often it checks for events depends on three settings:

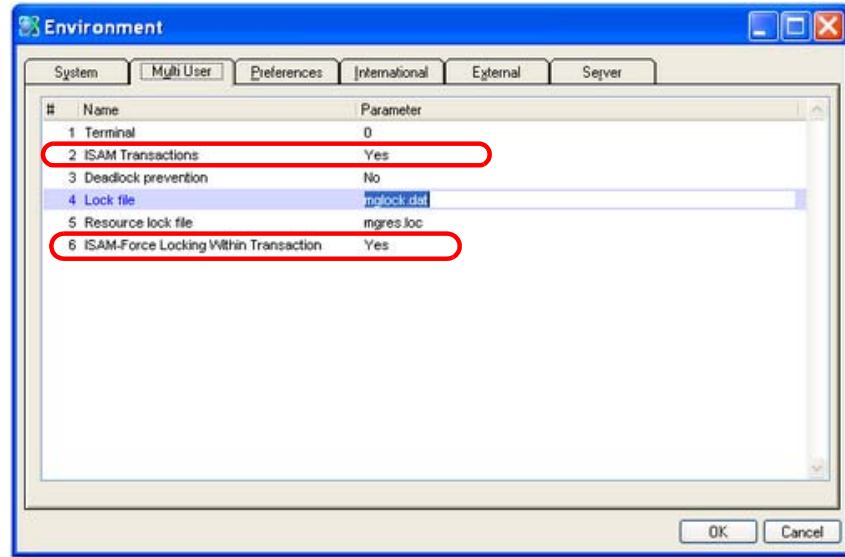
1. **Task Properties->Behavior->Allow Events:** If this is Yes or evaluates to TRUE, then the engine checks for events. Otherwise, it does not. An “event” includes pressing the keys, so if this is set to No then the user cannot cancel the batch task by pressing the escape key.
2. **Options->Environment->System->Batch Interval:** This affects all batch tasks in the application, but you can change it at runtime using ININPUT, if you need to.
  - **Zero:** Never poll based on time interval.
  - **N:** Poll every N milliseconds.
3. **Task Properties->Behavior->Record Event Interval:** This setting only effects the current task, and tells the engine to check for events based on numbers of records processed. You can set it to Yes or No, or use an expression to set it at runtime.
  - **Zero:** Never poll based on number of records
  - **N:** Poll every N records.

These settings work together. For instance:

- If **Allow Events** is set to No, there will be no polling for events, regardless of the other settings.
- If **Batch Interval** is 0 and Record Event Interval is 0, there will be no polling for events regardless of the Allow Events setting.
- If the **Batch Interval** is 300 and the **Record Event Interval** is 20, then the engine will poll for events every 300 milliseconds and every 20 records.



## How do I Implement Transactions with ISAM Files?



When you are using an ISAM DBMS, eDeveloper will support transactions similarly to how it does for a SQL DBMS. That is, when you set **Task Properties->Data->Transaction Mode** and **Transaction Begin**, the proper transaction requests will be sent to the ISAM DBMS.

However, with ISAM files you also have the option of turning the ISAM transactions on and off globally. That is, the transactions can be turned on in some tasks, but turned off for the entire application and the task transactions setting will be ignored.

### ISAM Transactions

1. To turn ISAM Transactions on globally, set **Options->Environment->Multi-User->ISAM Transactions** to Yes.
2. Alternatively, you can change this directly in the Magic.ini file, as

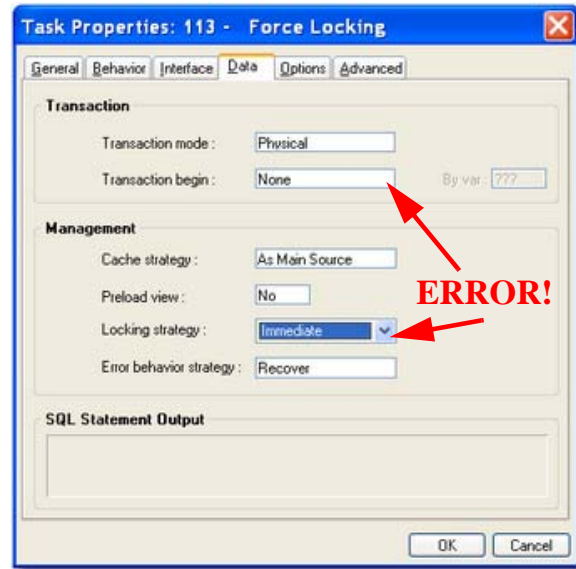
```
ISAMTransaction = Y
```

## ISAM-Force Locking Within Transaction

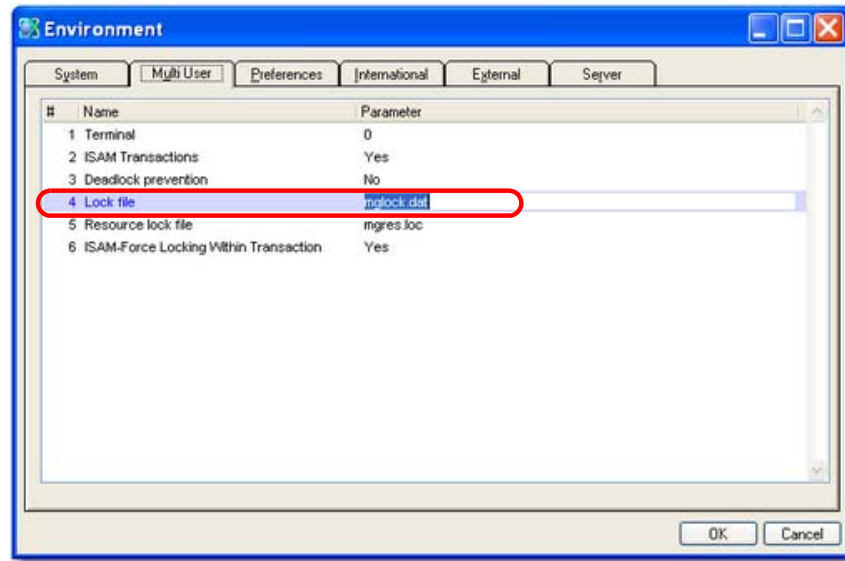
If you have ISAM Transactions turned on, then you have the additional option of setting Force Locking Within Transactions on or off.

If **Force Locking Within Transactions** is Yes, then whenever you specify a Locking Strategy that isn't "None" (Immediate, On Modify, Before Update), you *must* enter a Transaction begin that isn't "None". In other words, you can't lock the record unless there is a transaction active.

If you do set a locking strategy but no transaction begin, then you get a syntax error when you check the program.



## How do I Determine the Location of the eDeveloper Locking File?



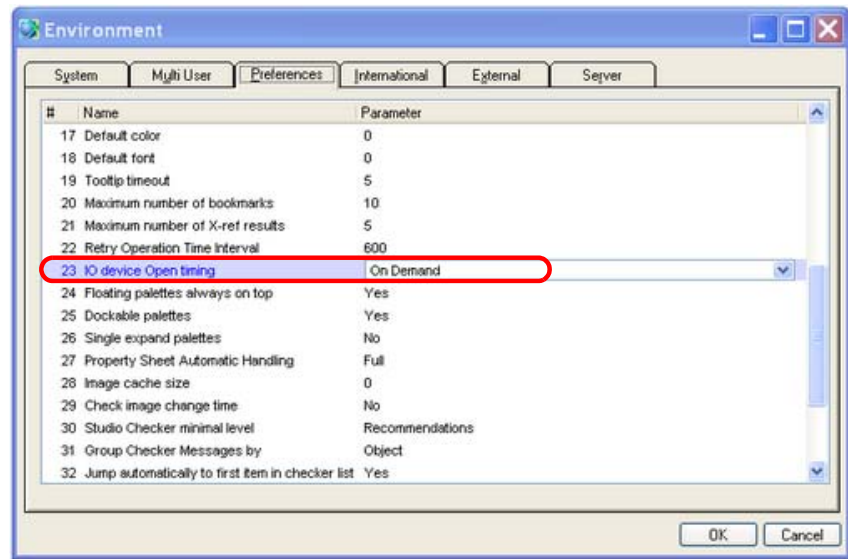
When eDeveloper locks a record, it uses a lock file to keep track of that lock.

Then name of the lock file is specified in the Magic.ini. You can view it or change it under **Options->Environment->Multi-User->Lock File**.

The path of the lock file is not specified there, however. The lock file will be located in the directory of the file being locked. That location could be specified in the Location column of the Database specification, or in the Data Source Name column in the Data Repository.

**See also:** Chapter 18, “How do I Create a Database Table Using eDeveloper?” on page 457.

## How do I Prevent the Creation of an I/O File Until it is Actually Used?

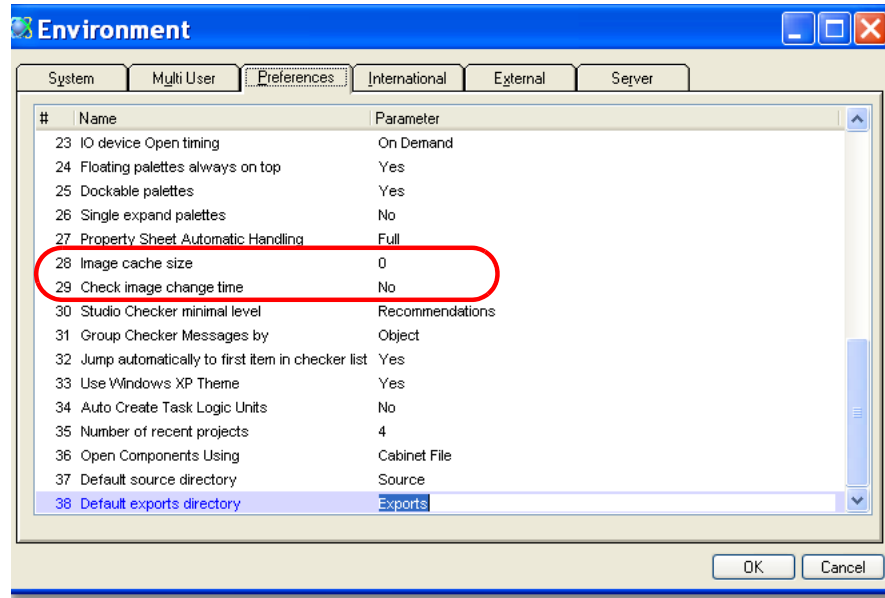


As soon as an I/O file is opened, a new file or print job is created. However, since files are opened before a task actually begins processing, that means that a task which does not end up creating any output still ends up creating an empty I/O file. This can be particularly irritating in certain report setups, because a blank sheet of paper ends up being printed.

You can prevent this by using **Options->Environment->Preferences->IO device Open Timing**. If you set this to *On Demand*, then the I/O file will not be opened until there is something to output. In other words, it will be opened as soon as there is an Output Form operation. If it is set to *Immediate*, then the I/O file will be created as soon as the task that declares it is opened.

**Hint:** You can further optimize your report jobs by being careful where you put your report header output operations. If these operations are in Task Prefix, then they will output a form before any records are processed, which may again result in an empty report. However, if you put them in a Group prefix or have them print automatically as Page Headers, then they will not print until there is some record being processed.

## How do I Optimize Image Access During Runtime?



Access to a lot of graphic images can slow down processing. One way to avoid this is to use caching, so if the same image is accessed over and over, it can be used from a memory cache.

You can control how the image cache works with two Environment settings, **Image cache size** and **Check image change time**.

### Image Cache Size

This setting lets you control the maximum number of kilobytes of memory used for caching displayed images. A value of zero means there is no limit to the cache size. If the image size goes over the Image Cache Size, then the least used images are removed.

### Check image change time

When this is set to Yes, then eDeveloper can detect if the image has changed since it was last used, and pick up the new copy. This is a feature you may need to evaluate carefully. The processing is faster when you set this to No, because the timestamp doesn't need to be fetched from disk. However, if in fact the image does change dynamically while you are processing, you will want to set this to Yes so you don't pick up an older copy of the image.

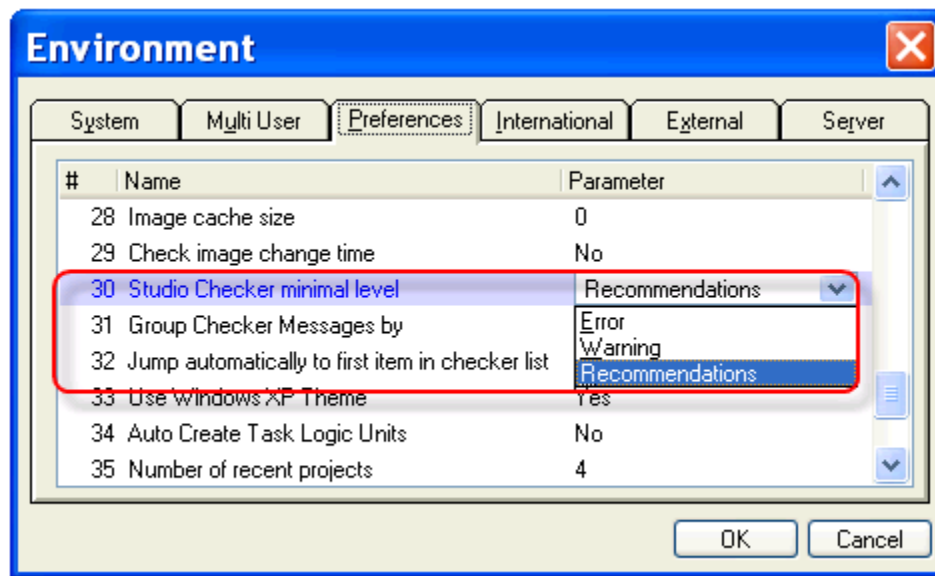
## How do I Control the Displayed Checker Messages?

eDeveloper has a good syntax checker, which will tell you if an object has any problems. You can run the checker on one object by pressing **F8** (**Options->Check Syntax**) while on that object. You can also check an entire object repository from your current position to the end by pressing **Alt+F8** (**Options->Check to end**).

The errors, if any, will be displayed on a separate pane called the Checker Result pane. You can control several aspects of how these messages are displayed.

### Studio Checker Minimal Level

Options->Environment->Preferences->Studio Checker minimal level



This setting determines what level of severity you want to see in your checker messages.

- *Recommendations*: displays Recommendations, Warnings, and Errors.
- *Warning*: Displays Warnings and Errors.
- *Error*: Displays Errors only.

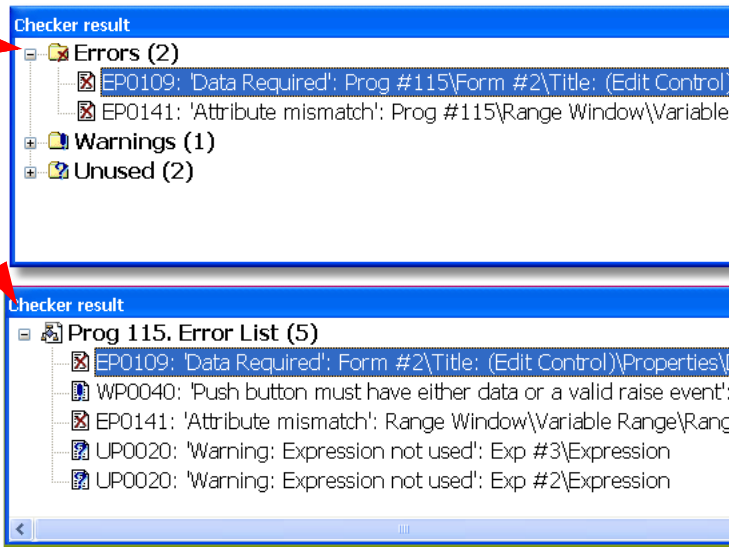
### Group Checker Messages by

Options->Environment->Preferences->Group Checker Messages by

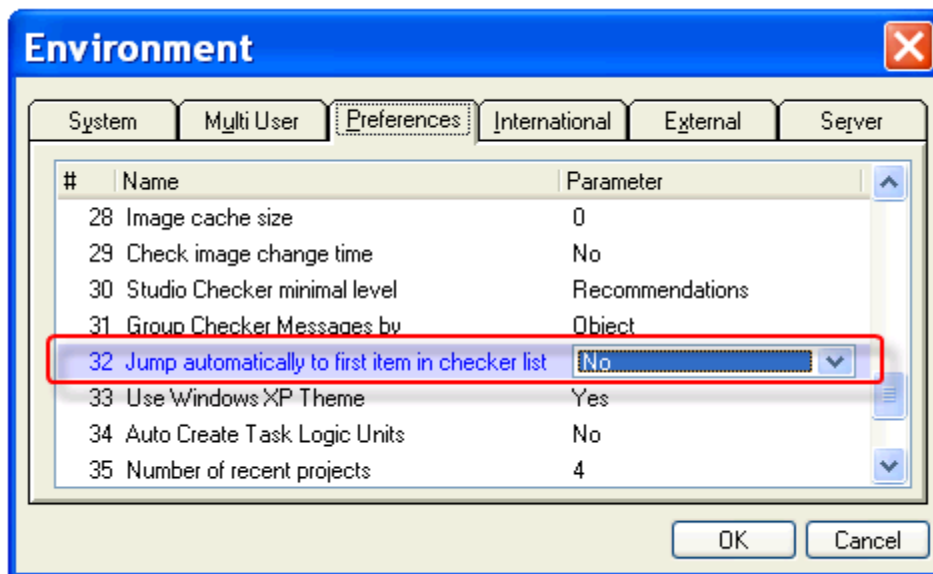
This option determines the grouping of the checker messages.

- *By Type* lists the messages in order of severity, with the most severe problems at the top of the list.
- *By Object* lists the messages in order of the object being checked.
- *By Type and Object* provides both types of lists, one after the other.

For any of these lists, zooming (F5 or double click) on the message will bring you into the code that the message refers to, which makes fixing the error fast and easy.



### Jump Automatically to the first item in checker list



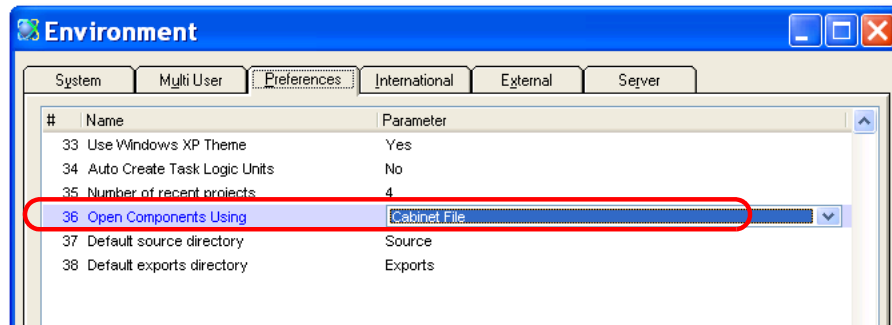
If this is set to Yes, then the checker will automatically jump to the first item in the checker list, open up the object and position on the part of the object that is in error.

**Allow Access to Checker Messages**

- Yes: Allows the developer to access the checker messages.

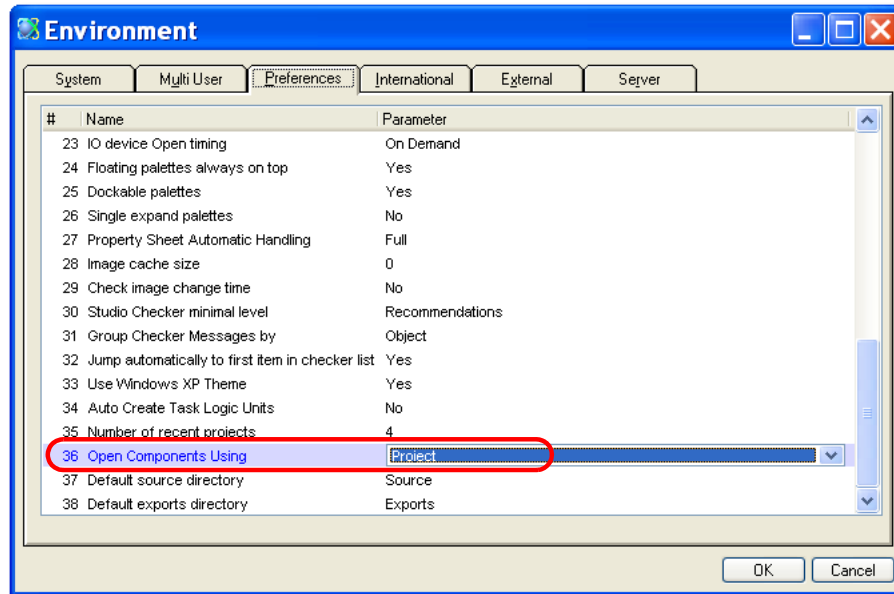


## How do I Develop an Application Using Components Without the Need to Create a Cabinet File for Each Component?

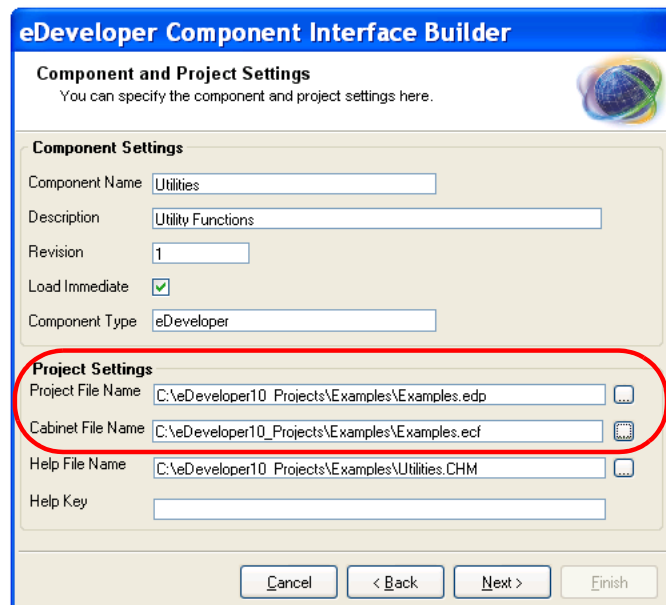


When you are working in the Studio with your own Components, you have the option of using those components in either the Project File (*.edp*) form or the Cabinet File (*.edf*) form. Working with the Project File is often preferable, because you can make changes easily to your Component while you are working with it.

## Running a Component from a Project File

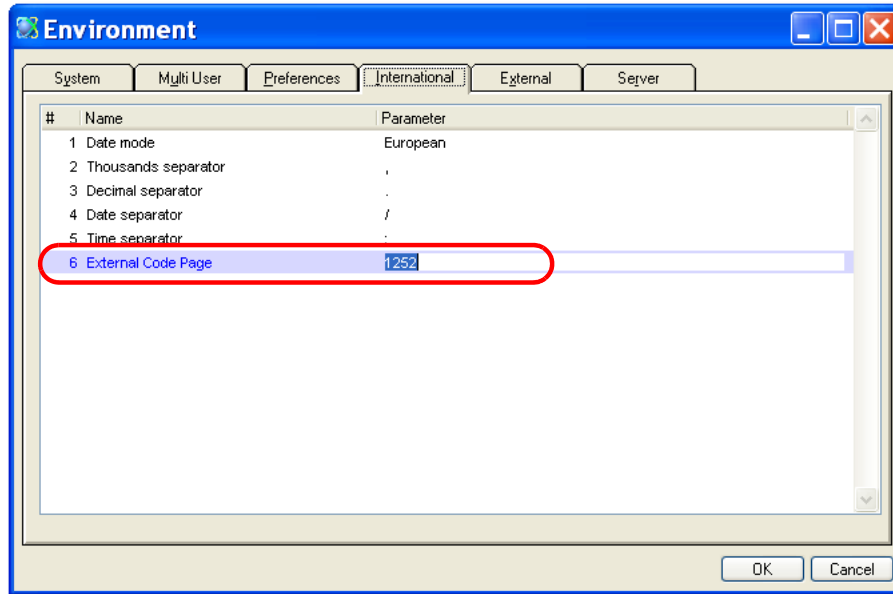


1. Set **Options->Environment->Preferences->Open Components Using** to **Project File**.



2. This will open the *Project File Name* that was specified when you created the component.

## How do I Determine the Codepage for Translation of ANSI Strings to Unicode and Vice Versa?



Translations of code from Unicode to Ansi and vice versa are done using Codepages. By default, eDeveloper uses the code page of the operating system. However, you can set it in **Options->Environment->International->External Code Page**.

The **CodePage()** function also will change the code page that is currently being used.



---

## Chapter 18: Defining Data Sources

---

### How do I Create a Database Table Using eDeveloper?

It is very simple to create a database table in eDeveloper. All you need to do is specify the columns in the table, and eDeveloper will handle the details of creating the table in the DBMS. Furthermore, the process is basically the same no matter what kind of data source you are creating, whether it is an ISAM file, an SQL table, or a memory table. Even XML file definitions follow the same basic format.

There are two basic processes for working with SQL data sources in eDeveloper:

1. The table does not exist, and we want to create it in eDeveloper and in the SQL database
2. The table already exists in an SQL database, and we want to bring it into eDeveloper

The second case is covered in Chapter 18, “How do I Access an Existing Database Table?” on page 464. The first case is covered here.

Here is a summary of the basic steps. We’ll go into them in more detail below.

- Make sure the Gateways and DBMS are loaded
- **Set up the Database definition:** Load the gateway for the DBMS, set up the database definition.
- **Create the table:** Set up the table in the Data Source Repository.
- **Create the columns:** Create definitions for each of the columns you want in the table.
- **Create the indexes:** Create definitions for each of the indexes you want in the table.
- **Syntax check the table:** Use the eDeveloper syntax checker to check for errors (F8).
- **Test the table by creating a few records:** Generate a simple Browse program (**Ctrl+G**) to check that it works.

If the database definition is set up correctly, eDeveloper will automatically take care of the details of creating the SQL table definition.

**Note:** While these instructions are slanted toward SQL tables, creating the table as a memory table or ISAM file is almost the same. The differences have to do with setting up the database definition, and the fact that ISAM files and memory tables don't have the same naming constraints.

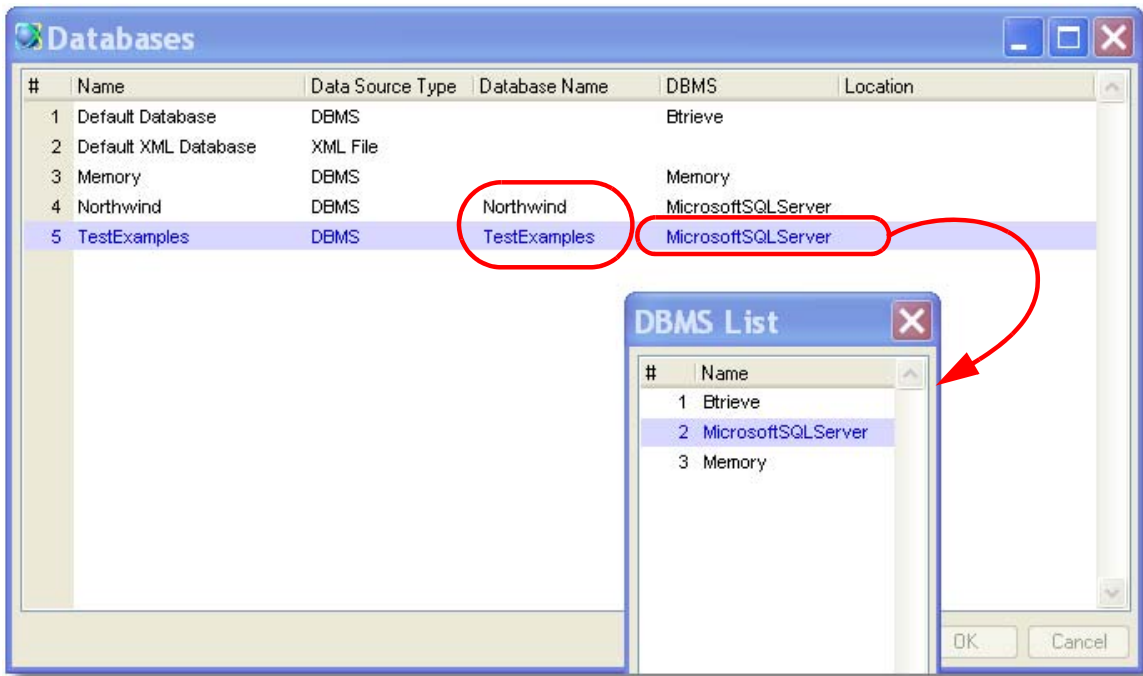
### Make sure the gateways and DBMS are loaded

```
201 [MAGIC_GATEWAYS]
202 ;MGCOMMO1=mgwsock.dll
203 MGDB00=C:\Program Files\MSE\eDeveloper 10.0\Gateways\MGBtrieve.dll
204 ;MGDB01=MGpervasiveSQL.dll
205 ;MGDB03=MGMySQL.dll
206 ;MGDB06=mgdb2400.DLL
207 ;MGDB13=mgOracle.dll
208 ;MGDB16=mgac32.dll
209 ;MGDB18=mgdb2.DLL
210 ;MGDB19=mgodbc.dll
211 MGDB20=C:\Program Files\MSE\eDeveloper 10.0\Gateways\mgmssql.dll
212 MGDB21=C:\Program Files\MSE\eDeveloper 10.0\Gateways\mgmemory.dll
213
```

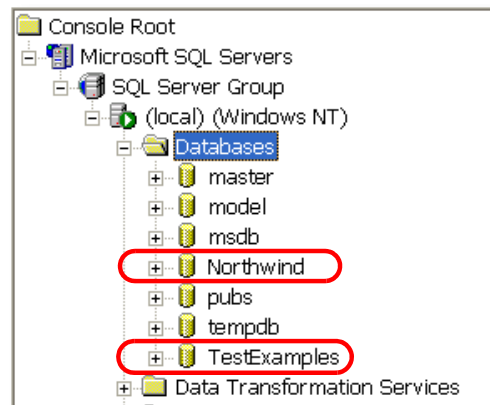
1. Before you can set up the database, you have to make sure the drivers are loaded. This is set up in the Magic.ini. When eDeveloper installs, the MAGIC\_GATEWAY section is set up automatically depending on what Gateways you chose during installation.
2. If you are accessing a new DBMS type, it's a good idea to check the Magic.ini, and the installation directory, to make sure these are installed correctly.
3. Also, of course, the actual DBMS client must be installed on the machine you are working on. In this example, we are accessing the Pervasive ISAM database, and the MS SQL database, so both of those products must be installed on our machine, and running when we start eDeveloper.
4. If you had to add the gateway in this step, then you will need to close and restart eDeveloper before you go to the next step.

### Set up the Database definition

The Database definition is the key to how the data source specification is used. In fact, one data source specification can be repeated and used with several database definitions, so that the same data source could be a memory table, and SQL table, or an ISAM table, depending on which database definition it points to.



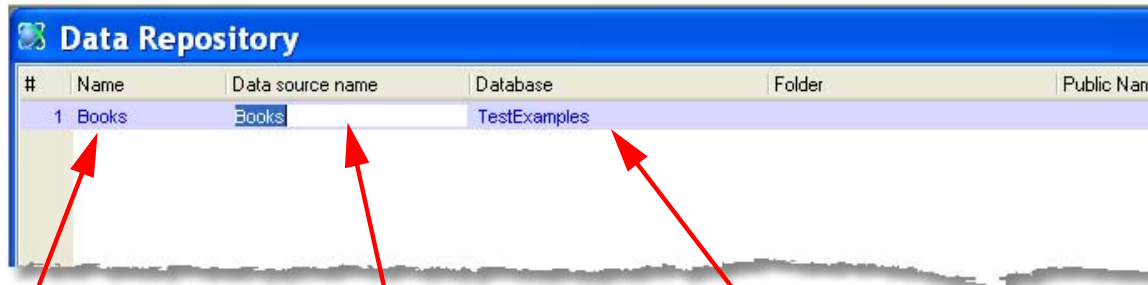
1. With no project open, go to **Options->Settings->Databases**. The DBMS you are using should show up on the DBMS list. Select the DBMS you want to use. Our example uses the Microsoft SQL Server.
2. Next, you need to choose the Database name. This is the name of the database in the DBMS. In our example we use two databases, Northwind and TestExamples.
3. While eDeveloper can create tables in the database, you have to create the databases themselves in the DBMS manager (or use existing ones).
4. Press **Alt+Enter (Edit->Properties)** to set up the user id and password as needed. Also, if you are going to create tables within the database from within eDeveloper, then you need to set **Properties->Options->Change Tables in Studio** to Yes.



### Create the table

Your next step is to create the actual table definition. While you do this, you need to keep in mind the constraints of the particular DBMS you are using. eDeveloper does not have many constraints, so while you are using memory tables you don't have to think much about such things as naming conventions. SQL, however, has constraints about what characters are allowed.

1. Open your eDeveloper project.
2. Go to the **Data Repository** (**Shift+F2**) and open up a line (**F4**). Here is where you create your table.



Give the table any *name* you like.

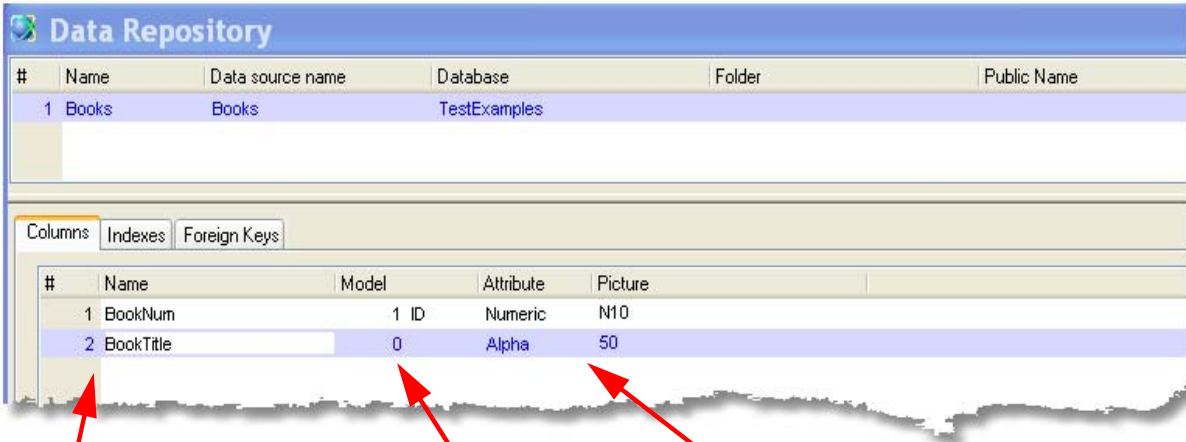
The *Data source name* will default to the name, but you can change it. This will be the actual table name in the DBMS database.

Zoom to select the *database* from your database list.



### Create the columns

Click on the bottom area, on the Column tab, to create your columns. For each column, use **F4** to open up a line, then enter the data as shown below. Each line will show the basics for that column. More details are shown in the Properties pane (**Alt+Enter**) for each line. Setting up the column definition is very much like setting up the variables in a program.



Give each column a name. The name entered here isn't used directly in SQL, but it's good practice to use the same name as will be in the SQL table. That means conforming to the SQL rules.

Select a model, if you want, to do the basic field definition. This is not required, but it is a good idea and will save you time later.

If you are not using a model, set the Attribute and picture for the column.

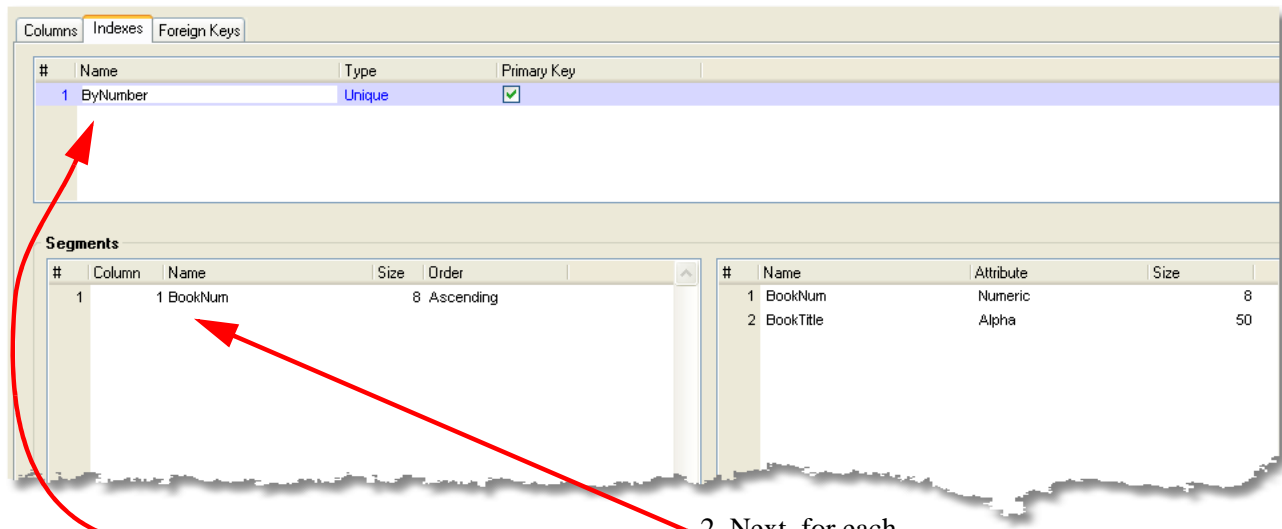
Next, you need to set up the more detailed definition of this field in the *Property Pane* (**Alt+Enter**). For most fields, the defaults will work (or you will have set them up in your field model). However, this is the section where you can indicate exactly how the data will be store in the SQL database.

An important thing to keep in mind here is that there are two descriptions for each bit of data. eDeveloper keeps the eDeveloper data attributes simple and generic (Alpha, Numeric, Date, Time, etc.). But you also have control over the DBMS-specific definition for the data (ZString, LString, Integer, Float, etc.) and the actual SQL definition (CHAR(50), INTEGER). By default, eDeveloper will choose the DBMS definition that seems to make the most sense, but you can override this if you like. This is explained in more detail in Chapter 18, "How do I Define the Mapping Between an eDeveloper Field and a Database Column?" on page 469.

**Hint:** It is always best to use Models to encapsulate most of your data definitions. This allows you to set the SQL defaults in one place.

## Create the indexes

Next, you will need to create the indexes. Do this in the bottom area of the screen by clicking on the Indexes tab.



1. For each index, press F4 to open up a line. Give the index a name, and indicate if it is unique, and if it is a primary key. You can set more detailed information in the Properties Pane (**Alt+Enter**).

2. Next, for each index, press F4 to create segments. Zoom to the area on the right to select columns that will be part of the segment, and indicate whether that segment will be Ascending or Descending. Again, you can set more detailed information in the Properties Pane (**Alt+Enter**).

## Syntax check the table

After you have your columns and indexes entered, use the syntax checker (**F8**, or **Options->Check Syntax**) to make sure you don't have any major errors.

1. Position the cursor on the table you want to check.
2. Press **F8**.

The errors, if any, will appear in the Checker pane.

## Test the Table by creating a few records

Last, make sure you can create some records. The easiest way to do this is to use the *Program Generator* utility. You can do this easily:

1. Position the cursor on the table you want to check.
2. Press **Ctrl+G** (**Options->Generate Program**).
3. Press **OK**.

A browser window should appear. You should be able to view, modify, and add records from within the browser. If you can do that, then you know the table is ok. You will be able to view the table you just created in the SQL DBMS using the DBMS tools.



Here is our table in an eDeveloper Browse program.



And here is the same table in MSSQL.

Once you have the table created in your DBMS, keep in mind that you need to keep the definition in the DBMS and the definition in eDeveloper synchronized. It is best to always maintain the definition in one place or the other. If you change the definition in eDeveloper, the eDeveloper will automatically reconfigure the table in the DBMS, if **Properties->Options->Change Tables in Studio** is set to Yes for that database.

## How do I Access an Existing Database Table?

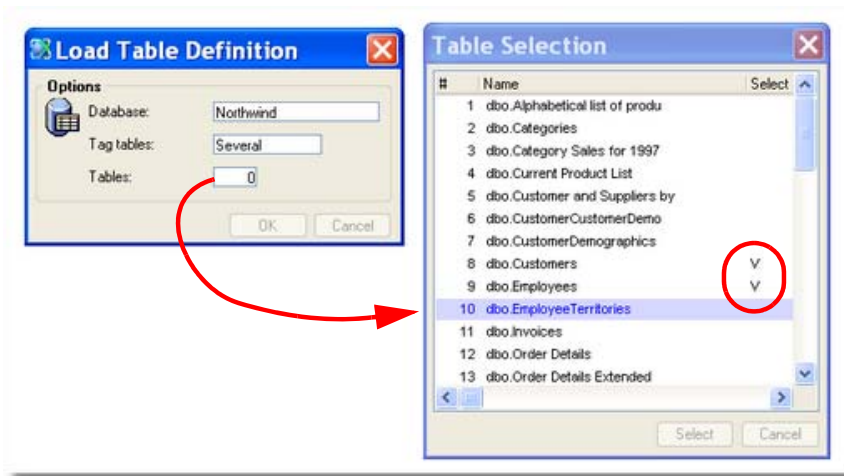
If a database table already exists in the DBMS, it is a simple matter to bring it in to your eDeveloper application.

**Prerequisite:** You need to:

- Have the DBMS installed and running,
- Have the gateways for that DBMS installed
- Have a database definition defined for that DBMS

These are covered in more depth in Chapter 18, “Make sure the gateways and DBMS are loaded” on page 458 and Chapter 18, “Set up the Database definition” on page 459.

### Accessing an existing database table

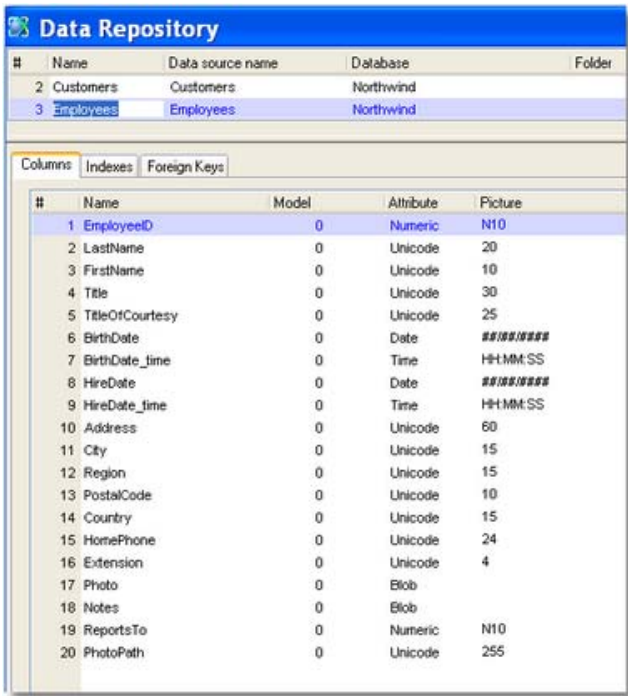


1. Select **Options->Get Definition (F9)**. The *Load Table Definition* dialog box will open.
2. From the Database field, zoom to select your database.
3. From the Tag tables field, select *Several* or *All*.
4. If you selected *Several*, zoom from the *Tables* field to select which tables you want. For each table you want to bring in, position the cursor on the Select column and press the spacebar.
5. Press **Select** when you are done choosing tables. That will bring you back to the *Load Table Definition* dialog box.

6. Press **OK**. That will close the dialog, and the tables will appear in the data repository.

Now you will have the definitions in eDeveloper.

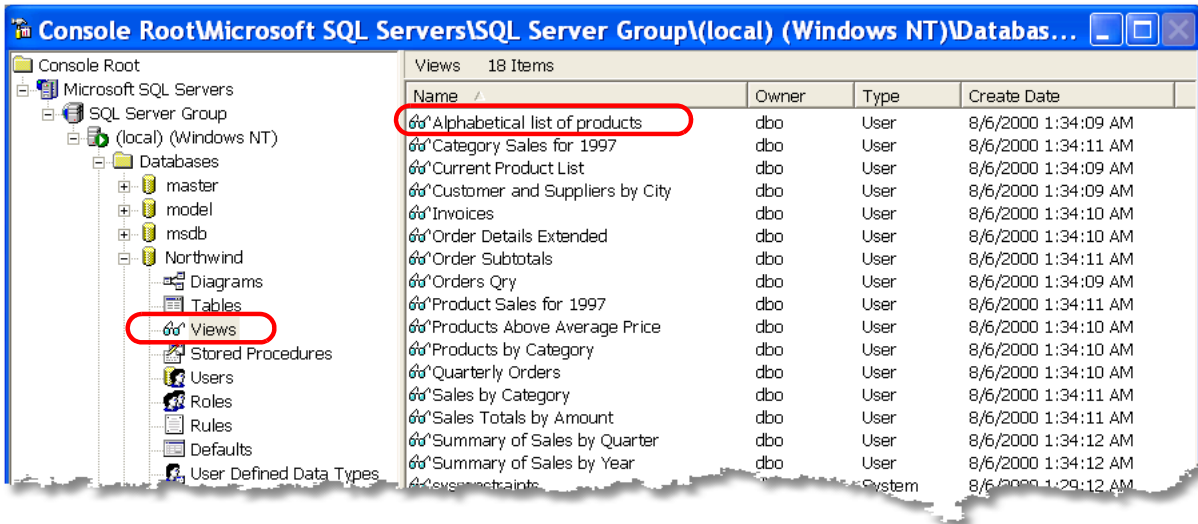
**Note:** If the table has dates and times, you may get a message “Translate SQL Datetime columns to eDeveloper date & time field pair?”.



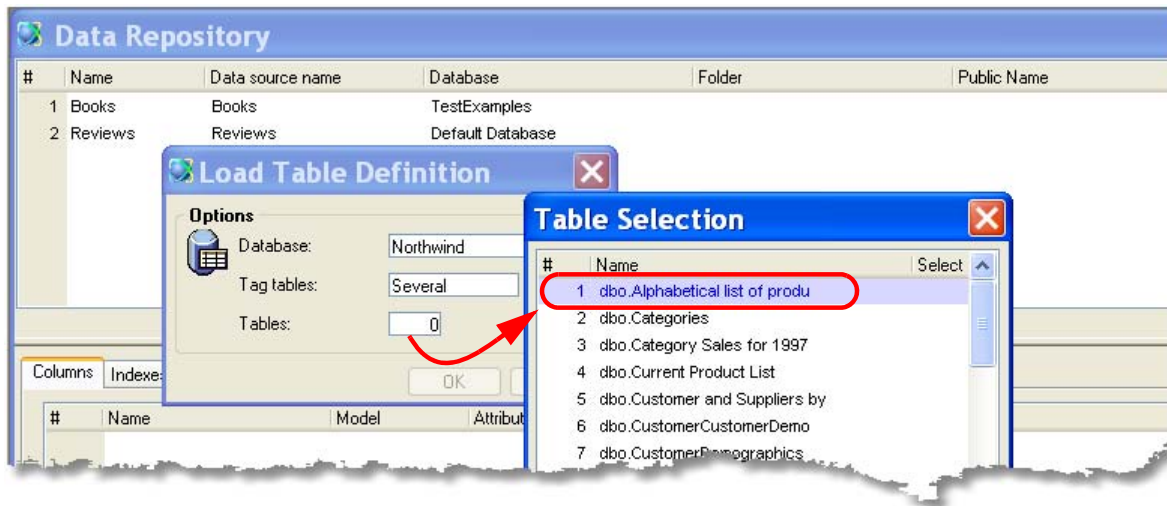
The screenshot shows the 'Data Repository' window with two tabs: 'Columns' and 'Indexes'. The 'Columns' tab is active, displaying a list of columns for the 'Employees' table. The columns are numbered 1 through 20. The 'Data source name' is 'Employees' and the 'Database' is 'Northwind'. The 'Columns' tab shows the following data:

#	Name	Model	Attribute	Picture
1	EmployeeID	0	Numeric	N10
2	LastName	0	Unicode	20
3	FirstName	0	Unicode	10
4	Title	0	Unicode	30
5	TitleOfCourtesy	0	Unicode	25
6	BirthDate	0	Date	####/####/####
7	BirthDate_time	0	Time	HH:MM:SS
8	HireDate	0	Date	####/####/####
9	HireDate_time	0	Time	HH:MM:SS
10	Address	0	Unicode	60
11	City	0	Unicode	15
12	Region	0	Unicode	15
13	PostalCode	0	Unicode	10
14	Country	0	Unicode	15
15	HomePhone	0	Unicode	24
16	Extension	0	Unicode	4
17	Photo	0	Blob	
18	Notes	0	Blob	
19	ReportsTo	0	Numeric	N10
20	PhotoPath	0	Unicode	255

## How do I Retrieve Data from a Database View?



Within a database there are database *tables*, and database *views*. A view is a sort of virtual table. It does not store data; rather it stores a SELECT statement which extracts data from one or more tables by some criteria.



eDeveloper, however, treats database views just as it does database tables. When you go to do a *Get Definition*, the database views are listed along with the tables. If you select a database view, you will build an eDeveloper table which will work like any other SQL table. You can tell it is a database view, however, because the name will be in brackets.

See Chapter 18, “How do I Access an Existing Database Table?” on page 464 for more information on using *Get Definition*.

## How do I Alter a Database Table Definition?

Once you have a database table defined in eDeveloper, you can change it when you need to. How you do the changes will depend somewhat on whether or not the table is also being used by another application, and whether or not there is “live” data in a production environment. We will discuss the following cases:

- **eDeveloper only:** The data is only defined by eDeveloper.
- **External data:** The data is defined in the DBMS, or by a 3rd party. This is often the case, for instance, when accessing data from an accounting application or purchased database.

### Altering database tables defined only in eDeveloper

If a database table is defined only in eDeveloper, changing it is easy. Your program references are automatically kept in sync if at all possible:

- If you *add* a column, the programs in the repository will be automatically updated
- If you *delete* a column, you need to make certain that no programs are accessing that column. Use the **Edit->Find and Replace->Find Reference (Ctrl+F)** utility on the column to check if it is used. If it isn't used, just delete it.
- If you *change* a column, the references will be changed too, if possible. If there are overrides on the display of the field, then the field display will not be changed. Also, if you change the attribute of a field from, say, numeric to alpha, then operations that move data to that field will no longer be correct.

Now, what about existing data? If in your database repository, **Properties->Options->Change Tables in Studio** is Yes, then eDeveloper will automatically reformat your data for you, and even make a backup copy just in case.

This is not something you would want to use with production data, however. When you deliver your finished application, you should also deliver a utility to reformat the existing data. The easiest way to do this is to keep the old database table entry in the repository, as well as the new one, and create an eDeveloper program to move data from the old one to the new one.

### Altering the eDeveloper data source to match an external table

Now, if you have a data source that is defined elsewhere, your approach should be different. Here, you need to bring in the definition from another source. Note that you need to be sure these tables have a database definition where **Properties->Options->Change Tables in Studio** is No, so that eDeveloper doesn't try to change the existing data.

You can do the changes manually by typing in the changes onto your existing data source definition, as you would for an eDeveloper-only table.

Or, you can use the **Options->Get Definition (F9)** utility to bring the definition in to eDeveloper automatically. Here is how you do it:

1. Use the **Get Definition** utility to bring in the table definition, as discussed in Chapter 18, “Accessing an existing database table” on page 464.
2. Print out the definition, or take a screen shot of it, so you can view it while you do the next step.

3. Edit your existing table definition (not the one you imported). If columns were added in the new definition, add blank entries to your existing definition. If columns were moved, move them in your old definition. The idea is to make the entries match in position and approximately in content. For instance, if Customer Name was the 5th line down in the new definition and was 20 characters long, and the new definition has it as the 6th line down and 30 characters, add a “dummy” column above it, but don’t worry about the number of characters.
4. Do the same for the indexes.
5. Once the two definitions match positionally, use the Overwrite function to overlay the new definition onto the old one.

This works because eDeveloper references the database columns by position, using an internal ID number. Usually, existing tables, especially those from commercial applications, do not change much, and when they do it is to add fields, so the process does not take long.



## How do I Define the Mapping Between an eDeveloper Field and a Database Column?

eDeveloper does a good job of making it easy to handle the huge variety of data types, by insulating you from the details of the implementation. When you are working in eDeveloper, you only have to deal with a handful of different types -- alpha, numeric, date, time, boolean -- and you don't have to worry about how those variables are actually stored in memory or in a database.

However, you do have control over the actual database storage. These details are handled in the Properties (**Alt+Enter**) for each column in the database table.

The details about the data attributes of the field in eDeveloper are contained in the *Details* section. Here we see this is a large numeric field with 10 digits, which can hold any value from -2,147,483,648 to -2,147,483,648.

In the *Storage* section, we see exactly how this field is actually implemented. It is stored as a 4-byte signed integer. The *Def/Null* and *SQL* sections give further information about the details of the field implementation. These are summarized below, but you can get more information about any one option in eDeveloper by positioning the cursor on it and pressing **F1**.

Column Properties Numeric : EmployeeID	
Categorized    Alphabetic	
<b>Model</b>	[default]
<b>Details</b>	
Picture	N10
Attribute	Numeric
Range	\-2147483648-2147483647
<b>Input</b>	
<b>Appearance</b>	
<b>Style</b>	
<b>Def/Null</b>	
<b>Storage</b>	
Char. Set	No
Default storage	Signed Integer
Stored as	Signed Integer
Modifiable	No
Size	4
Definition	Normal
Update style	Absolute

### Def/Null

Def/Null	
Null allowed	Yes
Null value	Please enter your first name
Null default	No
Database default	

**Null allowed:** If this is set to No, then the next three fields are greyed out and don't apply to this field; nulls will not be used at all. If Yes, then you can set the values for *Null value*, *Null display*, and *Null default*.

**Null value:** The value that the field is considered to contain when it is NULL. For instance, you might want to

If this is not specified, then the actual value of NULL is fetched from the DBMS section of the Magic.Ini file. NULL values are often some strange value that would never normally be entered. How-

ever, in your programming you don't have to remember the actual value because you can use the **NULL()** and **ISNULL()** functions.

**Null display:** What the user sees when the value is NULL. For instance, the Null value might be some odd date such as 01/01/1901, but the user could see "Please enter your birthday".

**Null default:** If this is Yes, then the default value for the field is NULL. Otherwise, the default value is whatever is entered in the *Default value* property.

**Default value:** eDeveloper will automatically initialize fields when they are first created; you don't have to do it manually. Most fields default to zero or blank, as you would expect. But if you have some specific value you want to use as an initialization value, you can enter it here.

**Database default:** eDeveloper passes this string to the DBMS in the CREATE TABLE statement. This allows you to set up the default value inside the DBMS. For example, when you create a database default 'defvalue' in an MSSQL table, eDeveloper generates the following: CREATE TABLE owner1.table1 (Col1 CHAR(10) NOT NULL DEFAULT 'defvalue'). eDeveloper adds this string without any additional formatting to the CREATE TABLE statement.

## Storage

Storage	
Char. Set	
Default storage	No
Stored as	Signed Integer
Modifiable	No
Size	4
Definition	Normal
Update style	Absolute

**Char.Set:** Allows you to choose between ANSI or OEM storage.

**Default Storage:** If set to Yes, the mapping will be determined by the DBMS, and the *Stored As* property will be ignored. This is useful if you want to use the same Data Source definition for multiple DBMS's.

**Stored As:** This is the actual data storage type. Zoom from this field to view and select the storage type. Note that for SQL columns, this can be further specified in the SQL section *Type* property.

**Modifiable:** Specifies if a user can change this field once the record is created. If *Modifiable*=No, then the user can enter a value when in Create mode, but not in Modify mode.

**Size:** The number of bytes allocated to the column.

**Definition:** Normal or String.

**Update Style:** This only applies to SQL columns when using Deferred transactions. If it is set to Differential, then eDeveloper updates the value based on the difference, rather than the actual value in mem-

ory. This is important when multiple users are working with the same field simultaneously, for example, when updating the current amount in stock.

## SQL



SQL	
Database information	
DB Column name	EmployeeID
Type	INTEGER IDENTITY
User type	

**Database Information:** You can encode SQL code here to pass to the underlying DBMS.

**DB Column name:** What the SQL column will be called in the DBMS.

**Type:** The underlying SQL type. Usually you don't need to specify this: eDeveloper chooses the type based on the Stored As property. But if you want to specify it directly, you can. Also, if you bring in the definition from the DBMS using *Get Definition*, you will see the underlying type here.

**User Type:** Some DBMS's allow you to create your own "user types" within the DBMS. If you do this, you can specify them here and the user type will be used by eDeveloper in creating the table.

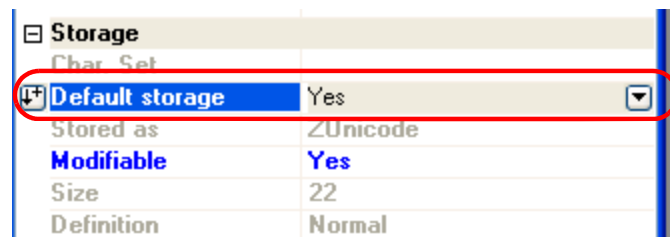
## How do I Set Data Source Mappings to Support Working with Multiple DBMS's?

One of the strengths of the high-level data definitions in eDeveloper is that you can use the same data source definition with multiple types of DBMS's. For instance, you can have a definition that works with an MSSQL table, but can also be used by Oracle or MYSQL.

There are some restrictions here, because every DBMS has its own way of doing things. Also, you have to be sure that any user-defined entries are the same in the various databases.

However, eDeveloper provides a couple of options to make this easier, by allowing you to define the data at a higher level and allowing eDeveloper to set up the DBMS-specific implementation at runtime.

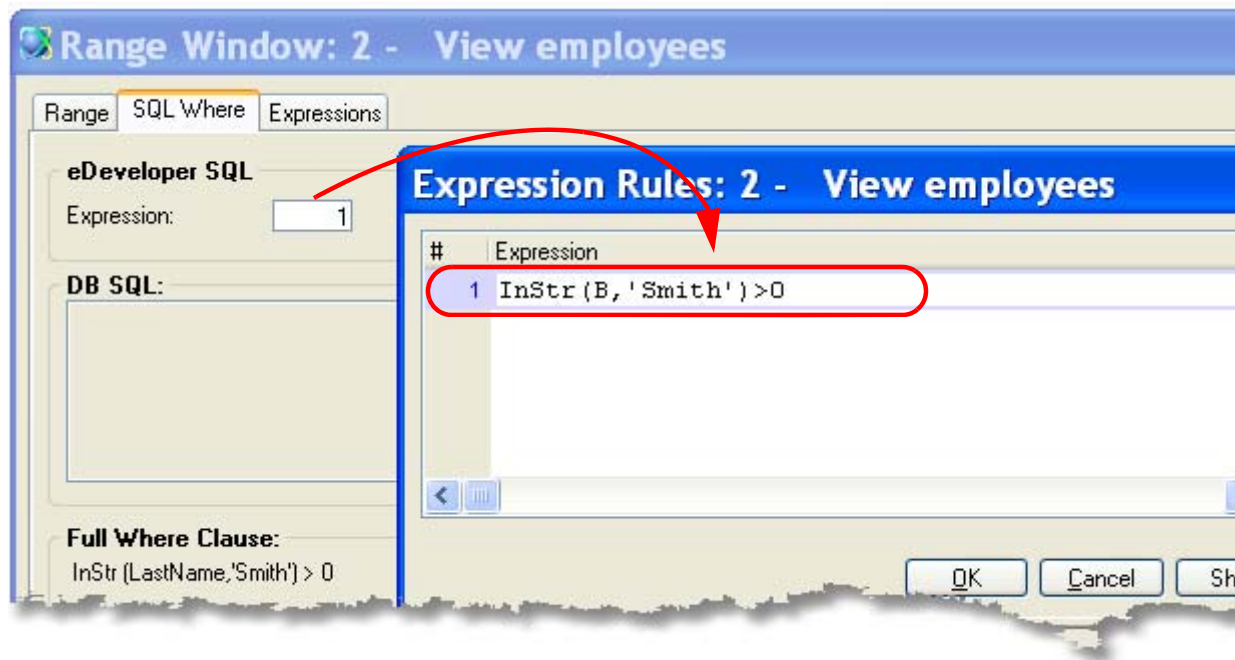
### Defining fields to be portable



#### Column Properties->Storage->Default Storage

If this property is set to Yes, then eDeveloper will ignore the Stored As property and instead use whatever storage works for the particular DBMS in use. For instance, if you have a numeric field, one DBMS might define that as INTEGER while another defines it as NUMBER or PACKED DECIMAL.

## Creating portable WHERE clauses



The syntax of WHERE clauses can vary between DBMS's. For instance, the function to fetch a substring of a certain string is *Substr* in DB2 and Oracle, and *Substring* in MSSQL. Obviously you cannot create a WHERE clause that works for both.

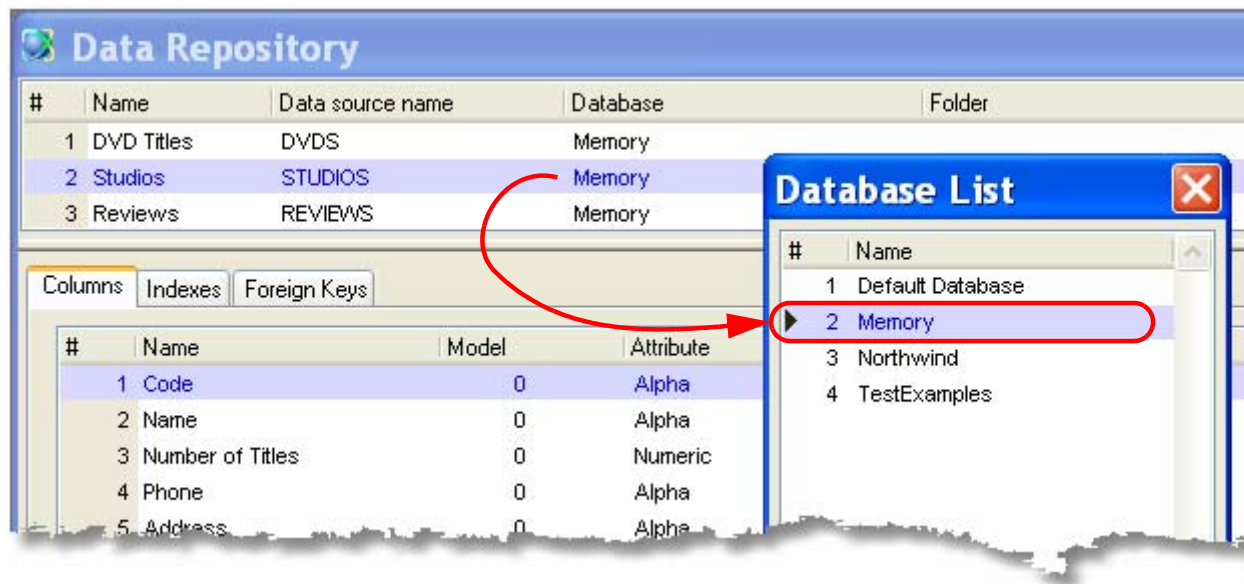
You can avoid this problem by using the **Task->Range/Locate->SQL Where->eDeveloper SQL** field to encode an expression in eDeveloper. Here, the syntax is purely eDeveloper syntax, but it will be translated to create the underlying SQL WHERE command at runtime.

## How do I Define a Table Entry for Non-Persistent data?

Although eDeveloper has array functions, it is generally easier to create a small table entry to hold temporary data, and treat them like any other table. This allows you to use Models to define the fields and the field display easily, and use can use functions like **MTblGet()** and **MTblSet()** to quickly populate the table and store it. Using a table entry means you don't have to keep track of indexes, and you can use Range and Locate functions to find the data you want.

Non-persistent tables in eDeveloper are called *Memory tables*. You create them as you would any other table, only you select a different database.

### Creating a Memory Table



1. Create your table as you would for any other database table (See Chapter 18, “How do I Create a Database Table Using eDeveloper?” on page 457 for details).
2. From the Database column, select the *Memory* table.

**Note:** If the Memory table is does not show up on the Database List, then it isn't defined in the Magic.Ini or **Options->Settings->Databases**. Normally, these entries are created when eDeveloper is installed. The Magic.Ini entry is under [MAGIC\_GATEWAYS]MGDB21, which points to the mgmemory.dll.

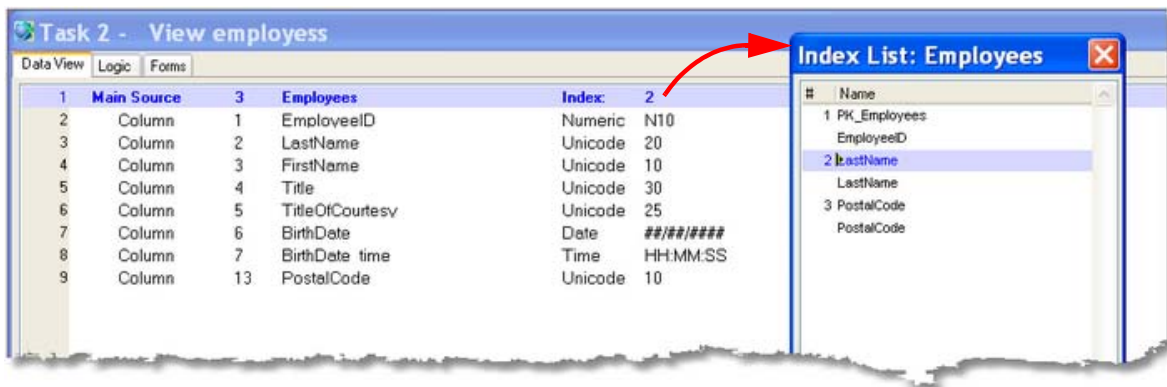
## How do I Retrieve Records from a Database Table in a Predefined Order?

When you are using a database table in eDeveloper, you can specify the order in which you want those records to appear. There are two aspects to this:

- **Specify the Index** to use: When you select a data source, you will also select which index to use.
- **Specify Ascending or Descending within that Index:** For each 2-way index, you can specify the direction to search.
- **Creating a sort on the fly:** You can also specify that the records be sorted at runtime. This option is a bit slower than the predefined indexes, but not significantly unless the table is very large.

These are described in detail below.

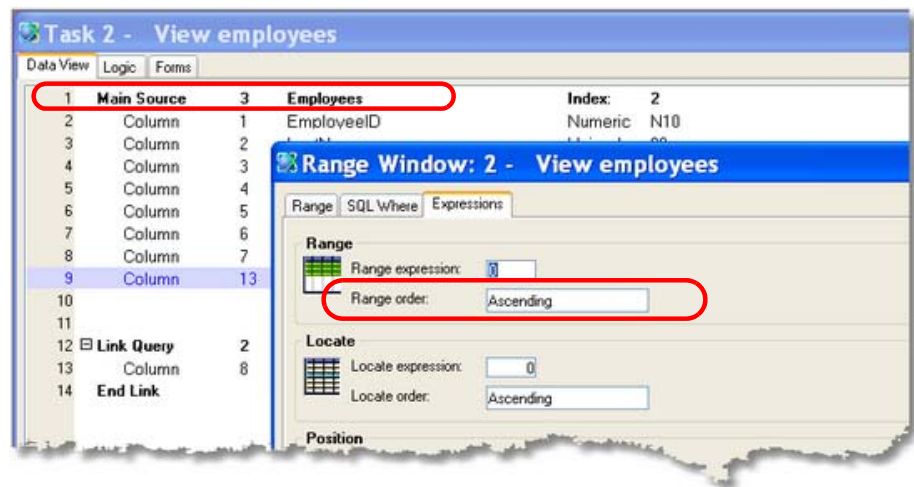
### Specifying the Index



When you select a data source in a program, you will have an *Index* column. **Zoom** from this column to select the index to use. For instance, in this example, we selected Index 2, which means the records will appear by employee Last Name.

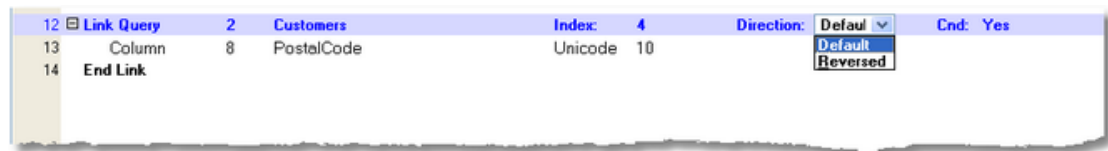
This applies both to the Main Source and the linked sources. if you do not specify an index, the records will be fetched sequentially, or in whatever order the DBMS decides is most efficient.

Specifying direction for a Main Source



You can specify the direction for a Main source in **Task->Range/Locate->Expressions (Ctrl+R)**, in the *Range order* and *Locate Orders* fields. The *Range Order* specifies the direction the records will be sorted for the display. The *Locate Order* is used when the user does a Locate or you are using the Locate column to position on a record.

Specifying direction for a linked source

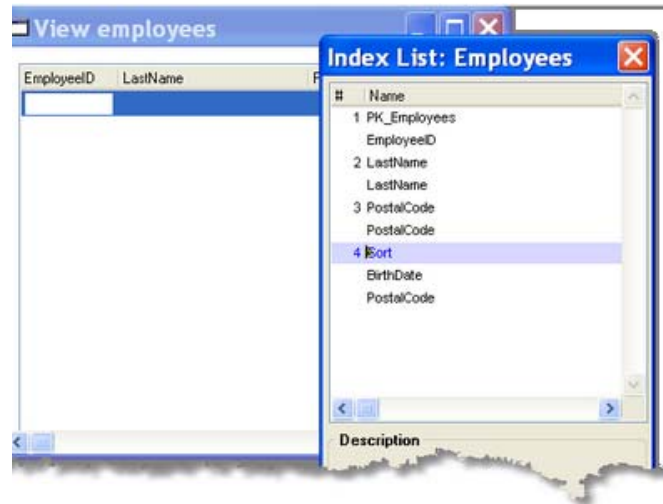


When you are linking to a source, the direction isn't usually as important because you can only position on one record. However, it does become important when you are trying to fetch the very first or very last record in the table. If you have selected an index with, say, the postal code in ascending order, then using a direction of *Default* will give you the smallest postal code in the table, while a direction of *Reversed* will give you the biggest postal code in the table.

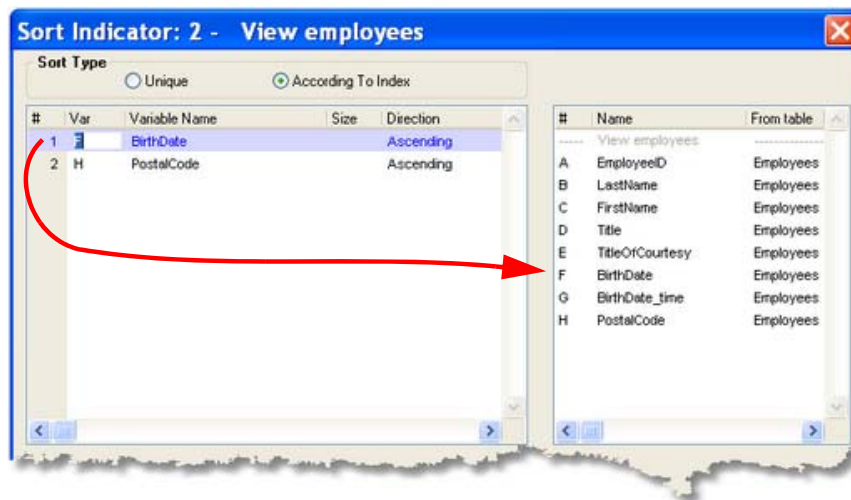


### Creating an index on the fly

Besides the indexes you have defined on your database table, you can instruct eDeveloper to create a virtual index at runtime. This index will show up on the index list if the user selects **Options->View by Key (Ctrl+K)**.



1. Select **Task->Sort (Ctrl+T)**.



2. For each item you want to participate in the Sort:
  - Press **F4** to open up a line.
  - **Zoom** to select the variable you want to participate in the Sort.
  - Set the Direction for that variable (**Ascending** or **Descending**).
3. When you are finished, press **OK**.

## How do I Browse a Database Table?

It is very convenient, when creating an application, to look at live data. This is very easy to do in eDeveloper, thanks to the *Generate Program* utility (**Options->Generate Program**, or **Ctrl+G**).

### Creating a Browse Program

1. In the *Data Repository*, position the cursor on the table you want to view.
2. Press **Ctrl+G** (**Options->Generate Program**).
3. Select:
  - **Mode=Execute** to view the data directly with a temporary, or **Mode=Generate** to create a new program in the Program Repository.
  - **Option=Browse**
  - Zoom from the *Columns* field to select only certain columns, if you want, and to change their order, if you want.
4. Press **Enter** or the **OK** button.

If you selected **Mode=Execute**, you will be presented with a view of the data in the database table. The default mode is query, so you will have to select **Options->Modify Records** if you want to change the data.

If you selected **Mode=Generate**, then there will be a new program in the Program Repository. You can run this program by pressing **F7**. You can also change the program if you want, to make it easier to view the data that is important to you during testing.

**Hint:** *Using the Program Generator is a good way to quickly begin a program. You can generate the program, then alter it to create your final product.*

---

## How do I Dump Data from a Database Table to a Text File?

It is often useful to dump data from a database table into a text file, to send data to a spreadsheet, to another application, or to convert the data within eDeveloper. eDeveloper provides a utility to do this automatically, the *Generate Program* utility (**Options->Generate Program**, or **Ctrl+G**).

### Creating a text export program

1. In the *Data Source Repository*, position the cursor on the table you want to view.
2. Press **Ctrl+G** (**Options->Generate Program**).
3. Select:
  - **Mode=Execute** to view the data directly with a temporary, or **Mode=Generate** to create a new program in the Program Repository.
  - **Option=Export**
  - Zoom from the *Columns* field to select only certain columns, if you want, and to change their order, if you want.
4. For *Text file*, enter the name of the text file you will create.
5. Press **Enter** or the **OK** button.

If you selected **Mode=Execute**, the data will be exported into the text file you specified.

If you selected **Mode=Generate**, then there will be a new program in the Program Repository. You can run this program by pressing **F7**.

**Note:** A plain text export can be very useful, but has some limitations. The particular problem happens to be memo fields which may contain a CR/LF, or BLOB fields. These sorts of files may require something more involved, such as XML output.

## How do I Load Data to a Database Table from a Text File?

It is often useful to import data from a text file. This is useful when you are importing data from a spreadsheet, or from a file that you have previously exported (Chapter , “How do I Dump Data from a Database Table to a Text File?” on page 479). eDeveloper provides a utility to do this automatically, the *Generate Program* utility (**Options->Generate Program**, or **Ctrl+G**).

### Creating a text import program

1. In the *Data Source Repository*, position the cursor on the table you want to view.
2. Press **Ctrl+G** (**Options->Generate Program**).
3. Select:
  - **Mode=Execute** to view the data directly with a temporary, or **Mode=Generate** to create a new program in the Program Repository.
  - **Option=Import**
  - Zoom from the *Columns* field to select only certain columns, if you want, and to change their order, if you want.
4. For *Text file*, enter the name of the text file you will import.
5. Press **Enter** or the **OK** button.

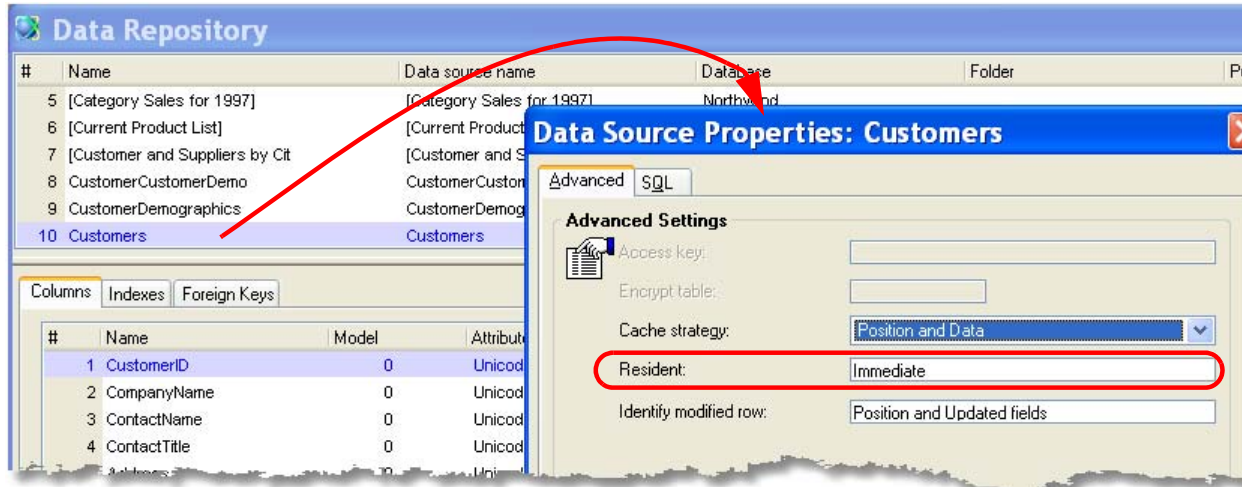
If you selected **Mode=Execute**, the data will be imported from the text file you specified.

If you selected **Mode=Generate**, then there will be a new program in the Program Repository. You can run this program by pressing **F7**.

**Note:** A plain text export can be very useful, but has some limitations. The particular problem happens to be memo fields which may contain a CR/LF, or BLOB fields. These sorts of files may require something more involved, such as XML input.

---

## How do I Fetch Data from a Database Table a Single Time for the Whole Application?



One of the more time-consuming activities for a program is reading data. If a table is accessed frequently, it will often speed up the application greatly if the table is read once and kept open. For read-only tables that are not updated frequently, it makes sense to read the entire table into memory and just access it as needed.

You needn't do this manually, however. Within the *Data Source Properties* (**Alt+Enter**) for each data source is the Resident setting. If this is set to anything other than No, the data from the table is read into memory and stays there until the application closes. Such a table is called a *Resident Table*.

Resident tables are read-only. They are only fetched once, so it is assumed that these are used for data that doesn't change frequently. However, if you do want to refresh them while the application is running, you can use the **DbReload()** function.

**Note:** **Settings->Environment->Preferences->Load Resident Tables** must be set to Yes, or the Resident setting will be ignored.

The Resident setting controls when the data is fetched into memory, as shown below.

### Resident Settings

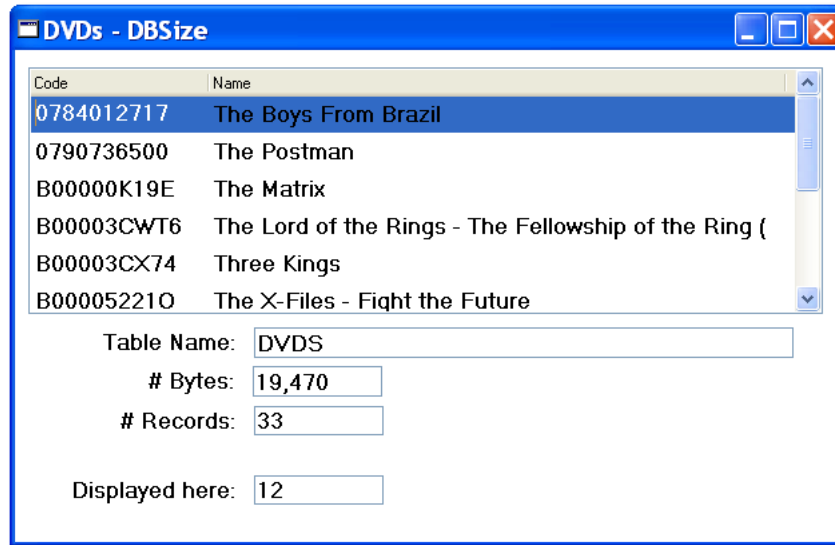
**No:** This is the default setting. The table is not treated as a resident table. It closes as soon as the task that uses it closes.

**Immediate:** The table is loaded as soon as the application is loaded, and remains open until the application closes.

**On Demand:** The table is loaded as the first time a program opens it, and remains open until the application closes.

**Immediate and on Browser:** The table is loaded directly from the browser and kept locally on the browser client side. The effect of a table set with this option, is that a recompute of a Link operation of that table is done locally.

## How do I Determine the Bulk Size When Fetching Records from a Database Table?



Sometimes you may want to know the size of a database table. There are three functions to do this:

- **DbSize()** returns the size of the table, in bytes.
- **DbRecs()** returns the number of records in the table.
- **DbViewSize()** returns the number of records in the current data view.

Each of these works slightly differently, as explained below.

### DbSize()

DbSize() is used when you want to determine the size of the table in bytes. The syntax is:

**DbSize(dsources#, tablespec)**

where:

- **dsources#** is sequence number of the table in the Data Source Repository.
- **tablespec** is the name of the table, if you want to specify a different one than the default.

In this example,

```
DbSize('1' DSOURCE, '')
```

returns 19,470, the byte size of the table.

### DbRecs()

DbRecs() is used when you want to determine the number of records in the table. The syntax is:

### DbRecs (*dsource#*, *tablespec*)

where:

- *dsource#* is sequence number of the table in the Data Source Repository.
- *tablespec* is the name of the table, if you want to specify a different one than the default.

In this example,

```
DbRecs ( '1' DSOURCE , ' ' )
```

returns 33, the total number of records in the table.

### DbViewSize()

DbViewSize() is used when you want to determine the number of records in the current data view. The syntax is:

### DbViewSize (*generation*)

where:

- *generation* is the generation number for the task (0 for the current task, 1 for the parent task, and so on).

In this example,

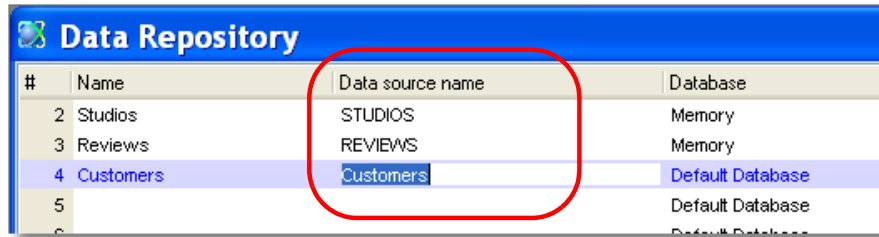
```
DbViewSize ( 0 )
```

returns 12, the total number of records in the table. This is because, although there are 33 records in the table, the user has used a Range to show only the records beginning with the letter 'T'.

**Note:** By default, the records in the data view are only fetched as needed, so DbViewSize() will only show a subset of the actual records that meet the selection criteria. If you want to get an accurate number for this function, you need to set **Task Properties->Data->Preload View** to Yes. This will load all the records into the view before the task starts. This not only makes the DbViewSize() accurate, it also allows the scroll bars to work as you would expect.



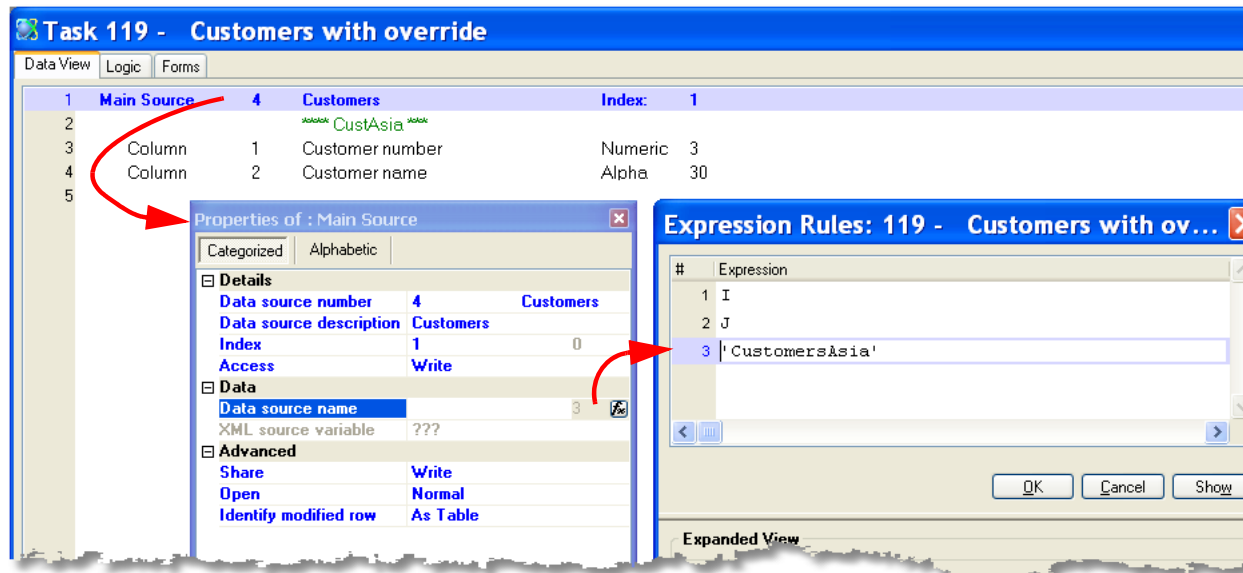
## How do I Dynamically Set a Data Source Name?



#	Name	Data source name	Database
2	Studios	STUDIOS	Memory
3	Reviews	REVIEWS	Memory
4	Customers	Customers	Default Database
5			Default Database
6			Default Database

Ordinarily, the Data source name is hard-coded in to the Data source name column in the Data Repository. However, it can be overridden within your program, when the data source is declared, and in functions that access that data source. Further, you can set the name using logical names, so that the name can be set at runtime. These options are discussed in more detail below.

### Overriding the Data source name in a declaration



**Task 119 - Customers with override**

**Data View** | Logic | Forms

#	Name	Index
1	Main Source	1
2		
3	Column	1
4	Column	2
5		

**Properties of : Main Source**

**Details**

Data source number	4	Customers
Data source description	Customers	
Index	1	0
Access	Write	

**Data**

Data source name	3	
XML source variable	???	

**Advanced**

Share	Write
Open	Normal
Identify modified row	As Table

**Expression Rules: 119 - Customers with ov...**

#	Expression
1	I
2	J
3	'CustomersAsia'

OK Cancel Show

Expanded View

When you declare a data source, either as a Main Source or as a Linked Source, you have the option of overriding the Data source name. To do this, you just set an expression in the **Properties->Data source name -> expression** column. Here, we have a copy of our “Customers” database called “CustomersAsia”.

### Overriding the Data source name in a function

Now, if we wanted to find the number of records in our “CustomersAsia” table, we also need to specify that this is a different table. So instead of using

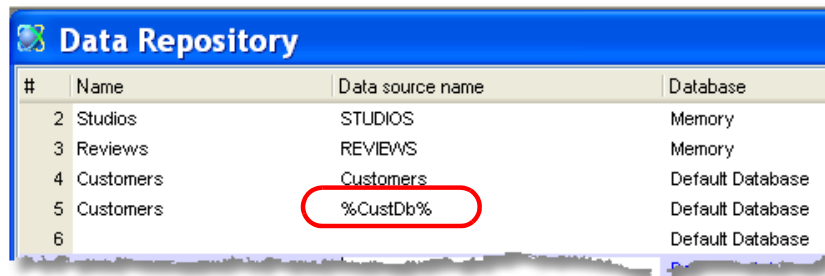
```
DBRecs( '4' DSOURCE', ' ' )
```

we would enter

```
DbRecs( '4' DSOURCE, 'CustomersAsia' )
```

Here we are using the second parameter of the DbRecs function to override the Data source name at runtime. Several of the Db functions use a second parameter for this.

## Using Logical Names for Data sources



#	Name	Data source name	Database
2	Studios	STUDIOS	Memory
3	Reviews	REVIEWS	Memory
4	Customers	Customers	Default Database
5	Customers	%CustDb%	Default Database
6			Default Database

However, it is not optimal to have these overrides hard-coded. A better idea is to use *Logical Names* so that the Data source name can be set at runtime. This is not only easier to maintain, but it gives us flexibility in other ways. For instance, instead of having just “CustomersAsia”, we could have “CustomersEurope” and “CustomersAustralia”. Or, we could have different named tables for different sets of archives.

Suppose we define a logical name %CustDB% for our Customers database. Then, our data source name would be %CustDB%, and our DbRecs function would read:

```
DbRecs( '4' DSOURCE, %CustDB% )
```

You can also use logical names in the Data Source Repository, allowing one table definition to be used for multiple actual tables at runtime.

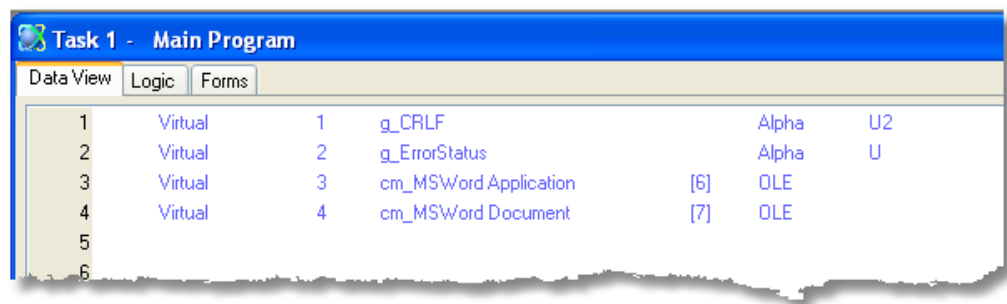
# Chapter 19: Main Program

## How do I Initialize My Application?

When you are setting up an application, there will be items you want to set up globally, before any of your programs run, and use in many programs. In eDeveloper, these items are kept in the Main Program, which is a special program at the top of the Program Repository.

You can think of the Main program as being the “parent” task of any program in the project. The Task Prefix runs before your program runs, even when you are just testing in Debug mode. You have access to any function or variable in the Main program while your program runs. And when the project is closed, Task Suffix of the Main program runs (or after you are finished testing your program in Debug mode).

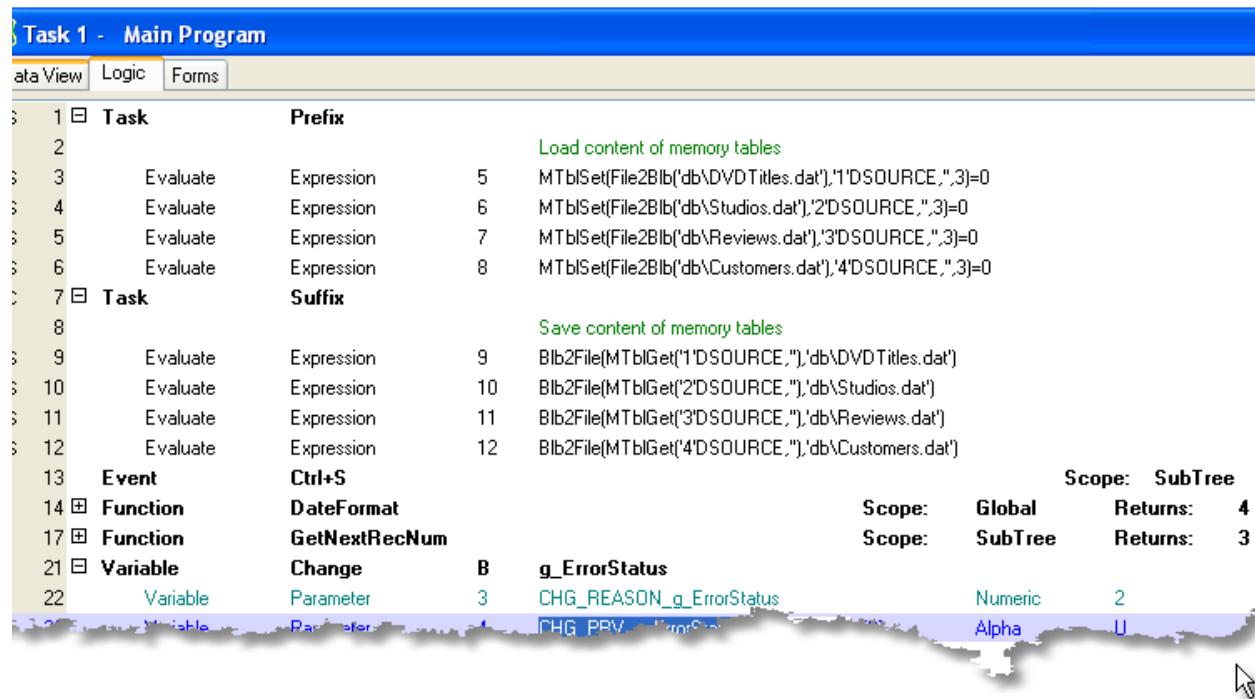
## What to set up in the Data View section of the Main Program



In the *Data View* section of the *Main Program*, you can put variables that will be used throughout the application, including OLE objects. You can initialize these variables in Task Prefix.

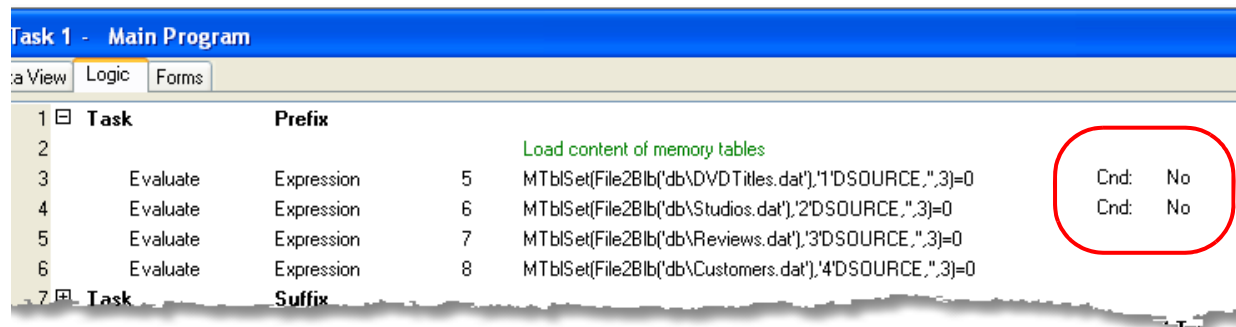
If you have an application that uses one type of variable a lot, such as a BLOB that is used for SOAP services, the Main Program’s Data View can be a handy place to keep that variable.

## What to set up in the Logic section of the Main program



1. In *Task Prefix*, put the operations you want to have execute before any programs run. Here, we have operations to initialize tables, by moving data from saved BLOB files into eDeveloper tables. You can also call programs here, for instance, to track user logins.
2. In *Task Suffix*, put the operations you want to have execute when the user is exiting the project. In this example we store the data from the tables back into Blobs. Again, you can call programs here to perform various tasks.
3. You can also enter *Function logic units*. Any functions you enter here will be globally available, and will show up on the eDeveloper function list.
4. *Event logic units* will be globally available if entered here. In our example, we have set Ctrl+S to call a Spellchecker for whatever field the user happens to be on. Doing this sort of logic in the Main Program means you don't have to do it multiple times in the individual programs.  
You can also use Event logic units to globally trap errors, using the *Error* type events, or trap *ActiveX* events.
5. *Variable Change* logic units can be used to give messages or do logging.

## How do I Skip Initialization Code?



Main Program

Whatever initialization process you have in the Main Program can be easily turned off by entering a condition in the condition column. Setting the Cnd: to No will prevent that code from executing.

Often, however, you will want to turn off the initialization only under certain circumstances, such as when you are testing. When you are testing programs, or using the Program Generator to look at files, the Main Program is automatically executed. This can slow things down a bit, and if you have some program that requires interaction in the Task Prefix (such as logging in to a timeclock), it can get time-consuming.

So, to conditionally prevent execution of code in these circumstances, you can use the **RunMode()** function. Runmode() returns:

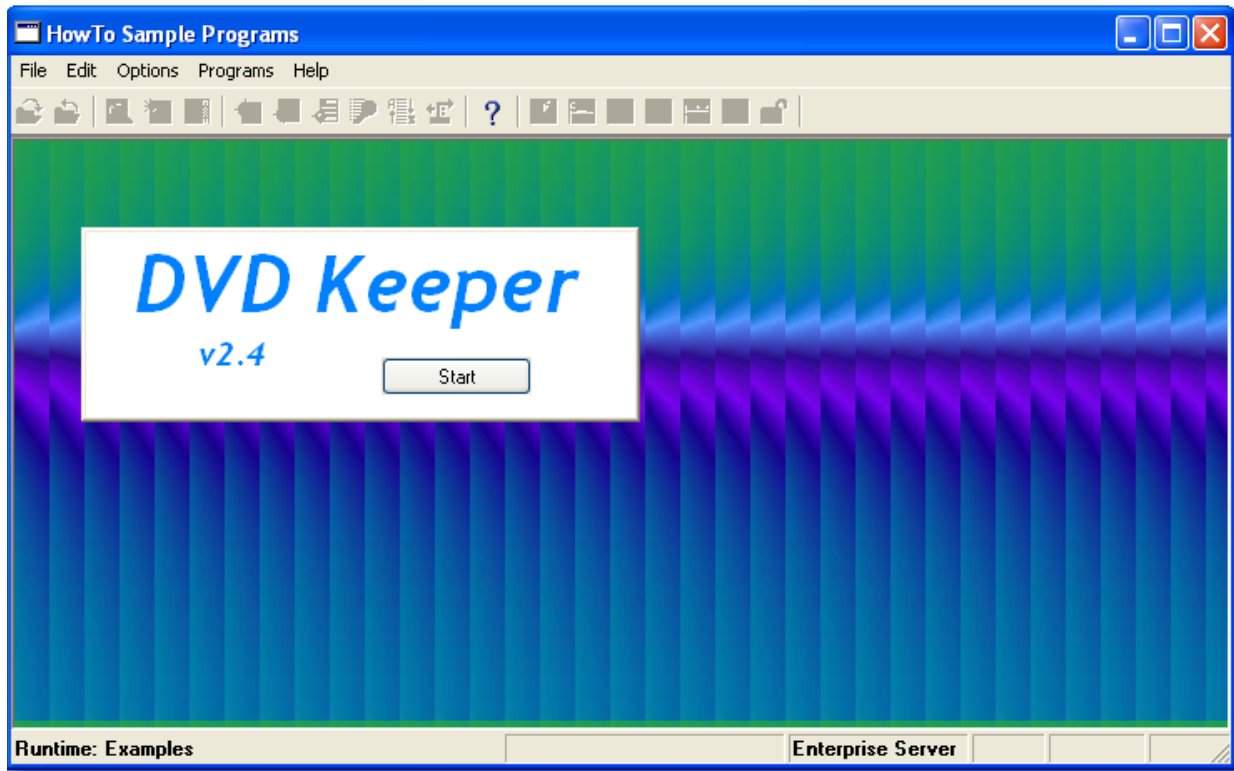
- **-1**: If the application is running as an Enterprise Server
- **0**: If the application is in full client/server runtime mode.
- **2**: If you execute a program using **Debug->Run (F7)** or **Options->Generate Program (Ctrl+G)**.
- **3**: If you execute the application using **Debug->Run Project (Ctrl+F7)**.

So, for instance, to disable initialization code only during development, you would enter a condition of

```
Runmode ( ) < 2
```

and the code would only run in production.

## How do I Implement a Background /Wallpaper for My Application?



You can set up your background and wallpaper in the Main Program. The Main Program forms work just as the other program forms do, except that they will show as long as the project is running, behind the other task forms. You cannot have any data entry on the Main Program form, but you can have interactive controls such as push buttons.

The Main Program forms work just as the other program forms do, except that they will show as long as the project is running, behind the other task forms. You cannot have any data entry on the Main Program form, but you can have interactive controls such as push buttons.

In our example, we use wallpaper for the background, and also have a text box with the title of our application and version number, with a Start button to bring up a customized, stay-up menu.

## Implementing a background in the Main Program

1. In *Task Properties*, set **Task Properties->Interface->Behavior->Show form**=Yes
2. In *Form Properties Details*, set **Window Type** = Fit to MDI
3. In *Form Properties Appearance*, set **Wallpaper** to whatever file you want to use for wallpaper

Other items, such as text or push buttons, you can add as you would for any other form.

**Hint:** You can customize the background further by using an expression for the file used for wallpaper, or by using an expression in the *Main Display* property of *Task Properties* to specify the form at runtime.

The screenshot shows the 'Form Properties GUI Display - Main Program' dialog box. The 'Details' section is expanded, showing the following properties:

Property	Value	Value
Model	[default]	
Window Type	Fit to MDI	0
Show in Window Menu	No	0
Form units	Dialog units	
Vertical factor	8	
Horizontal factor	4	
Show grid	Yes	
Grid X	1.000	
Grid Y	1.000	
Form name	Main Program	0
User State Identifier		0
Context Menu	0	0
Allow Drop	No	0
SDI		
Input		
Title bar	No	0
System menu	No	
Minimize button	No	
Maximize button	No	
Average palette	No	
Default Button		0
Appearance		
Wallpaper	%Images%\My_Wallpaper.jpg	0
Font	1	0
Color	1	0

The 'Default Button' section is also visible, with the text: 'Set the name of the push button control to be used as the default button of the form'.

# How do I Set and Use Global Variables (per Context Only)?

Task 1 - Main Program									
Data View Logic Forms									
1	Virtual	1	g_CRLF	Alpha	U2	Init:	4	ASCIIChr (13)%ASCIIChr (10)	
2	Virtual	2	g_ErrorStatus	Alpha	U				
3	Virtual	3	g_Company Name	Alpha	60	Init:	1	Translate('%Company%')	
4									
5	Virtual	4	cm_MSWord Application	[6]	OLE				
6	Virtual	5	cm_MSWord Document	[7]	OLE				
7									

Any variables you declare in the Main Program are available from anywhere in your project. In this example, for instance, we have set up a “Company Name” virtual, so we can use it in various paperwork and exports.

You can initialize the virtual by using an Init expression, as we did here. **Translate(‘%Company%’)** fetches the company name as a logical name from the Magic.Ini. Alternatively, we could have called a program in the Task Prefix to fetch the Company Name from a database table, or brought in the data with a link to a table.

**Note:** If you link to a table in the Main program, then at runtime, the Data source is read-only, and the Data source is kept open as long as the project is open.

## Using Global Variables

Task 22 - DVD Titles

Data View Logic Forms

1	Main Source	1	DVD Titles
2	Parameter	1	P: Studio
3			
4	Column	1	
5	Column	2	
6	Virtual	1	
7	Column	3	
8	Column	4	
9	Virtual	2	
10	Column	6	
11	Column	7	
12	Column	8	
13	Column	9	
14	Column	10	
15	Column	11	
16	Column	5	
17	Column	12	

Expression Rules: 22 - DVD Titles

#	Expression
1	J
2	CndRange (J<>' ', J)
3	' images\' &Ks'.jpg'
4	N*IF (P, 1-O/100, 1)
5	C

Index: 0

Variable List

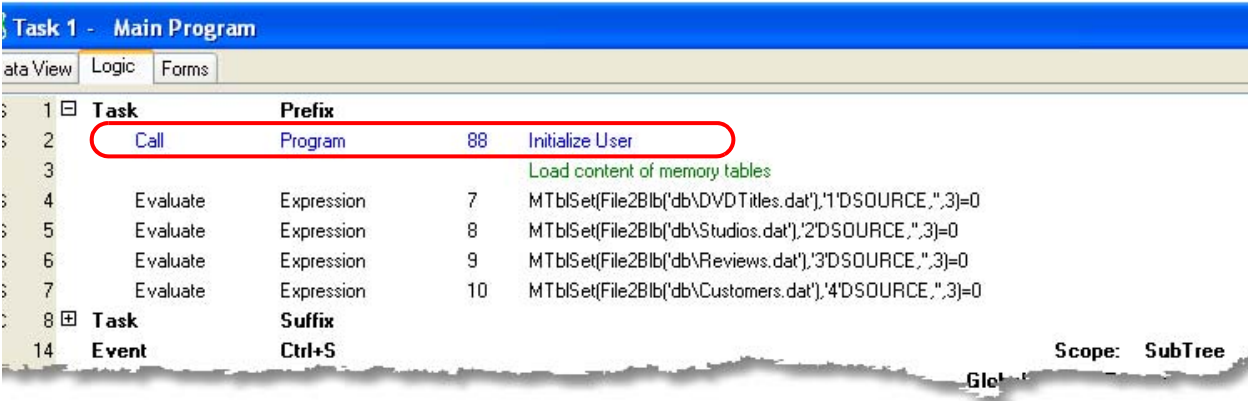
#	Variable Name	Attribute	Data Source
----- Main Program			
A	g_CRLF	Alpha	Virtual
B	g_ErrorStatus	Alpha	Virtual
C	g_Company Name	Alpha	Virtual
D	cm_MSWord Application	OLE	Virtual
E	cm_MSWord Document	OLE	Virtual
----- DVD Titles			
J	P: Studio	Alpha	Parameter
K	SN	Alpha	DVD Titles
L	Title	Alpha	DVD Titles
M	Tab	Alpha	Virtual
N	List Price	Numeric	DVD Titles

Once you have the variables in the Main Program, you can access them as you would any other variable. They show up at the top of the variable list from any task you are working on. You can fetch data from them in your Expressions, or move data to them using Update operations.





# How do I Run a Certain Procedure Upon Application Invocation?



Any programs you want to run when an application starts should be called in the *Task Prefix* logic unit of the Main Program. You can call a program here as you would anywhere else in eDeveloper, including calling components. If the program you are calling is in the same project, the called program has access to the Main Program's variables, so you may not need to pass parameters.

How do I Run a Certain Procedure Upon Application Termination?

Main Program

- Main Program

LogicForms

Task	Prefix	Suffix	
Call	Program	89	LogOut User
			Save content of memory tables
Evaluate	Expression	11	Bib2File(MTblGet("1'DSOURCE",""),'db\DVDTitles.dat')
Evaluate	Expression	12	Bib2File(MTblGet("2'DSOURCE",""),'db\Studios.dat')
Evaluate	Expression	13	Bib2File(MTblGet("3'DSOURCE",""),'db\Reviews.dat')
Evaluate	Expression	14	Bib2File(MTblGet("4'DSOURCE",""),'db\Customers.dat')

EventCtrl+S

FunctionDateFormat

Scope:SubTree

Scope:Global

Returns:6

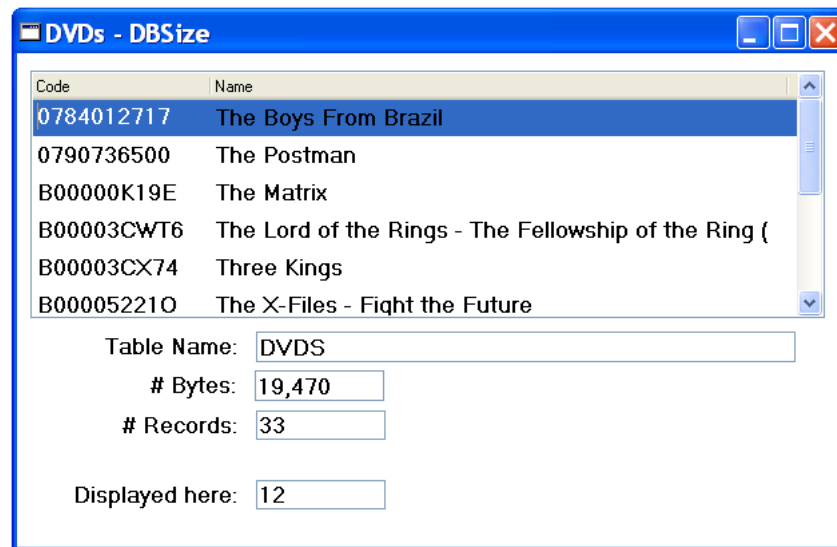
Any programs you want to run when an application ends should be called in the *Task Suffix* logic unit of the Main Program. You can call a program here as you would anywhere else in eDeveloper, including calling components. If the program you are calling is in the same project, the called program has access to the Main Program's variables, so you may not need to pass parameters.



## Chapter 20: Data View

### How do I Retrieve the Number of Records in a Task's Data View?

Sometimes you may want to know the size of a database table. There are three functions to do this:



- **DbSize()** returns the size of the table, in bytes.
- **DbRecs()** returns the number of records in the table.
- **DbViewSize()** returns the number of records in the current data view.

DbViewSize() is explained below.

## DbViewSize()

DbViewSize() is used when you want to determine the number of records in the current data view. The syntax is:

**DbViewSize** (*generation*)

where:

- *generation* is the generation number for the task (0 for the current task, 1 for the parent task, and so on).

In this example,

```
DbViewSize(0)
```

returns 12, the total number of records in the table. This is because, although there are 33 records in the table, the user has used a Range to show only the records beginning with the letter 'T'.

**Note:** By default, the records in the data view are only fetched as needed, so DbViewSize() will only show a subset of the actual records that meet the selection criteria. If you want to get an accurate number for this function, you need to set **Task Properties->Data->Preload View** to Yes. This will load all the records into the view before the task starts. This not only makes the DbViewSize() accurate, it also allows the scroll bars to work as you would expect.

## How do I Determine a Task's Main Data Source?

Task 10 - DVD Titles List							
Data View   Logic   Forms							
1	<b>Main Source</b>	1	<b>DVD Titles</b>	<b>Index:</b>	2		
2	Column	1	SN	Alpha	U10		
3	Column	2	Title	Alpha	100	Range: 3	To: 3
4	Column	3	List Price	Numeric	####.##		
5	Column	4	Discount	Numeric	3%		
6	Column	6	Studio	Alpha	4	Range: 1	To: 2
7							
8	<input checked="" type="checkbox"/> <b>Link Query</b>	2	<b>Studios</b>	<b>Index:</b>	1	<b>Direction: Default</b>	
9	Column	1	Code	Alpha	4		
10	Column	2	Name	Alpha	50		
11	<b>End Link</b>						
12							

The *Main Source* of a task is always declared as the very first item in the **Data View tab**. The number and name of the Main Source are shown, as well as the index that will be used to display the records. You can specify the rest of the details, such as the share mode, in the **Properties Pane (Alt+Enter)**.

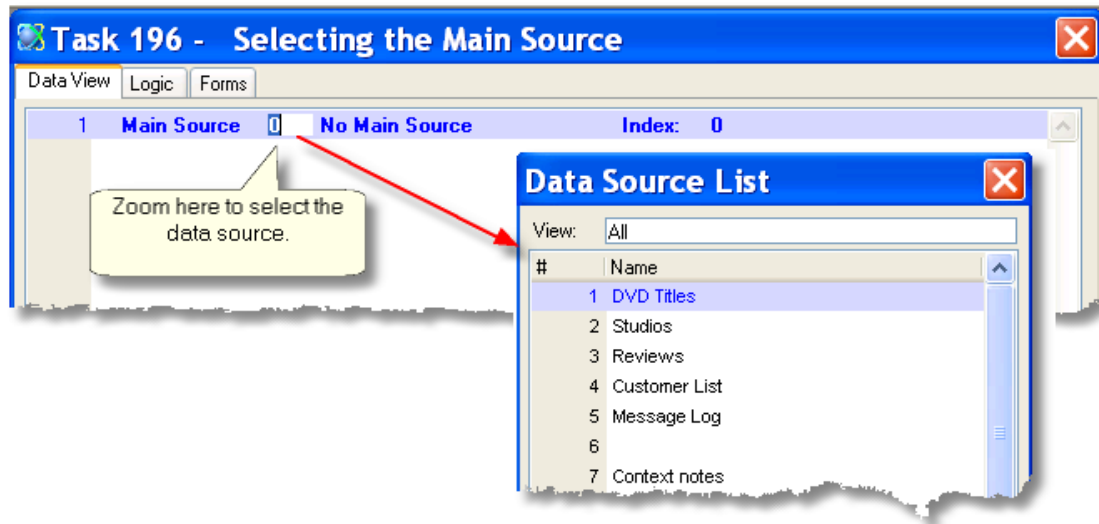
The only way a task can retrieve multiple records is if it has a *Main Source*. Once the *Main Source* is specified, the *Main Source* table drives the task. The number of records that are displayed (or cycled through, in the case of a batch task) are determined by the records that meet the Main Source Range criteria.

A Main Source is not required. If the task does not have a Main Source, then the first line will list:

Task 23 - Populate DB tables				
Data View   Logic   Forms				
1	<b>Main Source</b>	0	<b>No Main Source</b>	<b>Index:</b> 0
2				

Online tasks with no *Main Source* do not show a list of records; they can only display one record at a time. Batch tasks with no *Main Source* will, by default, loop forever so you need to specify an End Task Condition in the Task Properties.

## Entering the Main Source



1. Move the cursor to the field after "Main Source".
2. **Zoom**. A list of *Data sources* will appear.
3. Position the cursor on the *Data source* you want, and press **Enter**.



# How Can I Dynamically Set the Order of the Retrieved Records in a Task?

When a Main Source is specified, the task, by default, will retrieve every record in the database. The order in which they are retrieved is set by the programmer. There are three ways to do this:

- A hard-coded Index
- An Index Expression
- A Studio Sort

Each of these is explained in more detail below.

## Using a Hard-coded Index

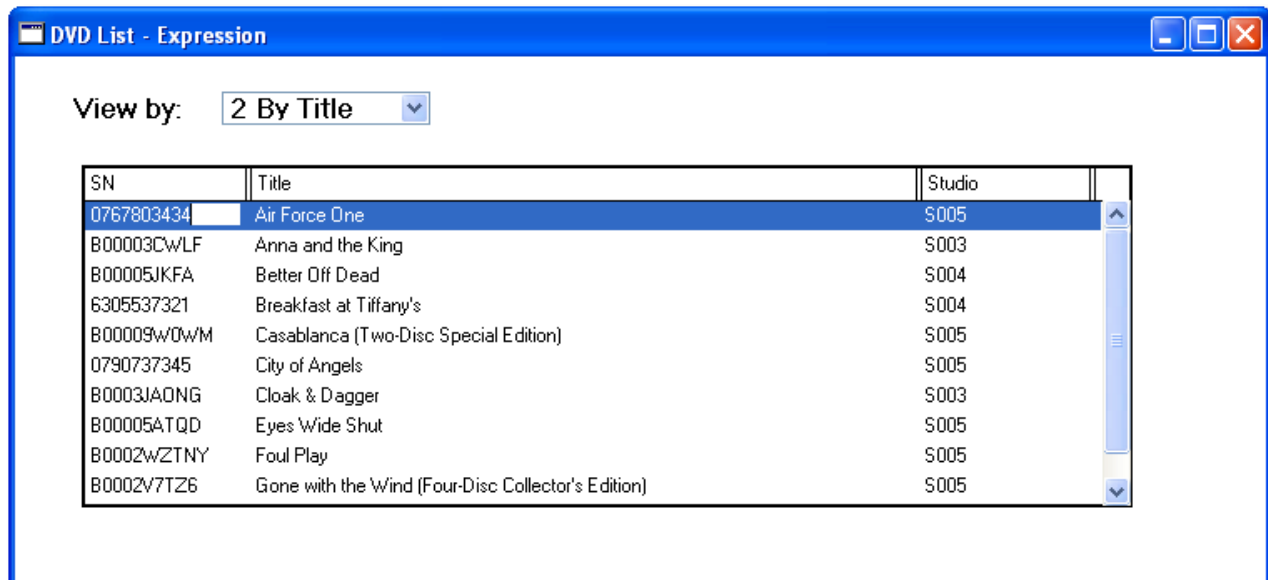


The easiest way to specify an Index is to simply specify it in the Index column of the Main Source. In this example, we sort the DVD titles by Title, which happens to be Index 2.

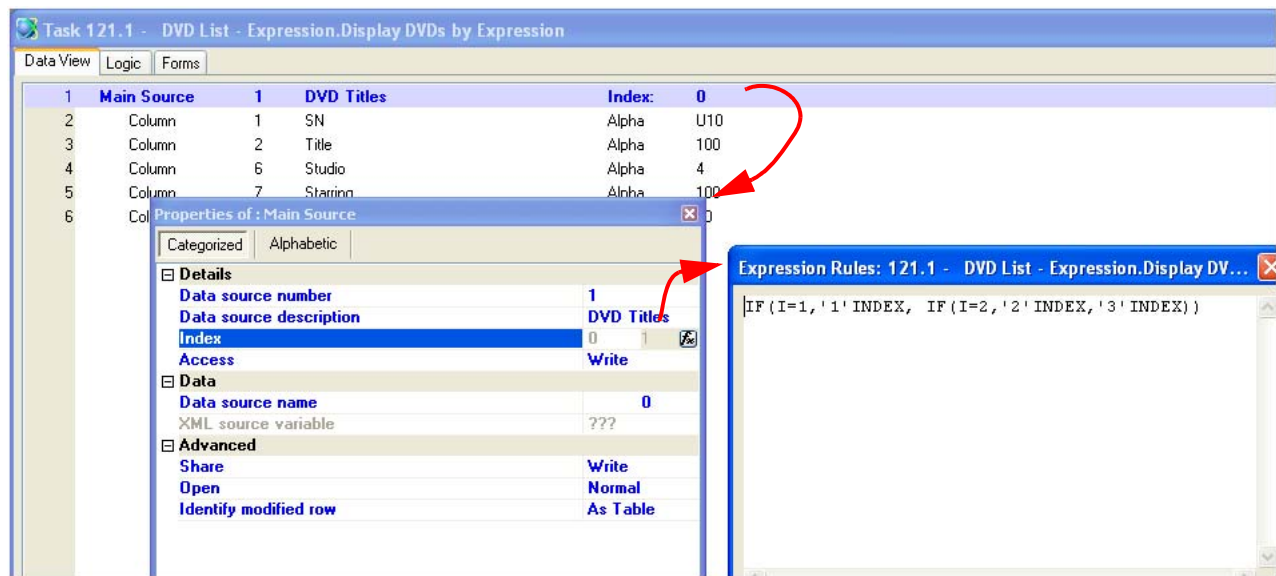
1. Go to **Data View->Main Source**.
2. **Zoom** (F5 or double-click) from the Index column. You will see a list of the available Indexes.
3. Press **Enter** to select the Index you want to use.

If the Index number should happen to change in the Studio (because someone added a new index, for instance), to the Index specified here would automatically change.

## Using an Index Expression



If you want to choose the index at runtime, you can use an Index Expression to choose the index. This is a good way to allow the user to choose the sort order, or you can use some logic to choose the correct sort order based on selection criteria chosen by the user.

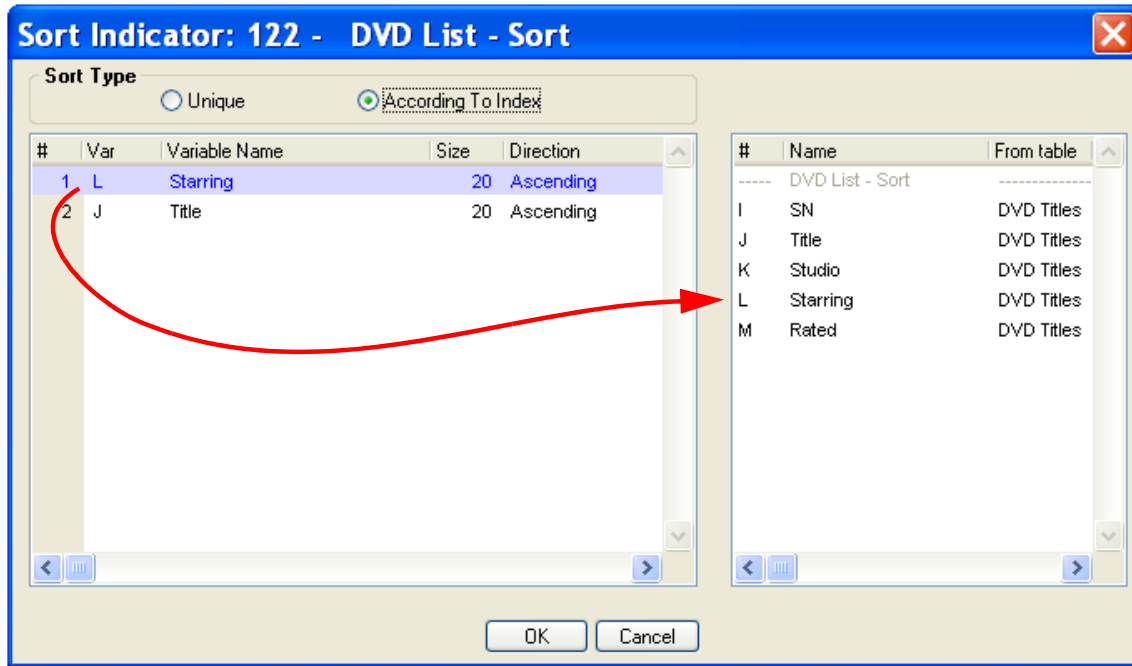


1. From your Main Source, press **Alt+Enter** to go to the *Properties of the Main Source*.
2. Go to **Details->Index**. The second column is the *Expression* column. **Zoom** from here to enter an *Expression* that, at runtime, evaluates to a valid Index number.

In our example, we are choosing an index based on the selection criteria that the user entered. Note that while we could have just used the value the user selected (1, 2 or 3) directly, we use an IF to map the

choice to INDEX literals. This is important, because if the index numbers should change in the future, the INDEX literals will change automatically.

### Using a Task Sort



If there is no index that does what we need, it is still a simple matter to display the records in the order you require. eDeveloper has a built-in Sort facility. You can specify any variable in this sort, and eDeveloper will use an SQL Order by, or, for an ISAM table, build a temporary index into the table on the fly at runtime.

1. Open the task you want to sort.
2. Select **Task->Sort (Ctrl+T)**. This will open the Sort Indicator window.
3. The left side of the window will either be blank (if no sort currently exists) or will contain a list of variables (if a sort already is being done for this task). This list will determine the sort order of the table. In our example, the list will be sorted first by “Starring”, then by “Title”.
4. To add a variable to the list:
  - Press **F4 (Edit->Create Line)** to open up a line.
  - In the *Var* column, type in the letter of the variable you want to use, or zoom to select it from the list on the right.
  - Some variables are too long to sort efficiently. You can cut down the size of the alpha field that is used to sort by, by entering the number of characters in the *Size* column. In our example, the “Starring” and “Title” fields are each 100 characters long, but we are only going to use the first 20 characters.
  - If you want, you can change the *Direction* to Descending. This is useful for dates especially, where you might want to have the newest items at the top of the list.

5. To delete a variable from the list, press **F3** (**Edit->Delete Line**).
6. When you are done, press OK.

**Hint:** *For ISAM tables, the Studio Sort is executed after the Range is executed. It makes sense to limit the range of records to some reasonable number when doing a Studio Sort, since an alternate table is being built on the fly.*

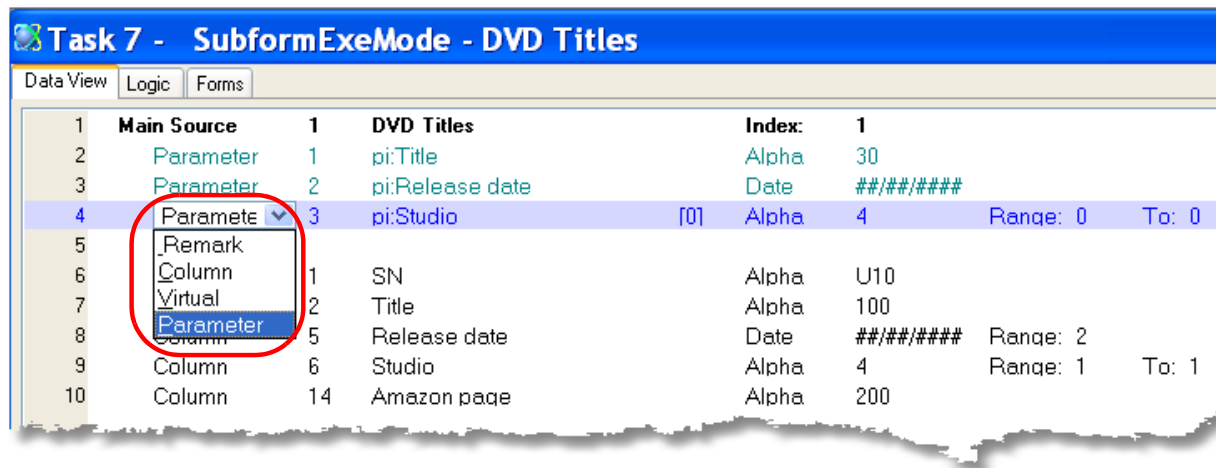
## How do I Define the Incoming and Outgoing Arguments in a Program?

Programs in any computer language are designed to pass arguments in and out. Arguments coming in are passed as *parameters*. A parameter can pass data in, or it can receive data going out, or both.

Values going out can be coded as parameters, or, they can be coded as *returned values*. Returned values are typically used for programs that act as functions; that is, they can be used in the middle of an expression via a *CallProg()* function, but the returned value can also be received in the Result column from any Call Program operation.

Below, we explain how to enter parameters and returned values in eDeveloper.

### Parameters

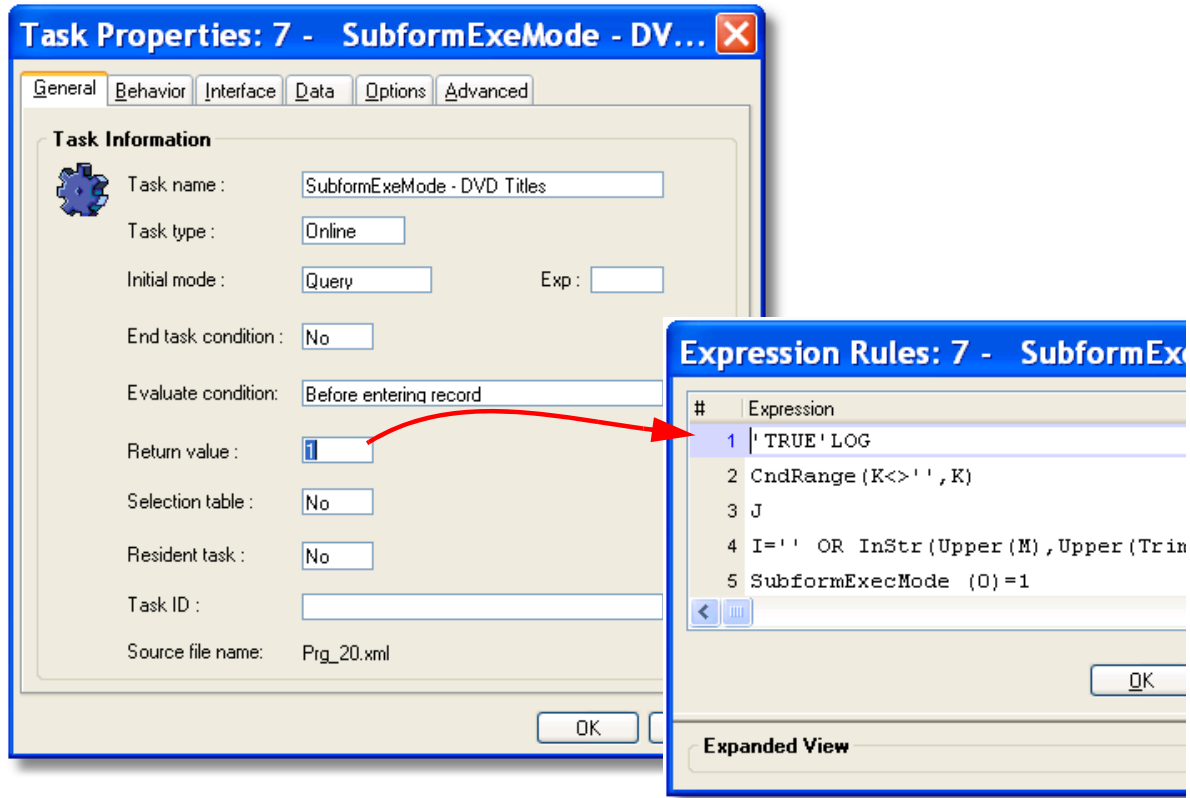


1. Parameters are entered in the same way as any other variable. The only difference is, you select *Parameter* rather than *Column* or *Virtual*.
2. Enter the *name* of the parameter. It doesn't matter to eDeveloper what you call it, but it is a good idea to have some naming convention so the users of this program can tell if it is an incoming, outgoing, or incoming/outgoing argument. One convention is to use 'pi' for Parm In, 'po' for Parm Out, and 'pio' for Parm In/Out.
3. Set the attribute and other properties as you would for any variable. It is a good idea to use *Models* for variables, to ensure that the fields lengths match.
4. It does not matter where the parameters are entered. They can be at the top of the *Data View*, or the bottom, or scattered between other variables. It is easier for maintenance, however, if you have a standard on where to put them; usually they are at the top of the *Data View*.

eDeveloper will present this list of parameters to the calling program in eDeveloper, and it is also used when creating Component, COM or SOAP objects. When you are calling this program, you will be able to see the name and data type of the parameters, and eDeveloper will syntax check the number and type of the arguments.

**Note:** If the calling task passes in an expression (*passing by value*), then obviously eDeveloper cannot update the value. So if you have a variable **G**, and you type **G** in the Var column, it can be updated. If you code an expression with **G** in it, then the value of **G** will be passed in to the program, but the variable **G** will not be updated. You can use expressions for passing information when you want to be certain that the value is not changed.

## Returned Values



1. Go to **Task Properties->General->Return Value**.
2. **Zoom** to the Expression Rules.
3. Enter an expression that will be the value you want to return.

**See also:** Chapter 5, "How do I Set a Program to Return a Value to the Calling Program?" on page 88.

## How do I Define Task Variables?

**Task 124 - DVD List - Variables/Link**

Data View Logic Forms

1	Main Source	1	DVD Titles		Index:	2
2	Parameter	1	pi.SN	[6]	Alpha	4
3						
4	Column	1	SN		Alpha	U10
5	Column	2	Title		Alpha	100
6	Column	6	Studio		Alpha	4
7	Column	7	Starring		Alpha	100
8	Column	12	Rated		Alpha	10
9						
10	Virtual	1	b.WatchClip	[40]	Alpha	U\Watch\Cli
11						
12	Link Query	2	Studios		Index:	1
13	Column	1	Code		Alpha	4
14	Column	2	Name		Alpha	50
15	Column	3	Number of Titles		Numeric	4
16	End Link					
17						
18						

Direction: Default

**Parameter:** Used to pass data in and out of programs.

**Column:** Used to select fields from a Data source. The Data source can be the Main Source or a Linked table.

**Virtual:** Temporary variables local to this task and its children.

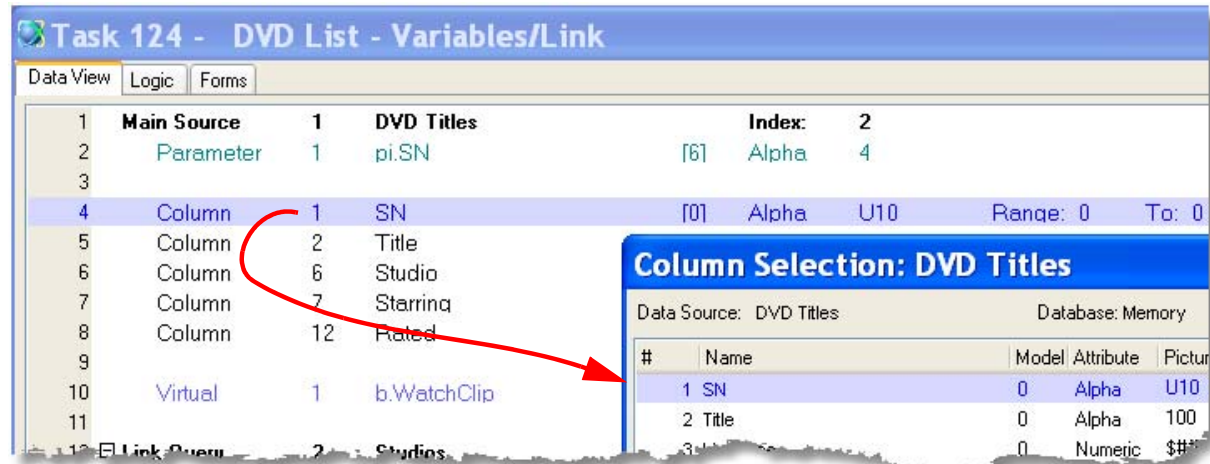
There are three kinds of task variables in eDeveloper:

- *Parameters* are used to pass data in and out of a program. These are entered the same way as virtuals are. In our example we are passing in the studio number (“SN”).
- *Columns* are fields that are selected from a data source. In eDeveloper, a data source can be an SQL table, a memory table, or even an XML file, but they are all handled the same way. In our example we are selecting columns from the Main Source (the DVD table) and a Linked source (The Studio file).
- *Virtuals* are temporary variables that will be used only in this task or its children. In our example, we are using a virtual to create a parkable push button.

Once they are declared, each of these variable types works identically in eDeveloper. The main differences have to do with what happens to the data after the task ends.

Parameters are explained in more detail in Chapter 20, “How do I Define the Incoming and Outgoing Arguments in a Program?” on page 505. Below you will see how to enter Columns and Virtuals.

## Creating a Column



1. If you are creating a column in a Linked source, position the cursor between the *Link* and *End Link*. Otherwise, the column will be assumed to be from the Main Source.
2. Press **F4** (**Edit->Create Line**) to open up a line.
3. Type “C” for *Column* (or select from the pull-down list). Tab.
4. **Zoom** (**F5** or **double-click**) to select the column you want, or just type it in. Tab.
5. You can change the name if you like by typing over it. This is sometimes useful, especially when, as in this example, you have more than one variable with the same name. Changing the name in the Data View doesn’t affect the Data Source table.

That is all there is to it. Since the properties of the columns are set up in the Data Source table, you don’t need to specify them here.

**See also:** Chapter 20, “How do I Define a Range for a Task’s Data View?” on page 510.



## Creating a Virtual

**Task 124 - DVD List - Variables/Link**

Data View Logic Forms

	Main Source		DVD Titles		Index:	
1	Parameter	1	pi.SN	[6]	Alpha	4
2						
3						
4	Column	1	SN		Alpha	U10
5	Column	2	Title		Alpha	100
6	Column	6	Studio		Alpha	4
7	Column	7	Starring		Alpha	100
8	Column	12	Rated		Alpha	10
9						
10	Virtual	1	b.WatchClip	[40]	Alpha	U\Watch \C
11	Virtual	2	ax.VideoWindow		OLE	
12	Virtual	3	v.VideoReturnCode	[0]	Numeric	3
13						
14	Link Query	2	Studios		Index:	1
15	Column	1	Code		Alpha	4
16	Column	2	Name		Alpha	50
17	Column	3	Number of Titles		Numeric	4
18	End Link					

Local Variable Properties Numeri

Categorized Alphabetic

**Model**

**Model** [default]

**General**

**Details**

**Picture** 3

Attribute Numeric

Range

**Input**

**Appearance**

**Style**

**Def/Null**

**Storage**

Char. Set Ansi

Modifiable Yes

Update style Absolute

1. Position the cursor where you want to create a virtual. They can be created anywhere.
2. Press **F4 (Edit->Create Line)** to open up a line.
3. Type “V” for *Virtual* (or select from the pull-down list). Tab.
4. You will now be on the variable name. Type in any name you like, then Tab.
5. Now you are on the *Model* field. This field is in brackets, and it is the number of the Model used for this field. In our example, the “b.WatchClip” field uses Model 40, but the v.VideoReturnCode does not use a Model.

Using a Model will save you time and make your code more maintainable.

To select a Model, **zoom (F5 or double-click)** to select the Model you want.

Now, if you are using a Model, the rest of the properties for the virtual are already set up. You can override the defaults, if you like, but if the Model is well-thought-out, you can just use it.

If you are not using a Model, tab and continue to the next steps.

6. The field after Model is the *Attribute* field. Here you select the variable’s Attribute, which you can choose from a pull-down list, of Alpha, Numeric, Date, Time, OLE, etc. Now **Tab** to the next field.
7. Now you are in the *Picture* field. Enter the Picture for the field. If you need more assistance, press zoom and a Picture dialog will help you.

At this point, the virtual should be usable, though some, like the OLE objects, are more complex. You can go to the **Properties Pane (Alt+Enter)** and set up more details for the virtual, if needed.



those bounds. For alpha fields, it also does some masking, so that if the search string is followed by a '\*', all characters after the string are ignored.

This kind of Range is commonly used to select all records of one type (the same status, same country, same parent record ID), or to select one particular record, by setting the FROM and TO ranges to the same value.

In our example, we are searching for all the records that match a search string that is passed in to our program. If the string begins with:

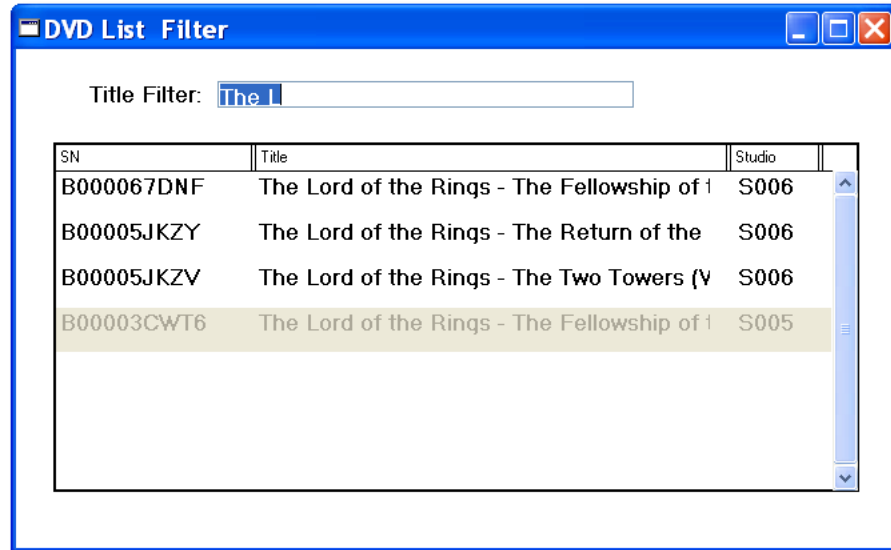
**The L**

then we include that record. So the results include all:

**The Lord of the Rings**

movies.

This kind of range, however, cannot be used to search for text in the middle of a string, or to do more complex matches. For that you need a *Range Expression*.



Multiple Views of the From/To Range

For the sake of convenience, the From/To Range can be entered and viewed in several different places. Here is a comparison for the Range example shown above.

Task 125.1 - DVD List - Variables Range.DVD List - Variables Range

ata View Logic Forms

1	Main Source	1	DVD Titles	Index:	2
2	Parameter	1	pi.Title	Alpha	100
3					
4	Column	1	SN	Alpha	U10
5	Column	2	Title	Alpha	100
6	Column	6	Studio	Alpha	4
7	Column	7	Starring	Alpha	100

Range: 1 To: 1

This is the Range as it appears in the *Data View*.

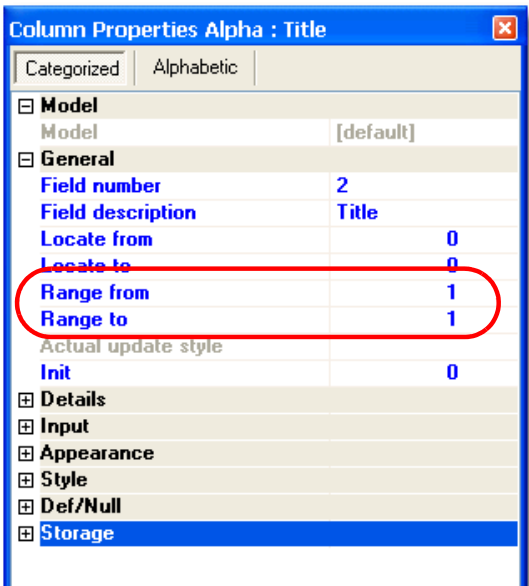
Range Window: 125.1 - DVD List - Variables Range.DVD List -

Range SQL Where Expressions

#	Var	Mode	Exp
1	Title	Equal	1 CndRange(pi.Title<>".Trim(pi.

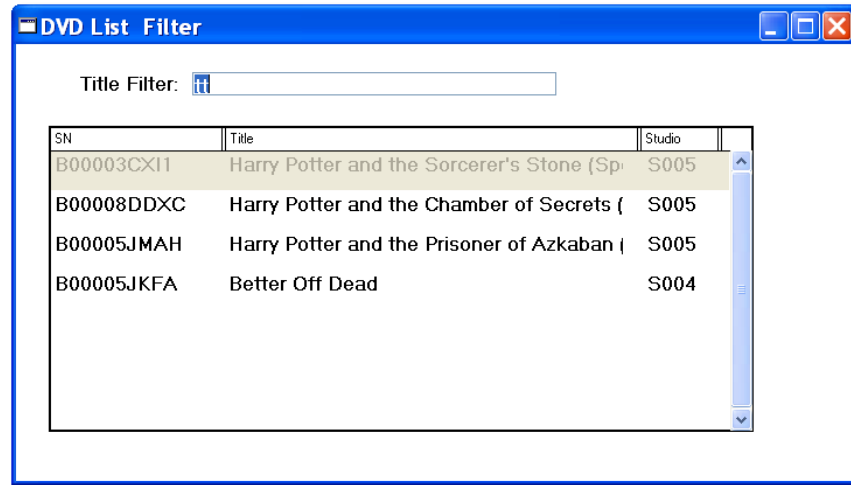
Here is the same range in the *Range Window* (Ctrl+R)

Here it is again in the  
*Column Properties*  
(Alt+Enter).

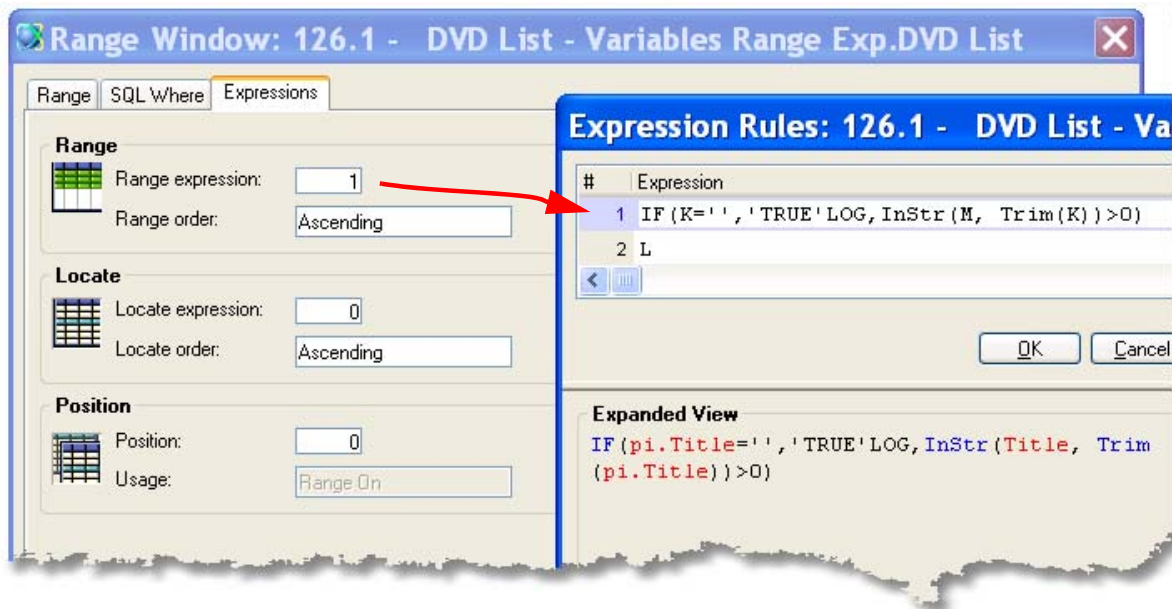


It doesn't matter which option you use; they work alike.

## Using a Range Expression

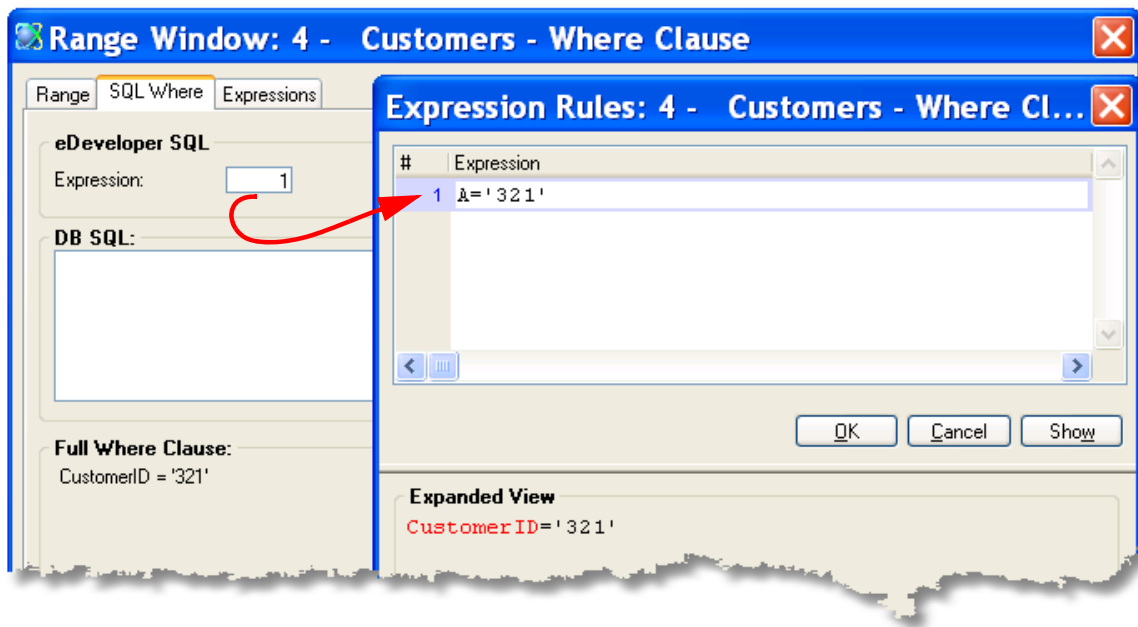


The Range Expression option gives you more flexibility in setting a Range. You can enter any Expression you like, and the record will be select if the Expression evaluates to TRUE. In our example, we are using an expression that will return TRUE if the value is located anywhere in the string.



Here, we have the expression entered into the **Range/Locate->Expressions** tab (Ctrl+R). If the parm is blank, it returns 'TRUE'LOG always, so every record is returned. Otherwise, it uses the **Instr()** function to check if the string is a substring of the current record. If it is, the Instr() function returns true, and that record is included in the view.

## Using an SQL Where clause

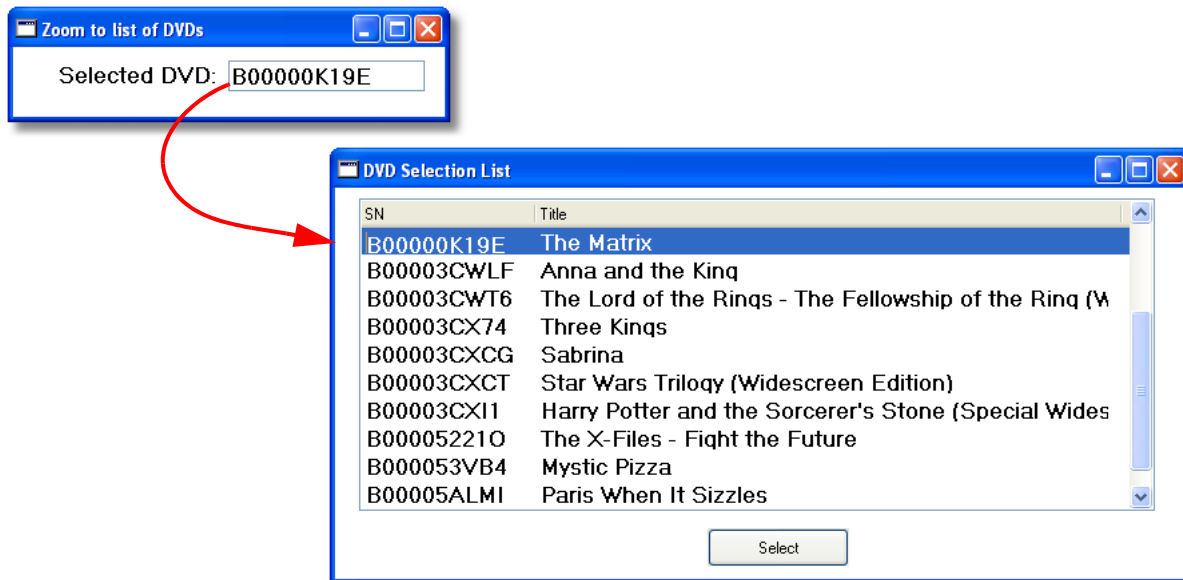


Data View

When you are working with an SQL table, the usual From/To Range options are translated into SQL statements at runtime. However, you can specifically enter SQL code in the *DB SQL* section of the *SQL Where* tab. The disadvantage to this is that SQL code is often DBMS dependent, so it might not be portable if you switch DBMSs.

However, you can alternatively enter an *eDeveloper SQL Expression*, which will be translated into an SQL clause at runtime.

## How do I Make the Task Position on a Certain Record When it Starts?

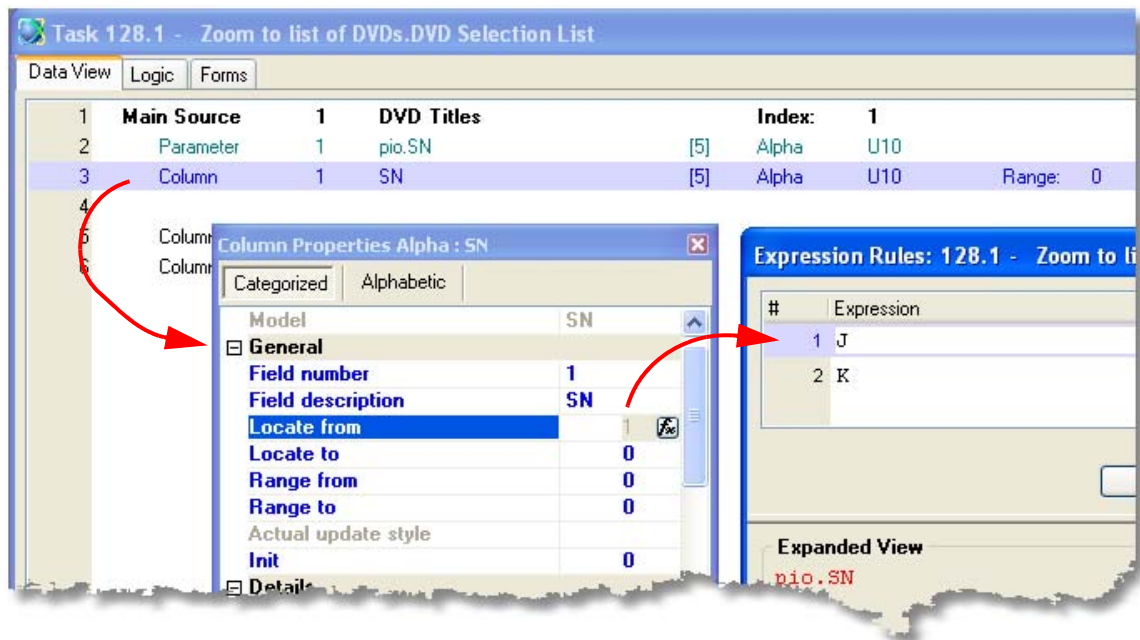


Sometimes you will want to show an entire list of items, but you want the user positioned on one record. For instance, in this example, when the user double-clicks on a field that already specifies a DVD title, they will be positioned on that title.

This is accomplished in eDeveloper by using the *Locate* property. Whereas the *Range* property will restrict the number of records that are displayed, the *Locate* property just changes which record is current.



## Using the Locate property



Data View

1. In the *Data View*, go to the *Column* that you will be positioning on. In this example, we are working with the serial number (SN) field, attempting to locate on the first DVD where the serial number matches the serial number passed in as a parameter.
2. Press **Alt+Enter** to go to the Column Properties, and find the Locate from property.
3. **Zoom** from the Locate From field (or press **fx**) to enter an expression. This expression will represent the data you are trying to locate on.

Now, when the program opens, it will attempt to position on the first record that matches the Locate expression. If the record isn't found, it will position on the next record.

## The effect of Locate Order

The default setting of Locate Order is Ascending. So in our example, the search went from the top of the list to the bottom. We used the *Locate from* property to position on the first record found that matched the criteria.

However, if we had set (**Ctrl+R**) **Range/Locate->Expressions->Locate Order** to Descending, then the search would have proceeded from the bottom of the list on up. In this case, we would have had to set the criteria in the *Locate from* property.

## Using both Locate From and Locate To

If you enter a Locate from and a Locate to, then if the record is not found, the user gets an error message "Record not found - positioned at beginning".

## Using a Locate Expression

If you would like to use a Boolean expression to match a record, you can enter it in **Range/Locate->Expressions->Locate Expression**. This works very similarly to a Range expression, which is discussed in Chapter 20, “Using a Range Expression” on page 514.

## Centering the record

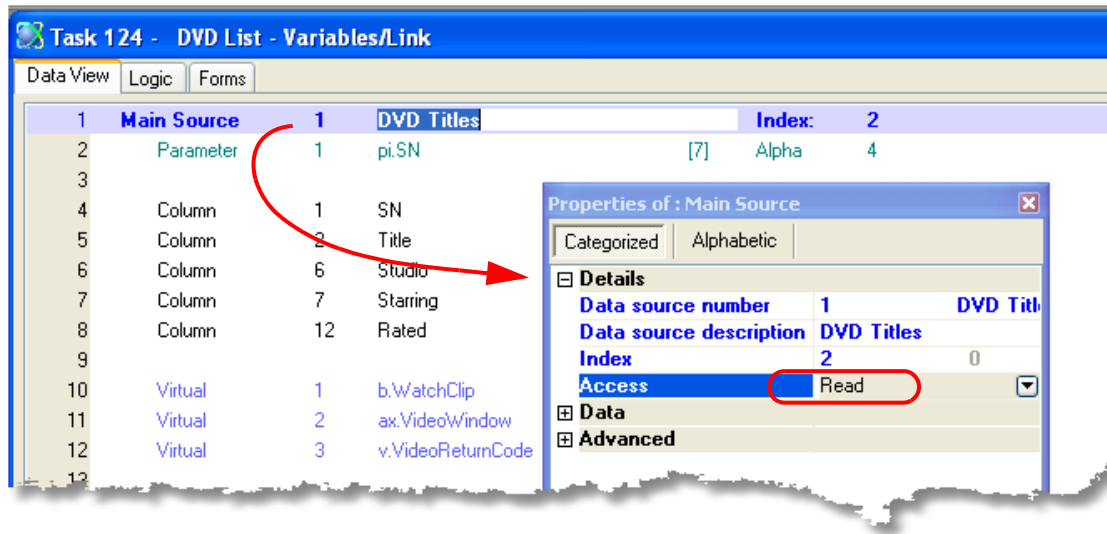


By default, the current record will show up at the top of the window. Sometimes this confuses the user, who may think that there are no other records above. **However, if you set Options->Settings->Environment->Center Screen in Online** to Yes, then the current record will appear in the middle of the window, as shown here.

## How do I Access a Data Source for Read-only in a Task?

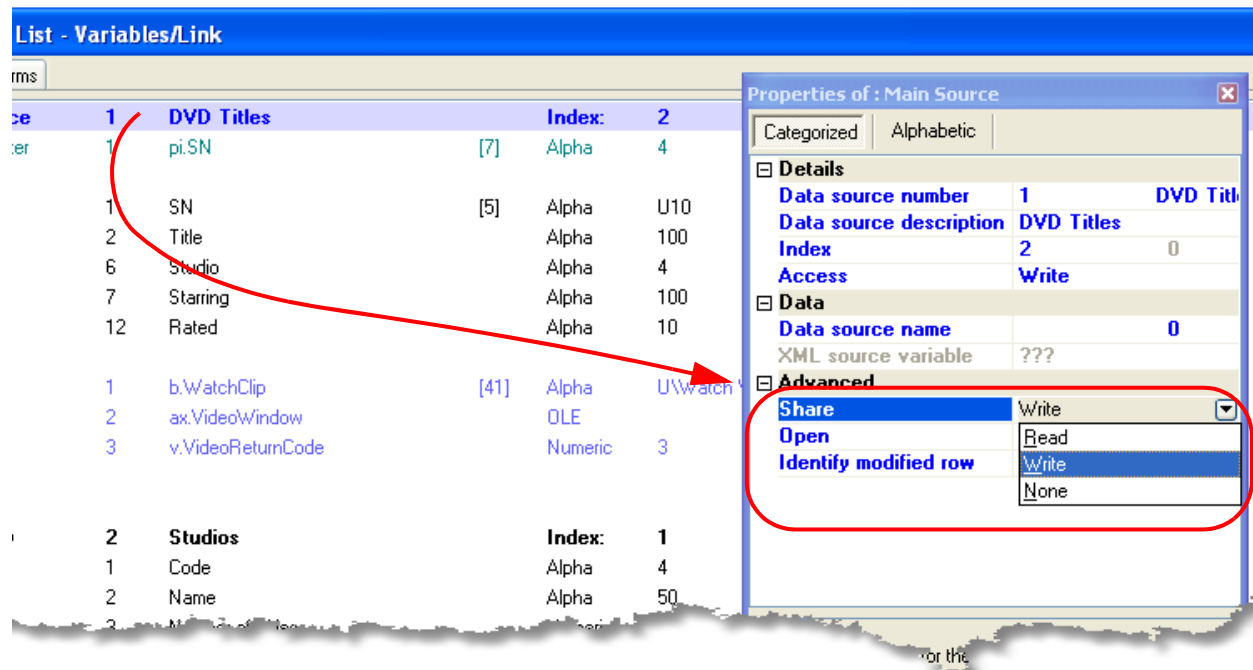
Most of the time when you are using Data sources, you are not updating them, you are reading data out of them to display, print, or use to update other Data sources. If you know you do not have to update a Data sources, it is good practice to open it in read-only. Doing so decreases the potential for record conflicts, and also speeds up the program.

### How to open a Data Source as read-only



1. Go to the *Main Source* or *Linked Source* line in the Data View.
2. Press **Alt+Enter** to go to the Properties Pane.
3. Set the *Access* property to *Read*.

## How do I Determine Access Between Multiple Users for a Data Source, Within a Task?



When multiple users are accessing a Data source, the DBMS, in conjunction with eDeveloper, generally handles the details at the *record* level well. That is, if two users try to access the same record at the same time, one of them gets locked out and gets an error message, and the integrity of the data is kept.

However, there are a few circumstances where you may not want anyone to access the *entire Data source* while updates are being done. This might be the case, for instance, if you are doing end-of-month reconciliation of an accounting table, or doing archiving of old records. You handle this by setting the Data source *Share* property.

The *Share* property works as follows:

- *Write*: Other tasks can open this Data source in access Write while this task is working.
- *Read*: Other tasks can open this Data source in access Read while this task is working, but not in Write.
- *None*: No other tasks can open this Data source while this task is working.

This property is implemented at the DBMS level, so if the DBMS is being accessed by non-eDeveloper programs, the Share value will affect those programs also.

**See also:** Chapter 18, “How do I Access an Existing Database Table?” on page 464.

## How do I Set the Value of a Task Variable?

There are several different ways that variables get modified in eDeveloper. These can be divided into a few categories:

- Values that get initialized automatically when the record is created. These include the *Default Value* properties, and also the *Init* property.
- Values that get moved in as part of the Operations of the program. This would include the Update operation, VARSET, and values that get passed back in arguments.
- Values that get changed by the user, when they type in a value or work with a control.

Here is a summary of how each of the options work.

### Automatically Set Variables

#### The Default Value Property

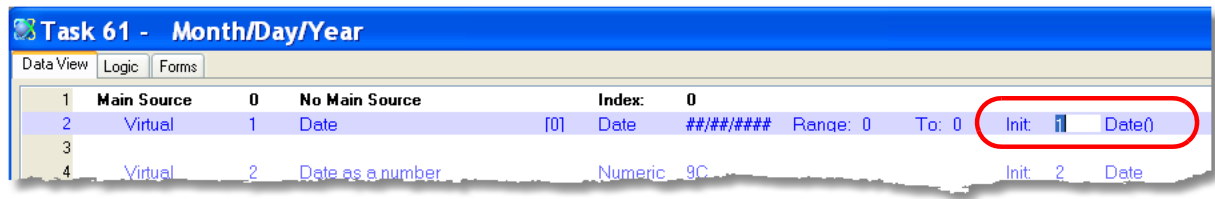
Any field can have a default value property. This property can be assigned at any level: in the field Model, in the Data Source, or within the task. If the field has a default value, then when the value will be used to initialize the field before the user sees the record.

The *Default value* property specifies what value will initialize the field when it is first opened.

The screenshot shows the 'Field Properties Alpha' dialog box with the 'Categorized' tab selected. The 'Details' section is expanded, showing the 'Picture' property set to 'U' and the 'Range' property set to 'Not yet released, Available, Discontinued, X Delete'. The 'Input' section is also expanded, showing the 'Appearance' property set to 'U' and the 'Style' property set to 'A'. The 'Def/Null' section is expanded, showing the 'Null allowed' property set to 'No' and the 'Null value' property set to 'A'. The 'Storage' section is expanded, showing the 'SQL' property set to 'A'. The 'Default value' property is highlighted with a red arrow pointing to it from the text on the left.

Field Properties Alpha	
Categorized	Alphabetic
Model	[default]
[-] Details	
Picture	U
Attribute	Alpha
Range	Not yet released, Available, Discontinued, X Delete
[-] Input	
[-] Appearance	
[-] Style	
[-] Def/Null	
Null allowed	No
Null value	
Null display	
Null default	No
Default value	A
Database default	
[-] Storage	
[-] SQL	

The Init Property



In addition to the Default value property on the field definition, you can also use the Init value when you select the field in the task Data View.

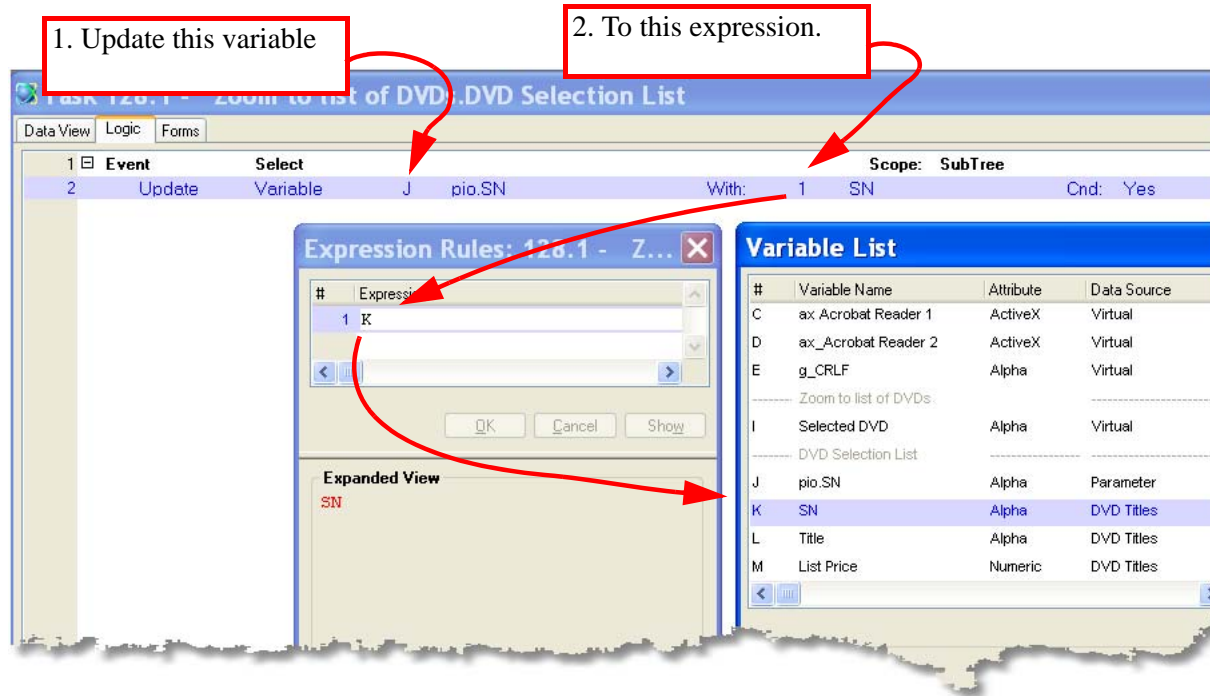
An *Init* works slightly differently from a Default value. Here is a comparison of the two:

Init	Default Value
Initializes a new field	Initializes a new field
Uses an Expression	Uses a hard-coded value
Enables re-calculation, if any of the values in the Expression change.	Only uses the hard-coded value

If you want a field to always be initialized to the same value, the Default Value property is the best way to go.

## Variables updated by the Task

### Update Operation



The most common way that variables are updated in a task is by using the Update Operation. You can enter an Update Operation in any logic unit.

The syntax of the Update Operation is shown above. It can also be explained as:

Update <variable> with <expression> if <Cnd>

The items in brackets are the ones you enter, mainly by zooming to the variables list or Expressions Rules, respectively.

### VARSet()

The **VarSet()** function is one of the few functions that allows you to update an variable directly from within the Expressions Rules. It is very powerful, and has been used for many purposes by developers over the years.

The syntax of VARSet() is:

VARSet(<variable>, <value>)

Where <variable> is entered as a VAR literal. A VAR literal is entered as, say, 'P'VAR, where 'P' is the variable ID. If the variables in a task change position, and 'P' turns to 'Q', then the expression will be automatically updated.

<value> can be any expression that evaluates to the proper value. It can be hard-coded, or it can be a variable, or it can be some function.

So for instance:

```
VARSet ( 'P' VAR , X+6 )
```

Will update the variable P to whatever was in X, plus 6.

VARSet gets more interesting because it can be used to access variables as though they were in an array. So for example:

```
VARSet ( 'P' VAR+1 , X+6 )
```

Will update variable Q to whatever was in X, plus 6.

**Note:** VARSet is used in very specific instances, where a dynamic variable reference is required. Usually, a regular Update operation is used for updating variables.

## Variables changed by the user

Of course, variables can be changed by the user. A variable that can be changed by the user has to be displayed on the form in a Control with the *Modifiable* and *Allow Parking* properties set to Yes.



## How do I Retrieve Data From Multiple Tables in a Single Task?

When you are working with Data sources in eDeveloper, there are two different ways to bring in data from a table:

1. In the *Main source*
2. In a *Linked source*

The *Main source* is the data source that will be looped through in a task. Data from the Main source can be displayed in a table on your form, and you can display one record or the whole table. You can only have one Main source per task.

A *Linked source*, on the other hand, only displays one record at a time. Linked sources are typically used to display information that is associated with whatever record is currently being viewed. You can have as many Linked sources as you like in one task.

**Note:** This does not mean you can only display one table on each form to the user. Even though you can have only one Main source per task, you can still display multiple tables on a form, by using Subforms. Although a Subform looks and acts as though it is part of the parent task, it is in fact getting data from a child task which has it's own Main source.

You retrieve data using a Linked source by using the *Link operation*. Let's see how it's done.

### Using a Link operation

**Task 179 - DVD List - Variables/Link**

Data View Logic Forms

Line	Operation	Data Source	Columns	Index	Direction
1	Main Source	1	DVD Titles	2	
2	Column	1	SN	[6]	Alpha U10
3	Column	2	Title	Alpha	100
4	Column	6	Studio Code	Alpha	4
5	Column	7	Starring	Alpha	100
6	Column	12	Rated	Alpha	10
7					
8					
9	Link Query	2	Studios	1	Direction: Default
10	Column	1	Code	Alpha	4
11	Column	2	Name	Alpha	50
12	Column	3	Number of Titles		4
13	End Link				
14					

1. Create a Link/End link

2. Select the appropriate index.

3. Set up the Locate columns.

4. Select the columns you need.

== Link to get the Studio Name ==

Locate: 1 To: 1

1. First, you create a *Link/End link* pair. Press **F4** to open up a line, then type "L". The *Link* and *End Link* will both be created. Then, choose the *Data source* you want to link to. Here we chose Data source 2, the Studios table. You can type in the *Data source* number, or zoom to select it from a list.

2. Next, tab to the *Index*. The *Index* determines the order in which the table will be searched. It is important that the *Index* match the search criteria you are using to find the record. For instance in this example, if the *Index* is the Studio Code, then the *Locate* should also be on the Studio Code.

When you select an *Index*, any columns that participate in the index will be automatically added to the link.

3. Set up the locate columns (see next section for details on how to do that).
4. Finally, select any other columns you want to include in the link. To do this:
- Press F4 to open up a line. “Column” will be automatically entered for the variable type. Tab to the next field.
  - Zoom to select the column you want, or just type in its number.
  - Rename the column, if you want (see below).l

Hint

Link Query

2

Studios

Index: 1

Column

1

Studios: Code

Alpha 4

Column

2

Studios: Name

Alpha 50

Column

3

Studios: Number of Titles

[0] Numeric 4

End Link

*You can rename the linked variables to make it more obvious which variable belongs to which data source.*

## Setting up the Link Locate columns

Set up the locate columns to point to the column(s) that you want to match.

Main Source		1	DVD Titles	Index:	2
Column	1	SN	[6]	Alpha	U10
Column	2	Title		Alpha	100
Column	6	Studio Code		Alpha	4
Column	7	Starring		Alpha	100
Column	12	Rated		Alpha	10

== Link to get the Studio Name ==

Link Query		2	Studios	Index:	1	Direction:	Default
Column	1	Studios.Code	[0]	Alpha	4	Locate:	1
Column	2	Studios.Name		Alpha	50	To:	1
Column	3	Studios.Number of Titles		Numeric	4		

End Link

Expression Rules: 179 - DVD ...

# Expression

1 N

OK Cancel Show

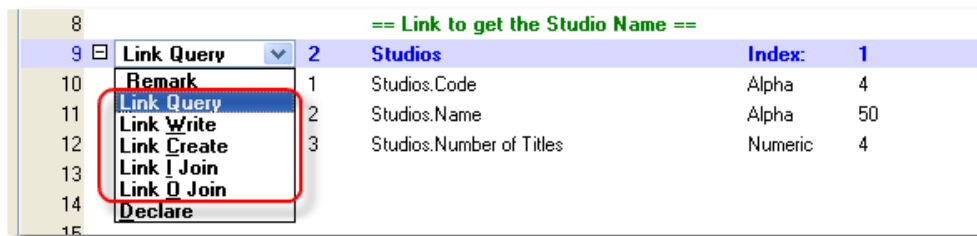
Expanded View

Studio Code

The *Locate* columns in a *Link* work something like the “Select Where” clause in SQL. The column marked *Locate:* is the lower value to match (“locate from”) and the *To:* column is the upper boundary. If they both point to the same value, as in this example, then the *Link* will try to get a record that exactly matches.

So in this example, the *Link Query* will fetch the record where *Studios.Code* matches *Studio Code* of DVD Titles.

## Kinds of links



There are several different types of *Links*, and each of them works slightly differently. All of them bring column variables into the current task, but *Link Write* and *Link Create* will also create records as they do so. Below are the different types and what they are good for.

Link Category	Description	When to use it
<b>Link Query</b>	Fetches an existing record.	To bring in an existing record, or to check if the record exists or not.  For instance, if you want to give an error message if a customer code doesn't exist in the customer table, you would use Link Query.
<b>Link Write</b>	Tries to fetch an existing record, but if the record doesn't exist, it will create it.	When you aren't sure if the record exists yet or not, but you want to create it if it isn't already there.  For instance, if you want to create a phone number record automatically, if the entered phone number doesn't already exist.
<b>Link Create</b>	Tries to create a record. Doesn't check to see if it exists first.	When you are sure the record doesn't exist already, it is faster than a Link Write.  For instance, if you want to create a log record whenever the user opens a certain screen, you could save the user id, date and time stamp when the form opens, and use that in a Link Create.
<b>Link Inner Join</b>	If the joined objects are both SQL tables, this implements an SQL Inner Join.	Wherever you would use an Inner Join in SQL. There are times when this is a faster link, for SQL.
<b>Link Outer Join</b>	If the joined objects are both SQL tables, this implements an SQL Outer Join.	Wherever you would use an Outer Join in SQL. There are times when this is a faster link, for SQL.

## The Link Success indication property

The screenshot shows a table with columns: Index, Link Query, Index, and Index. The table contains three rows of data. A red box highlights the 'Virtual' link query, which is a logical value. An arrow points from this box to the 'Success indication' property in the 'Properties of : Link Operation' dialog box. The dialog box has tabs for 'Categorized' and 'Alphabetic'. The 'Details' section shows the 'Success indication' property set to 'T'. The 'Data' section is expanded, showing the 'Advanced' tab.

Index	Link Query	Index	Index
1	Link Query	2	Studios
1	Column	1	Studios.Code
2	Column	2	Studios.Name
3	Column	3	Studios.Number of Titles
1	Virtual	1	v.Studio found?

Properties of : Link Operation

Details

Data source number	2	Stu
Data source description	Studios	
Index	1	0
Direction	Default	
Success indication	T	
Evaluate Link	Record	
Access	Write	
Condition	Yes	0

Data

Advanced

The link gives a return code to indicate if the link found a matching record or not. This is entered in the *Success indication* property. The *Success indication* flag is commonly used to validate data entered by a user.

To enter a *Success indication* flag:

1. Create a virtual with a *logical* attribute (numeric will work also, but logical values are more maintainable).
2. Zoom from the *Success indication* field, and select your virtual.

## The Link Condition

<b>Main Source</b>	<b>1</b>	<b>DVD Titles</b>		<b>Index:</b>	<b>2</b>		
Column	1	SN	[6]	Alpha	U10		
Column	2	Title		Alpha	100		
Column	6	Studio Code		Alpha	4		
Column	7	Starring		Alpha	100		
Column	12	Rated		Alpha	10		
<b>== Link to get the Studio Name ==</b>							
<b>Link Query</b>	<b>2</b>	<b>Studios</b>		<b>Index:</b>	<b>1</b>	<b>Direction:</b>	<b>Default</b>
Column	1	Studios.Code		Alpha	4	Locate:	1
Column	2	Studios.Name		Alpha	50	To:	1
Column	3	Studios.Number of Titles		Numeric	4		
<b>End Link</b>							
Virtual	1	v.Studio found?		Logical	5		

You can use the *Cnd:* property to prevent the Link from occurring. The *Cnd:* property can be hard-coded to Yes or No, or it can point to an expression. If the value is No or False at runtime, the Link does not occur. In our example, we use the *Cnd:* property to prevent the link if the Studio Code is blank.

## Using Range on Linked columns

There are Range columns in properties for linked columns, just as there are for columns in the *Main Source*. Using Range on the *Main Source* columns will restrain the number of records that are brought into the data view. For instance, if a column in the Main Source is set to:

*Range From:* '01/01/2001'DATE *Range To:* '12/31/2001'DATE

the Range is a set of dates between 01/01/2001 and 12/31/2001, you will only see the records between those two dates.

What is not so obvious is that when you use a Range on a linked source, the effect is very similar. That is, if you used a date range for a linked source of

*Range From:* '01/01/2001'DATE *Range To:* '12/31/2001'DATE

then you will only see Main source records *where the **linked record** has values between those dates.*

This is a very useful feature. However, it also means you have to be careful to not enter your Locate values in the Range section. The two sets of properties work very differently from each other.

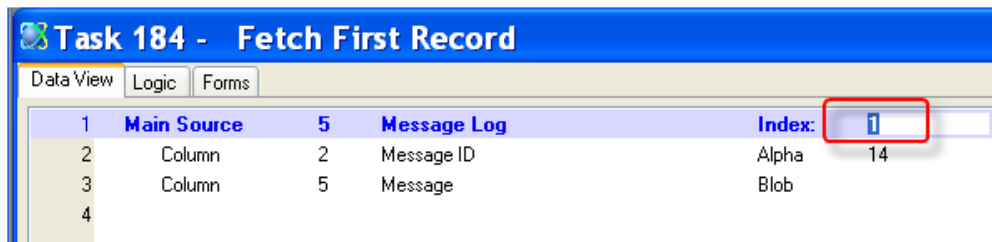
## How do I Fetch the First or Last Record of a Table in a Task?

Sometimes you may want to only fetch one record in a task, say, the first or last record of a table. For instance, if you want to update the “last login time” for a particular user, or find the greatest record number in a table.

There are a few steps in doing this, which we will go through below. The methods are different for a Main Source than a Linked Source, however, so there are two sections.

### Fetching the first or last record for a Main Source

#### 1. Define your index



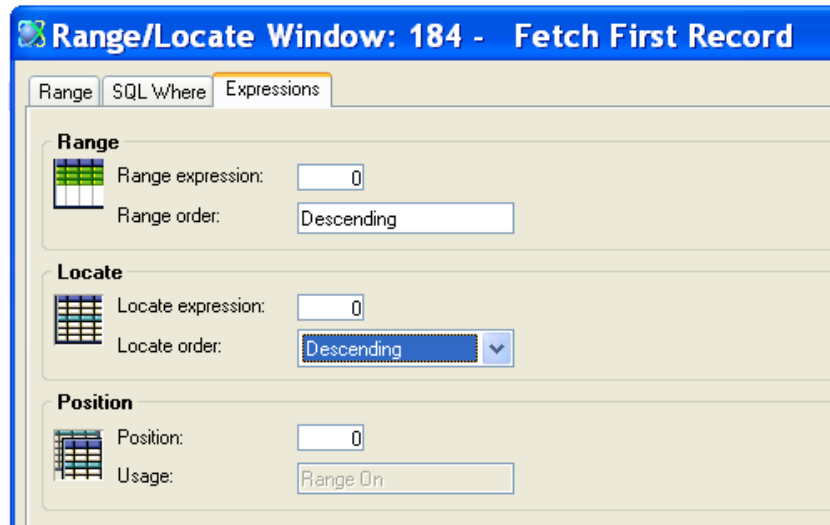
The first thing to think about when fetching the first or last record, is what exactly are you meaning by “first” and “last”. That is, for any given table, there might be multiple indices. For instance, in a Customer table, the “first customer” might mean the one with the smallest customer ID, or the one with the first alphabetical Customer Name, or the one with the smallest zip code.

In eDeveloper, you define which index you are using by selecting the index in the *Main Source* operation.

However, if you do not have an index that will define the search order the way you want, you can use the task’s *Sort Repository* to perform a runtime sort of the data.

You do not need to enter *Range* or *Locate* values in the Data View to get the first or last record.

## 2. Set up the search direction

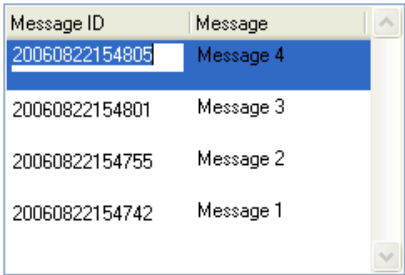
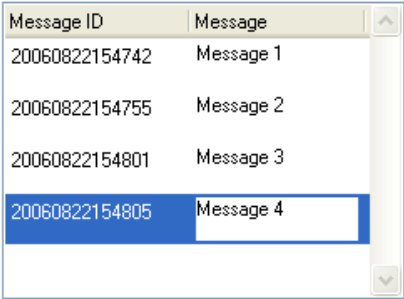


Next, you need to determine whether you are looking for the first or last record. If you are looking for the first record, you don't need to do anything here. However, if you are looking for the last record, then you need to tell eDeveloper to start at the bottom of the table (since it would be very inefficient to search all the way through the table to find the last one!).

To get the last record, you set the *Range order* or *Locate order* in **Range/Locate->Expressions** to *Descending*.



Range Order vs. Locate Order

Range Order	Locate Order
	
The table is searched in reverse order, so Message 4 comes first.	The table is in the same order, but eDeveloper parks on the last record.

When you are fetching just the last record, either *Range order* or *Locate order* will work. They function slightly differently, as you can see above. In an online task the differences are obvious: in a one-cycle batch task such as this one the result will be the same.

However, it is a common practice to just set both to *Descending* in this sort of one-cycle task, as it seems easier to understand.

### 3. Check the number of cycles

When you are fetching just one record, you will be using a batch task that cycles only once.

You set this up in **Task Properties** (**Ctrl+P**), on the **General** tab.

Set *Task Type* to Batch.

Set *End Task condition*, to Yes.

Set *Evaluate condition* to After updating record.

**Task Properties: 184 - Fetch First Record**

General Behavior Interface Data Options Advanced

**Task Information**

Task name : Fetch First Record

Task type : Batch

Initial mode : Modify Exp :

End task condition : Yes

Evaluate condition : After updating record

Return value : 0

Selection table : No

Resident task : No

Task ID :

Source file name: Prg\_287.xml

OK Cancel

Fetching the first or last record for a Linked Source

1. Define your index

3					
4	<input type="checkbox"/> Link Query	5	Message Log	Index: 1	Direction Reversed
5	Column	2	Message ID	Alpha	14
6	Column	5	Message	Blob	
7	End Link				
8					
9					

The first thing to think about when fetching the first or last record, is what exactly are you meaning by “first” and “last”. That is, for any given table, there might be multiple indices. For instance, in a Customer table, the “first customer” might mean the one with the smallest customer ID, or the one with the first alphabetical Customer Name, or the one with the smallest zip code.

2. Set up the search direction

3					
4	<input type="checkbox"/> Link Query	5	Message Log	Index: 1	Direction Reversed
5	Column	2	Message ID	Alpha	14
6	Column	5	Message	Blob	
7	End Link				
8					
9					

Next, you need to set the search direction. If you want to get the first record, set *Direction* to Default. If you want the last record, set *Direction* to Reversed.

That’s all you need to do. You don’t need to enter a *Range* or *Locate* value.



---

## Chapter 21: Expressions

---

### How do I Format an Expression in the Expression Window?

Some expressions can be very long and complex. For instance, suppose you have a nested IF statement:

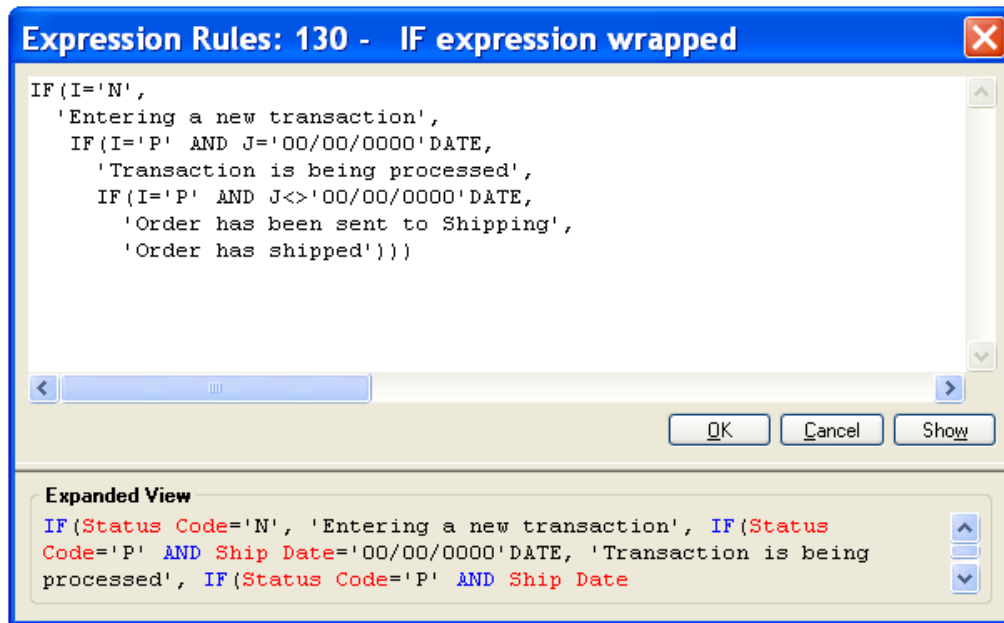
```
IF(I='N', 'Entering a new transaction', IF(I='P' and  
J='00/00/0000'DATE, 'Transaction is being processed',  
IF(I='P' and J<>'00/0000'DATE,'Order has been sent to  
Shipping', 'Order has shipped')))
```

While this is syntactically correct, it is difficult for a human to read. It is much easier to read if you add some line breaks and spacing:

```
IF(I='N',  
    'Entering a new transaction',  
    IF(I='P' and J='00/00/0000'DATE,  
        'Transaction is being processed',  
        IF(I='P' and J<>'00/0000'DATE,  
            'Order has been sent to Shipping',  
            'Order has shipped')))
```

Here is how you format an expression in eDeveloper.

## Formatting an Expression



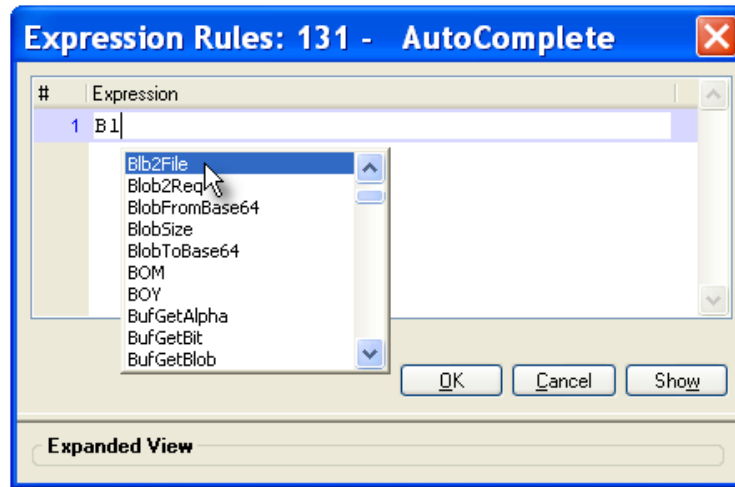
1. **Zoom** (**F5** or double click) to the *Expressions Rules*. Alternatively, you can press **Ctrl+E** to jump to the *Expression Rules* from anywhere.
2. Press **F6** (**Edit->Wide**). This will cause the Expression you are parked on to expand, so it is the only Expression in the Expression window.
3. Now that you are in Wide mode, you can press **Enter**, and a line break will appear in the expression. You can also add spaces to indent text as desired, or use tabs.
4. When you are done, press **Ctrl+Enter**, the OK button, the X box on the upper right, or press Escape twice. Pressing "Enter" to leave won't work in Wide mode, because the Enter key is interpreted as a line feed.

Note that when you add line breaks, this does in fact add a CRLF to the expression. This doesn't matter to eDeveloper, but if you are putting text out to, say, a message box, adding a linefeed to the text will cause the message to have a linefeed at runtime too. This allows you to format text for output purposes as well as for programmer readability.

## How do I Automatically Complete the Name of a Typed-in Function?

To help save you keystrokes, eDeveloper has a nice auto-complete feature for functions. When you are entering the function, you can allow eDeveloper to enter most of the text.

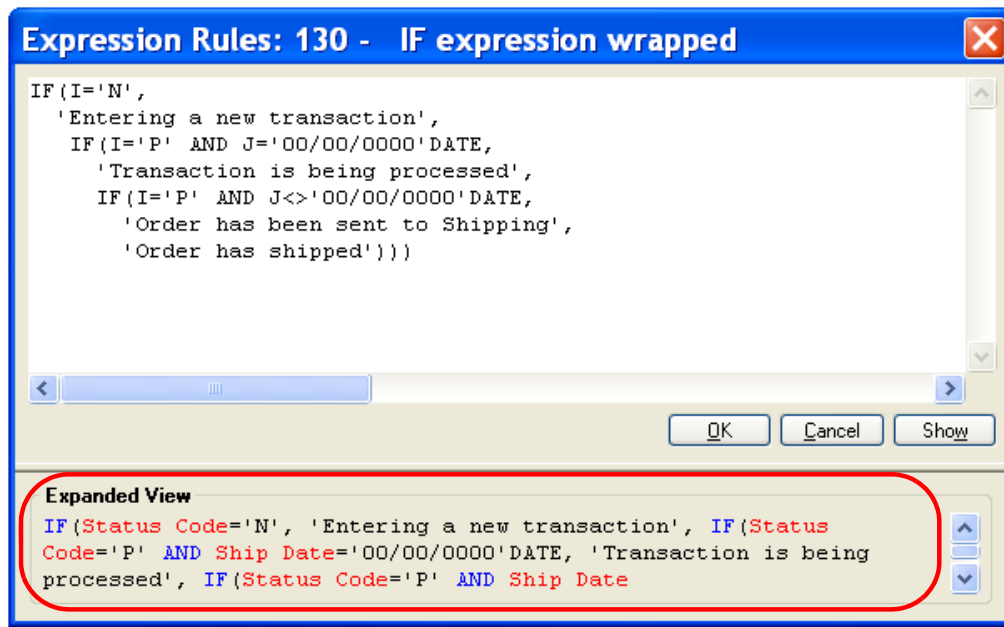
### Using the Auto-complete feature



1. Type in the first few characters of the function you want. Here we typed “bl”. Case doesn’t matter.
2. Press **Ctrl+Space**. A list of functions will appear, with the selection bar located on the first function that matches what you typed.
3. Move the cursor down to the function you want. You can do this by using the down-arrow key, or just continue typing the expression and the selection bar will move automatically.
4. When you are positioned on the function you want, press **Tab**. The function will be filled in, complete with the first parentheses.

**Note:** If the characters match only one function, the function will be automatically inserted with no list box.

## How do I Set the Colors of the Colored Elements in the Expanded Expression View?



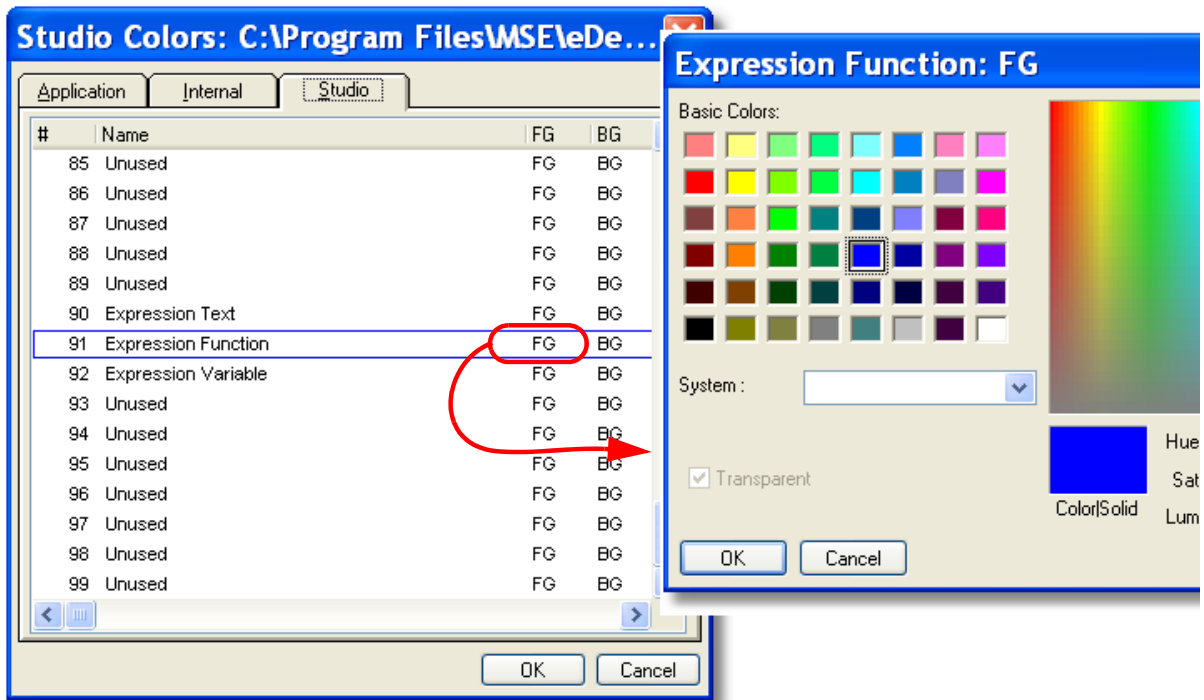
When you view the Expanded View of an expression, the text is colored. Three colors are used here:

- Expression Text
- Expression Function
- Expression Variable

You can set these colors to anything you like. Here's how.



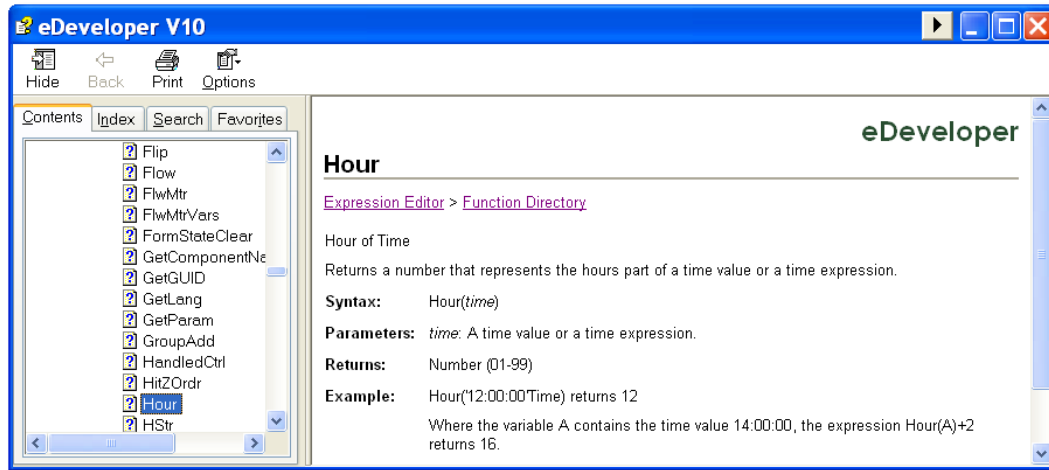
## Setting the Expression Colors



1. Go to **Options->Settings->Colors**.
2. Click on the **Studio** Tab.
3. Go to lines 90-92. For each color, there are two columns. *FG* sets the foreground color, *BG* sets the background color.
4. **Zoom** on the column you wish to change. A Windows color picker will be shown. Select the color you like, then press OK.

There are two special colors on the color picker. *Transparent* causes the color to show up as black in the color picker, but it picks up whatever the background color is when it is used. *System* colors inherit the color from Windows.

## How do I get to the Help Page of a Function?



When you are programming in any language, it's difficult to remember how all the functions work. Fortunately eDeveloper makes it easy to find a good help page while you are coding. There are two ways to do this.

### Finding a Help Page for a function from anywhere

You can always get to the help page for a function from anywhere in the Studio.

1. Press **F1**. This brings up context-sensitive Help.
2. Click on the *Index* tab.
3. Type in the name of the function.

You will jump to the help page for that function. Also, you can use the Search tab to find all references to the function.

**Hint:** You can leave the Help screen open while you are coding, especially while you are learning eDeveloper, to make it quicker to look things up.

### Finding Help from the Expressions List

In addition, you can jump directly to the help page for a function, from the Expression containing the function.

1. Go to the Expression list (**Ctrl+E**).
2. Position the cursor on the function
3. Press **F1**.

Now you will be positioned on the Help page for the function you were parked on.

### Finding Help from the Functions list

One really useful way to enter functions is the Functions list. Not only can you find the Help page easily, but when you select the function, the parentheses and parameter placeholders are entered automatically, making it easier to enter the function.

For more details on how to do this, see Chapter 21, “Selecting a Function from the Functions List” on page 550.

## How do I Manually Expand the Expression Line to Make it Easier to Edit Large Expressions?

When you are editing an expression, pressing **F6** puts the editor into Wide mode. In this mode, you only see the one expression in the window, and you can also use linefeeds and spaces to format the expression of you want. See Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537 for more details.

---

## How do I Syntax Check an Expression while in Wide Mode?

When you have entered an expression while not in Wide mode, pressing the Tab key check the expression. It will cause a bell to sound and an error message to be displayed on the status line, if the expression is in error. This is useful because you can check the expression before you do a complete syntax check on the program.

However, when you are in Wide mode, pressing Tab actually inserts a Tab character into the expression. So, when you are in Wide mode there is another key combination, **Ctrl+Enter** (**Edit->Confirm Expression**).

**See also:** Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537.

## How do I Enter Line Breaks into String Values?

When you are editing string values in the expression list, you can format those string values as you would in a text editor. That is, you can enter line feeds, and spaces.

In order to do this, you need to be in *Wide* mode first. This opens up the Expression in a little text-editing box. You can read more about this in Chapter 21, “How do I Format an Expression in the Expression Window?” on page 537.

# How do I Set an Expression Directly in the Task Editor?

All expression for a task are held on one table, the expression list. This makes them easy to find and to re-use.

However, in the task editor, you can enter short expressions directly on the line that will use them. This saves you time, especially for small expressions

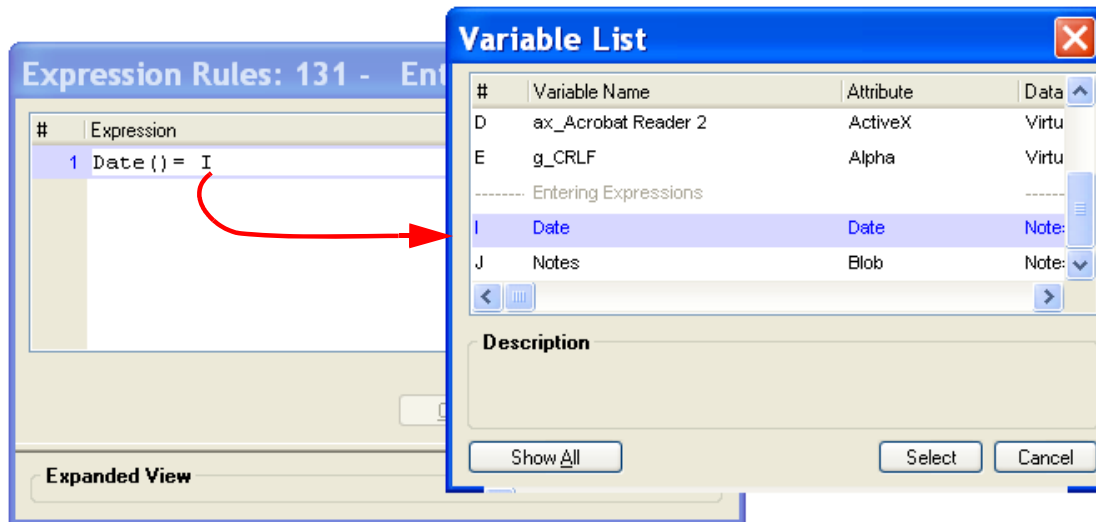
## Entering a Quick Expression

1	SN	[5]	Alpha	U10				
2	Title		Alpha	100				
5	Release date	[0]	Date	####/####	Range: 3	To: 0	Init: 0	
6	Studio		Alpha	4	Range: 2	To: 2	Date()	
14	Amazon page		Alpha	200				

1. Go to a field that requires an expression, such as the Init or Cnd columns in the Logic or Data View editors, or on a property sheet.
2. Type an equal sign ('=').
3. Continue typing the expression.
4. When you are finished, press **Enter** or **Esc** to save the Expression. **Ctrl+F2** will cause you to leave without saving.

Now, if the Expression you entered does not exist in the Expression list, then a new Expression will be automatically created, and the number will be brought back to the Expression field. However, if the Expression already exists, then the existing Expression will automatically be re-used.

## How do I Open the Variable List from the Expression Editor?



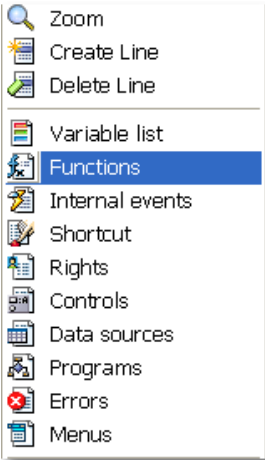
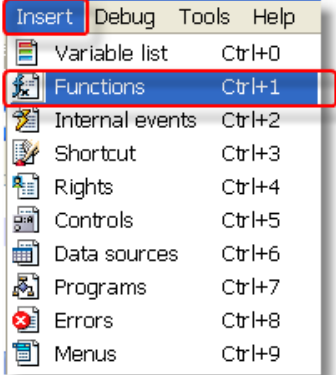
1. In the *Expression Editor*, position the cursor where you want to have the variable.
2. Press **F5**.
3. The *Variable list* will appear. You can locate the variable you want by:
  - Use the arrow keys to move up and down
  - use Locate (**Ctrl+L**) to find a variable in a longer list
  - or type the first letters of the variable and you will be automatically positioned on the match.
4. When you find the variable you want, press **Enter** or the **Select** button to bring the variable back into your expression.



# How do I Open the Functions List from the Expression Editor?

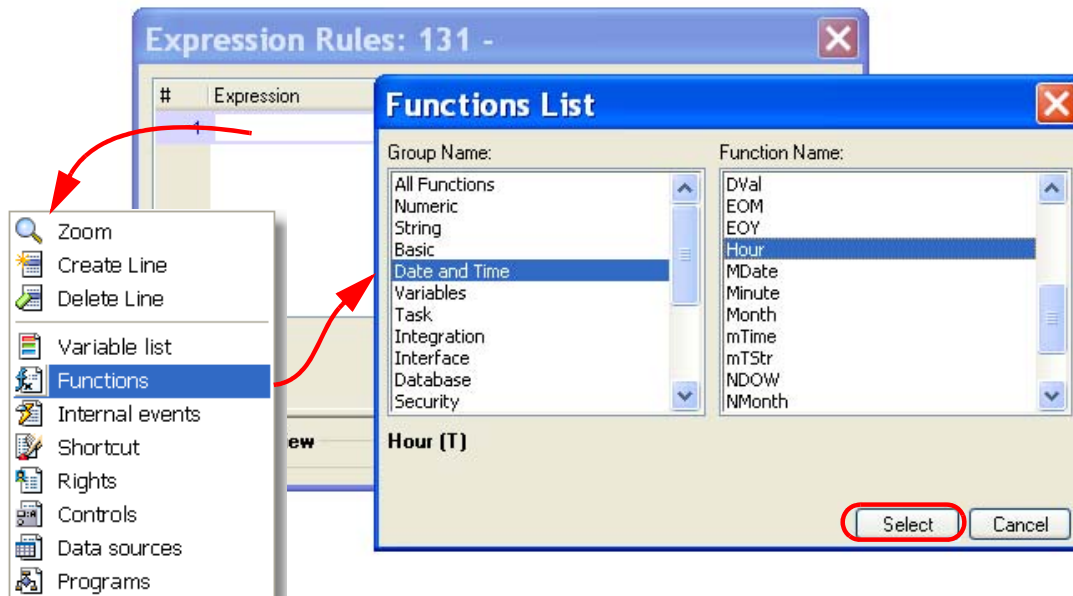
While you are in the *Expression Editor*, you can bring up the *Functions list* by:

- Pressing **Ctrl+1**.
- Selecting it from the **right-click** context menu.
- Selecting it from the overhead menu: **Insert->Functions**.

Context Menu	Insert->Functions
	

Once the *Functions list* is open, you select the function by following the steps below.

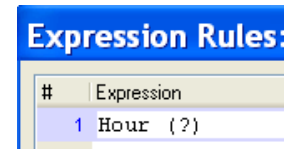
## Selecting a Function from the Functions List



1. In the *Expression* list, position the cursor on the spot where you want the function to go.
2. Select *Functions* from the **right-click** menu.
3. At this point, you will see a list of function by Group. This is extremely useful, because if you don't know what function you are looking for, you can find the general category on the left, and see all the functions for that category on the right.

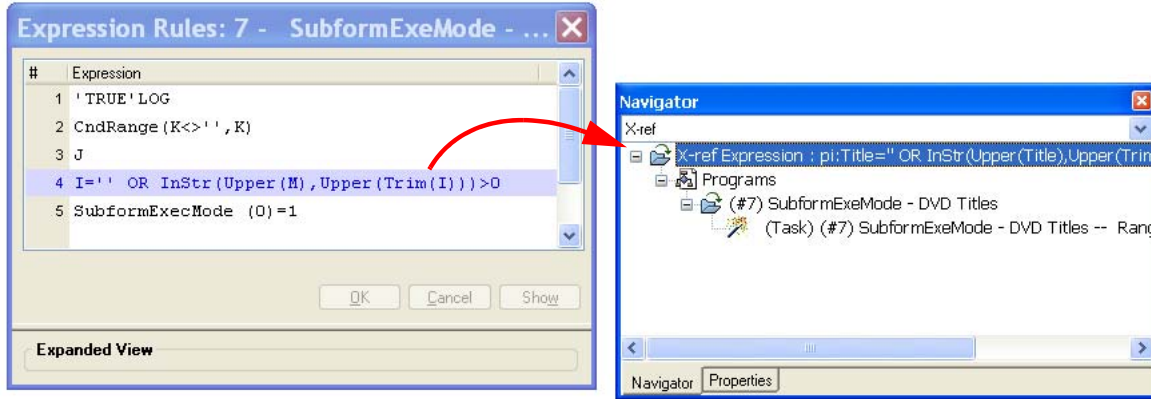
You can position yourself in either column by typing in the first characters of what you are looking for.

4. Once you are positioned on the function you are interested in, press F1 to view the help page for that function. If you aren't sure what function you wanted, you can repeat this process until you find what you were looking for.
5. When you have the function you want, press Enter or the Select button.
6. Now, the function will be written into the Expression, along with the parameter list.



## How do I Find Where an Expression is Being Used?

If you are considering changing an expression, you first need to know where it is used. This is very simple using the **Find Reference (Ctrl+F)** command.



### Using Find Reference on an Expression

1. Position the cursor on the expression you are investigating.
2. Press **Ctrl+F**, (or **Edit->Find and Replace->Find Reference**).
3. A popup box will appear, which you can ignore in this instance, just press OK. Since expression are only used within one program, there isn't a choice to be made.
4. You will then get an **X-ref list** in the Navigator. The lowest level of the tree shows where the expression is used. Clicking on it will position you on the place in the program were the expression is used.

## How do I Re-use Expressions Within Another Expression?

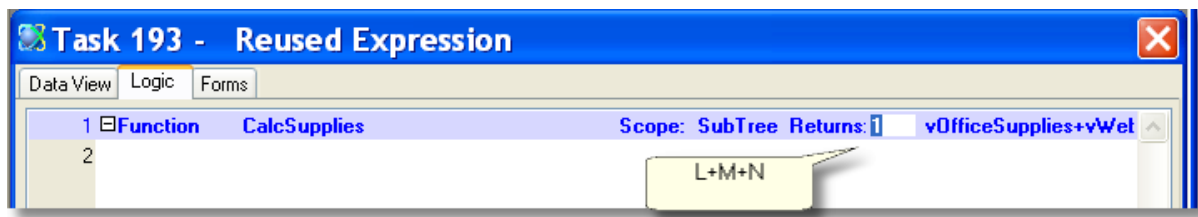
One of the nice things about the expression list is that one expression can be re-used in many places within a task. Commonly the same expression is used in many calculations, especially expressions used to reset variables, such as 0, 'TRUE'LOG, 'FALSE'LOG, or DATE().

However, you can create a simple user function that returns the expression, and use that in other expressions.

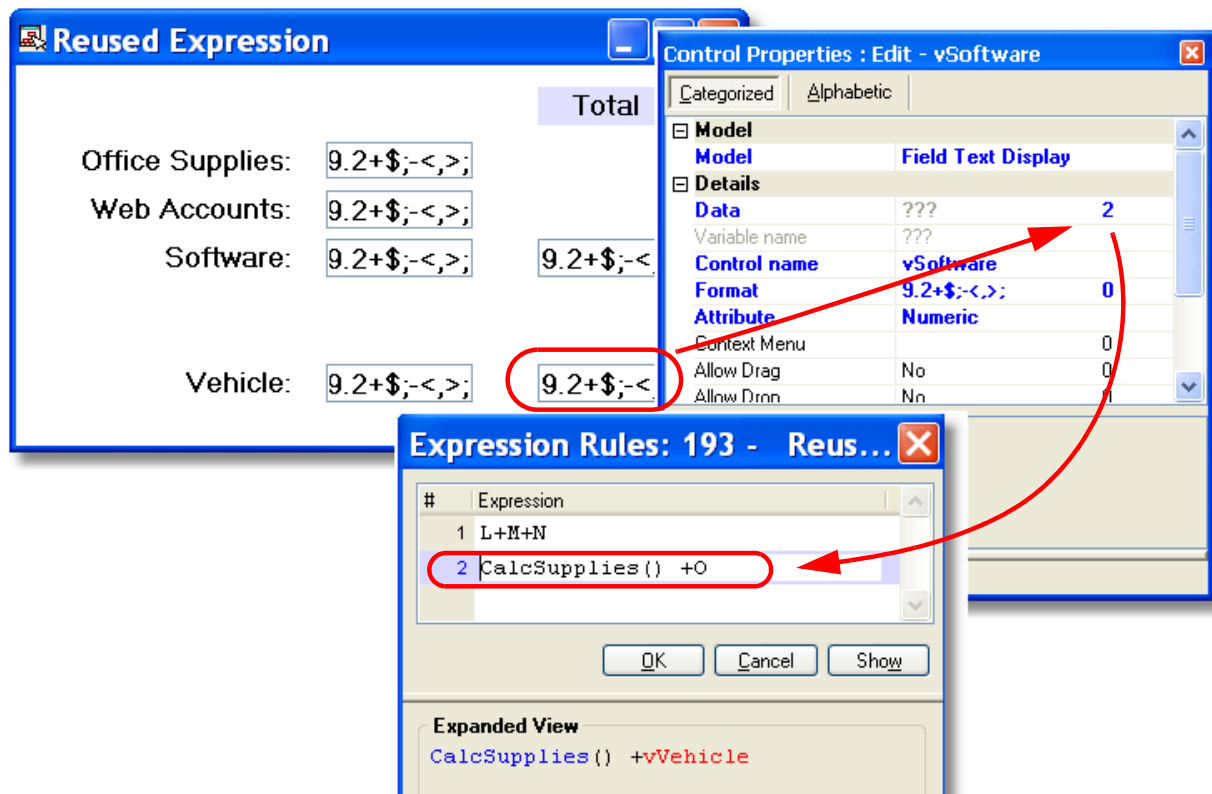
Alternatively, you can also use one expression within another expression, by using the **ExpCalc()** function. **ExpCalc()** allows you to enter a complex function once, and reuse it in another expression.

We will show you samples of both methods.

## Using a User Expression

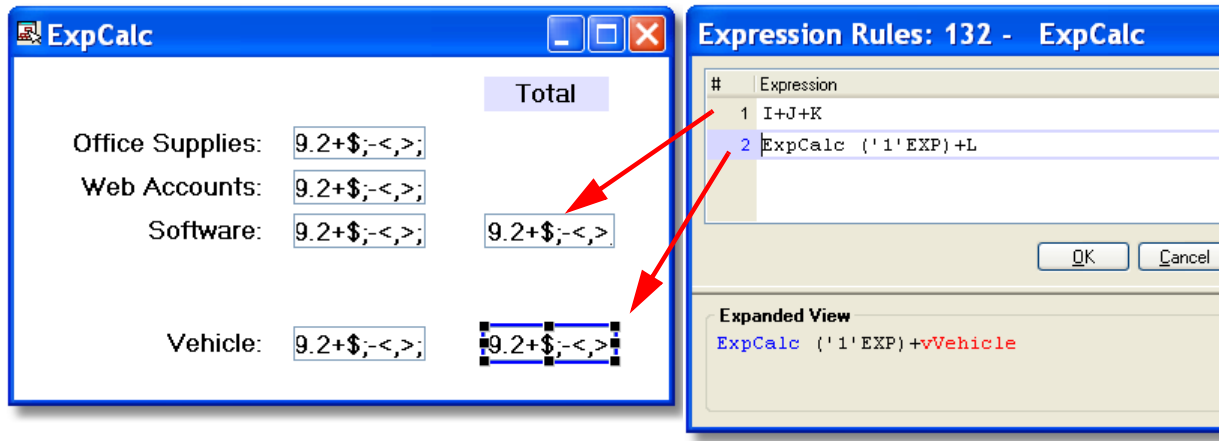


1. First, create a user expression that returns the expression. In our example, we name our function CalcSupplies, and it returns the expression L+M+N.



2. Now, you can use your function anywhere you would otherwise use the expression.

## Using ExpCalc()



**ExpCalc()** allows you to use the results of one expression inside another expression. For instance, in this example, three fields are totalled in expression #1, and displayed onscreen as a subtotal. The subtotal is then used inside expression #2, and the fourth field added on to it.

1. Enter **ExpCalc(**
2. Enter the number of the expression you want calculated, using the EXP literal. In this example, we are looking at expression #1, so we enter **'1'EXP**.
3. Add the final paren: **)**.

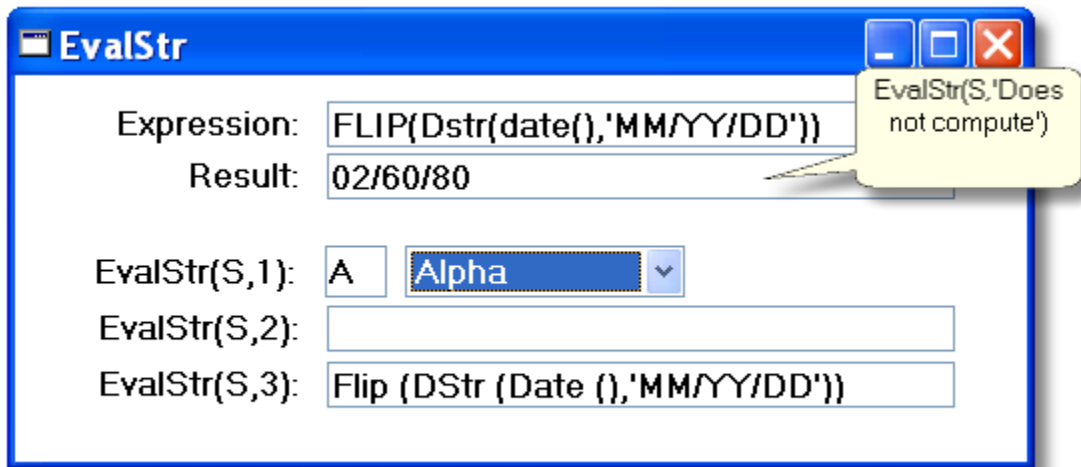
The EXP literal is what allows eDeveloper to keep track of the expression if it moves. For instance, if we added another expression in front of expression #1, then the '1'EXP would automatically change to '2'EXP.

## How do I Construct and Evaluate an Expression at Runtime?

There may be occasions where you want to construct an eDeveloper expression at runtime, then execute it. This is useful when you want to store “macro” instructions in a DB Table, for instance. The instructions can be created by one program, and executed by another.

**EvalStr()** allows you to do just that. It is a very powerful function that can execute any other eDeveloper function. In this test case, you can type in any text you like, and eDeveloper will execute it.

### Using EvalStr



**EvalStr(expression string, default value)** takes two parameters:

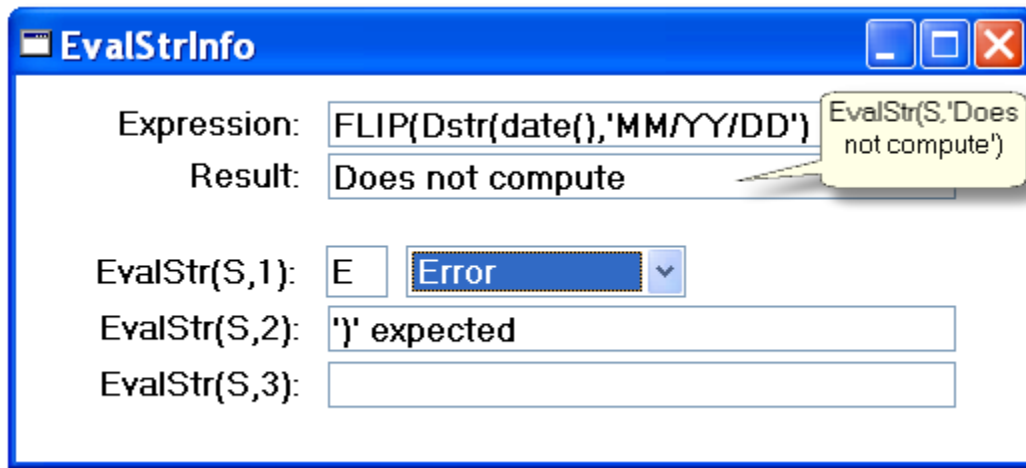
- **expression string** is the expression to be evaluated. It can be a literal string, in single quotes, or an alpha variable.
- **default value** is the value that will be sent back if there is a syntax error in the expression. In our example, if the alpha string doesn't work, the string 'Does not compute' is returned.

Note that there are some issues here. First, you have to ensure that the datatype of the result matches what you are expecting. The eDeveloper syntax checker has no way of knowing what the expression will evaluate to at runtime.

Also, there is no syntax checking of the string until runtime. If we typed in an invalid expression, the default value would be returned, but we wouldn't know what the error was.

You can handle these issues using the **EvalStrInfo()** function, described below.

## Using EvalStrInfo



**EvalStr(expression string, option)** takes two parameters:

- **string** is the expression to be evaluated. It can be a literal string, in single quotes, or an alpha variable.
- **option** is a number, 1, 2, or 3:
  - 1 = returns the Attribute of the expression
  - 2 = returns the parser error, if any
  - 3 = returns the result expression

*Option 1* returns a letter representing the attribute result of the expression: A for Alpha, N for Numeric, etc. It also returns E for Error, or \* for unknown type.

*Option 2* returns the parser error. In this example, we have entered an expression that isn't correct, and the error message is: ' )' expected.

*Option 3* returns the parsed expression, if there is no error.



## How do I Repeat Existing Expression in a Task?

If you have an expression in the Expression list and want to repeat it, you have three options:

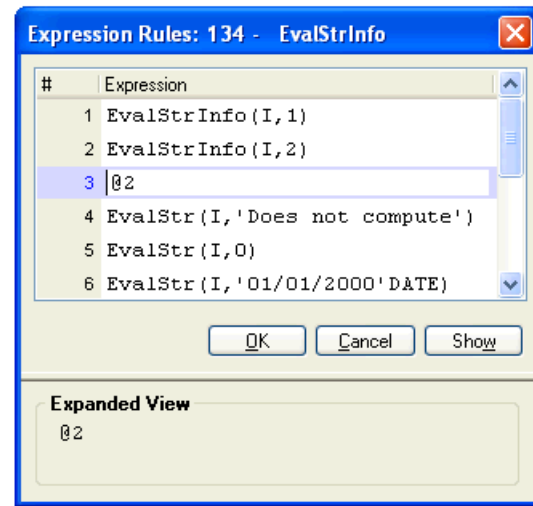
1. Copy and paste the text using the usual Windows **Ctrl+C** and **Ctrl+V**.
2. Using the usual eDeveloper function **Edit->Entries->Repeat Entry (Ctrl+Shift+R)**.
3. Use the *Repeat line* option on the Expression list, **@Line**.

Repeat Entry is explained in Chapter 1, “How do I Repeat an Entry in the Studio?” on page 12.

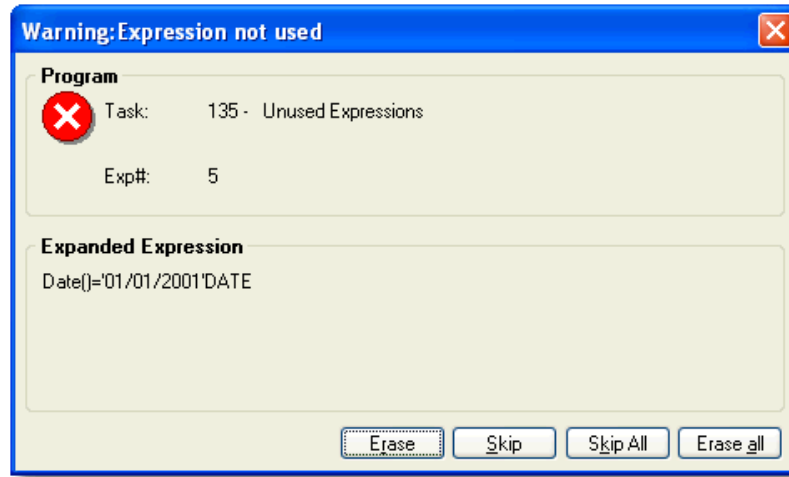
### Using Repeat Line (@)

1. Press F4 to open up a new line in the Expression Rules.
2. Type an @ sign.
3. Type the number of the expression you want to repeat.
4. Press Tab.

The expression will be copied. In this case, expression #2 will be copied into expression #3.



## How do I Find All Unused Expressions?



Having extra expressions in your programs makes maintenance more difficult, so it is a good idea to keep them cleaned out. Fortunately, eDeveloper makes this easy.

### Finding unused expressions

1. Set **Options->Settings->Environment->Preferences->Studio Checker Minimal Level** to Warnings or Recommendations.
2. Go to the program you want to check. (or to the header line at the top of the Program Repository to check all programs.)
3. Press **F8** (or **Alt+F8** to check the entire repository).
4. The warning screen shown above will appear for each unused expression. As each appears, you can decide:
  - To ignore this expression and go on to the next (Press Skip).
  - To ignore all unused expressions (Press Skip All).
  - To erase this expression (Press Erase).
  - To go look at the expression (Press Escape). This option stops the syntax check and brings you directly to the unused expression in the Expression Editor.
  - To erase all expressions (Press Erase All).

In addition, the unused expressions will appear on the Checker pane.

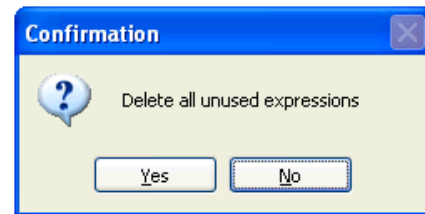
## How do I Clear All Unused Expressions?



To delete unused expressions, follow the instructions in Chapter 21, “How do I Find All Unused Expressions?” on page 558.

Then, when the first warning message appears, press the Erase all button. All the unused expressions will disappear.

In addition, if you are doing a syntax check of the entire Program Repository (**Alt + F8**), then you will get a prompt such as the one on the right. If you answer Yes, then the unused expressions will be automatically deleted without further prompting.



## How do I Define a String Value in an Expression?

When you are entering an expression in eDeveloper, all strings need to be in single quotes. Otherwise, the alpha characters are assumed to be variables or functions.

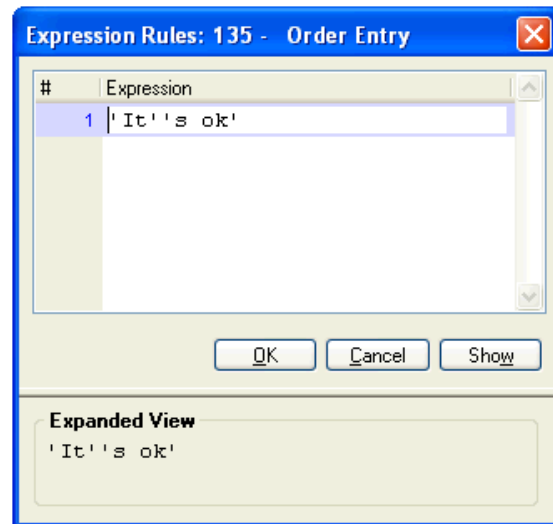
If you need to enter a single quote *inside* an expression, you can use two single quotes.

In this example:

**'It's ok'**

will display as

**It's ok**



## How do I Avoid Creating Duplicate Simple Expressions?

It is easy, especially when working on very complex tasks, to end up with several expressions that are all identical. For instance, there might be four expressions that are all 'TRUE'LOG or 0.

However, if you enter your expressions using **Quick Expressions** then if the expression already exists, that is the expression entry that will be used. A new entry will only be created if it does not already exist.

See Chapter 21, "How do I Set an Expression Directly in the Task Editor?" on page 547 for how to enter a Quick Expression.



## Chapter 22: Reports

---

### How do I Dynamically Export the Dataview of a Task into an XML, HTML, Text, or CSV file?

One of the more common user requests is to have data dumped into a flat file so it can be read into a spreadsheet, word processor, or other software. Conveniently, eDeveloper has nice built-in functionality to do just that, so you don't even have to do special programming.

There are two ways to do this.

- *The DataView functions:* Internally to the program, you can use the **DataViewTo** functions from within the program to do the export programmatically. These functions give you, the programmer, most of the control over what is printed and in what format. See Chapter 22, “How do I Export Data Into an HTML File?” on page 572.
- *The Print data utility:* Or, you can allow the end-user to access the *Print data* utility from any online screen. This utility gives most of the control to the user. See Chapter 22, “How do I Allow the End-user to Dynamically Export Data?” on page 564.

Either method gives the same variety of output options: text, CVS, XML, or HTML.

## How do I Allow the End-user to Dynamically Export Data?

The screenshot shows a window titled "Print Studio Records". Inside, there is a table with the following data:

Code	Name	State	Zip
S001	Twentieth Century Fox Home Video	CA	94025
S002	Buena Vista Home Video	CA	95128
S003	Universal Studios	CA	66044
S004	Paramount	TN	95428
S005	Warner Home Video	CA	97330
S006	New Line Home Entertainment	CA	94609

Below the table, there is a "Print (Ctrl+P)" button. To the right of the button, there are two input fields: "Phone" with the value "408 496-7223" and "City" with the value "Menlo Park".

You can allow the user to dynamically export data from any screen, using the *Print data* utility. This utility works much like a report generator. It allows the user to choose the format of the output, which fields are included, and in what order the fields will be printed.

**See also:** Chapter 22, “How do I Export Data Into an HTML File?” on page 572.

### Creating the data view program

1. Create a task that lists all the records and fields the user is likely to be interested in. The quickest way to do that is to create a Browse program with the *Ctrl+G* program generator (See Chapter 22, “How do I Create a Quick Browse Program?” on page 567). In this program, set **Task Properties->Options->Print Data** to Yes.
2. Now, when the program is run, the user can use the runtime *Print data* utility (**Ctrl+P**, or **Options->Print data**) to print any data that shows on the form, as explained below.



## Using the Print Data Utility

Now, the user can export data from your data view. Here is how the user would do it.

1. Run the program.
2. When the list of records comes up, use the *Sort* and *Range* options as desired to create a subset of records in the desired sort order.

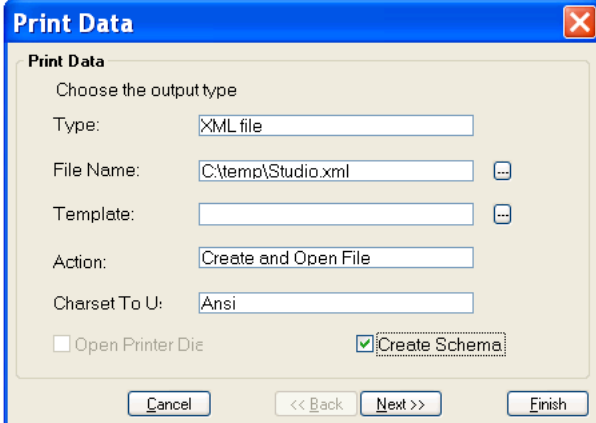


The 'Print Studio Records' dialog box displays a table of records with columns: Code, Name, State, and Zip. The records are as follows:

Code	Name	State	Zip
S001	Twentieth Century Fox Home Video	CA	94025
S002	Buena Vista Home Video	CA	95128
S003	Universal Studios	CA	66044
S004	Paramount	TN	95428
S005	Warner Home Video	CA	97330
S006	New Line Home Entertainment	CA	94609

Below the table, there is a 'Print (Ctrl+P)' button and two text fields: 'Phone' with the value '408 496-7223' and 'City' with the value 'Menlo Park'.

3. Press *Ctrl+P* (**Options->Print Data**). The Print Data dialog box will appear. Select the options according to what kind of output you need, and press **Next >>**



The 'Print Data' dialog box is used to configure the output type and file name. It includes the following fields and options:

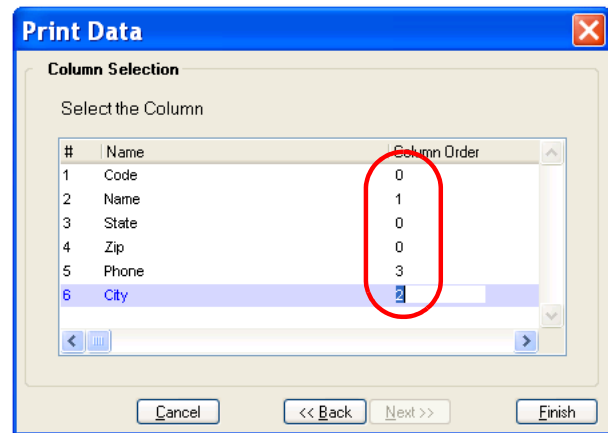
- Choose the output type**
- Type: XML file
- File Name: C:\temp\Studio.xml
- Template: (empty)
- Action: Create and Open File
- Charset To U: Ansi
- ☐ Open Printer Die
- ☒ Create Schema

At the bottom, there are buttons for 'Cancel', '<< Back', 'Next >>', and 'Finish'.

- Your next dialog will show a list of all the columns in the data view. If you don't want a column to print, set the Column Order for that column to zero. Otherwise, you can renumber the columns to change the order on the output file.

In this case, we are printing Name, City, and Phone, in that order.

Then press **Finish**.



- Your data will now be output to the file. Since we chose an XML file, the file is in XML format, but it could also have been HTML or CSV.

Since we chose "Create and Open", the XML will be opened by whatever program is used for XML on this computer.

```
<Print_data xsi:schemaLocation="urn:edevveloper.printdata C:\
- <Record>
  <Name>Twentieth Century Fox Home Video</Name>
  <City>Menlo Park</City>
  <Phone>408 496-7223</Phone>
</Record>
- <Record>
  <Name>Buena Vista Home Video</Name>
  <City>San Jose</City>
  <Phone>408 286-2428</Phone>
</Record>
```

**Hint:** You can save your data as type=XML, but use '.xls' as the file name extension. Then it will be opened in Excel using the Excel XML formatting, which gives you some nice formatting and header options.

	A	B	C
1	ns1:Name	ns1:City	ns1:Phone
2	Twentieth Century Fox Home Vide	Sort Ascending	408 496-7223
3	Buena Vista Home Video	Sort Descending	408 286-2428
4	Universal Studios	(All)	415 935-4228
5	Paramount	(Top 10...)	615 297-2723
6	Warner Home Video	(Custom...)	415 843-2991
7	New Line Home Entertainment	Menlo Park	415 836-7128
8	*	Nashville	
9		Oakland	
10		San Francisco	
		San Jose	

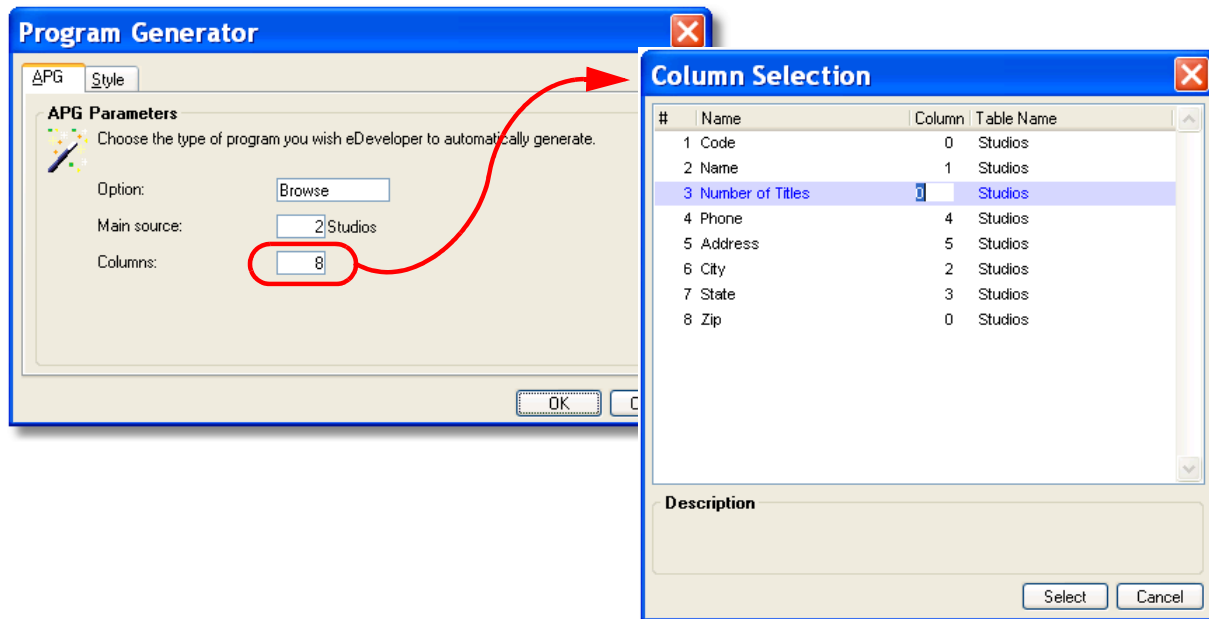
## How do I Create a Quick Browse Program?

It is very easy to create an eDeveloper program from scratch. However, you can create a program even more easily by using the *Generate program (Ctrl+G)* utility.

These simple browse programs make a very good start for a *Data view* program, to allow the user to do a Range and/or Sort before dumping records to a file.

### Generating a simple browse program

1. Open up a new line in the Program repository by pressing **F4** (**Edit->Create Line**).
2. Press **Ctrl+G** (**Options->Generate Program**).



3. The *Program Generator* dialog box will appear. Select:

*Option:* Browse

*Main source:* Whatever data source you want to export. You can **zoom** to select from a list.

*Columns:* Zoom here to select which columns will export. By default, all columns will export in the order they are in the file, but in this example, we are only exporting Name, City, State, Phone, an Address, in that order.

4. Click **OK**.

Your program is now created, and when you run it, it will show you every record from the main source. You can edit the program as you would any eDeveloper program, changing the form and adding ranges to limit which records are displayed.

## How do I Dump the Current View to a Text File in HTML, XML or Simple Delimited Format?

You can use the **DataViewTo** functions to export your current Data view to a text file. There are three different DataView functions, each of which is explained in its own section:

- **DataViewToHTML()** : Chapter 22, “How do I Export Data Into an HTML File?” on page 572
- **DataViewToXML()** : Chapter 22, “How do I Export Data Into an XML File?” on page 576
- **DataViewToText()** : Chapter 22, “How do I Export Data into a Text File?” on page 569 and Chapter 22, “How do I export Data Into a CSV File?” on page 571

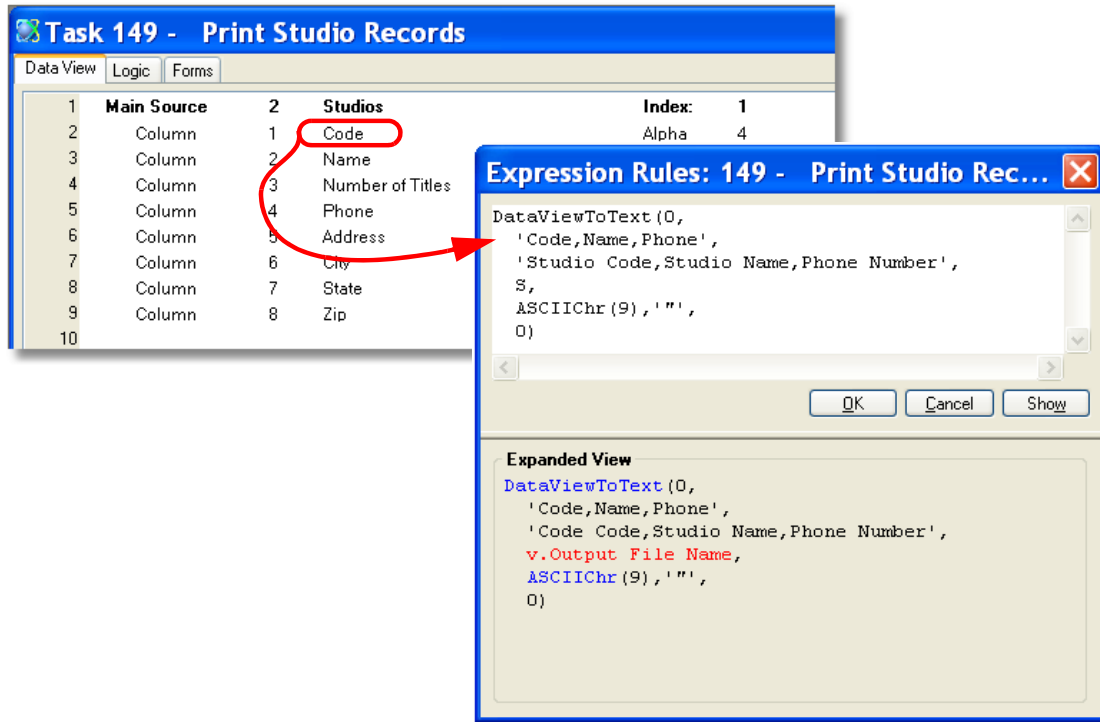
Alternatively, you can allow the end user to dynamically export their current data view without doing any specific programming: see Chapter 22, “How do I Allow the End-user to Dynamically Export Data?” on page 564.

**See also:** Chapter 22, “How do I Dynamically Export the Dataview of a Task into an XML, HTML, Text, or CSV file?” on page 563.  
Chapter 5, “How do I Create Tasks that Dump Data Records Into Flat Text Files and Vice Versa?” on page 97.

## How do I Export Data into a Text File?

You can use the **DataViewToText()** function to export data from your program into a text file. This function gives you, as the programmer, the most control over the export, so you can, if you choose, limit which records can be exported.

### Using DataViewToText()



**DataViewToText()** will export the current data view, either in the current task or its ancestors. The syntax is:

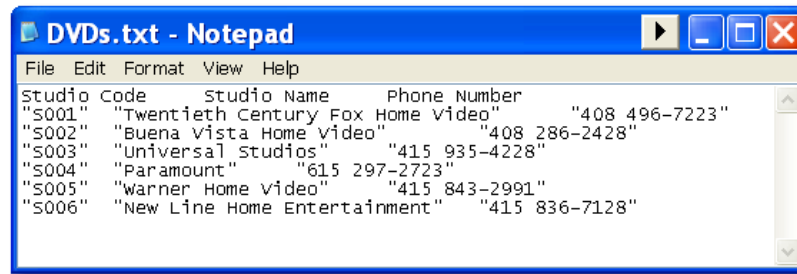
**DataViewToText(Generation, VariableList, HeaderList, DelimiterString, StringIdentifier, CharSet)**

Where:

- **Generation** is the generation number. Zero for the current task, 1 for the task's parent, etc.
- **VariableList** is a list of the variable names you want to export. This is the text name of the variable, as it appears in the Data View section of the task. The names are separated by commas (no spaces between variable names, and case matters).
- **HeaderList** is a list of titles that will be used in the export file. In our example, we used "Studio Code" for the "Code" field, to make the output more readable. If the string is empty, no headers will be sent. If the string is '@', the variable list will be used as the header list.
- **DelimiterString** is the string that will separate the values. In our example, we used `ASCIIChr(9)` to use the Tab character.

- **StringIdentifier** is a string that will print before and after a string in the output. In our example, each string is surrounded by double quotes.
- **CharSet** is a number that sets the character set to be used: 0=ANSI, 1=Unicode, 2=UTF-8.

The result from our sample is shown below. The tab character doesn't show, but you can see how it has moved the data into specific columns.



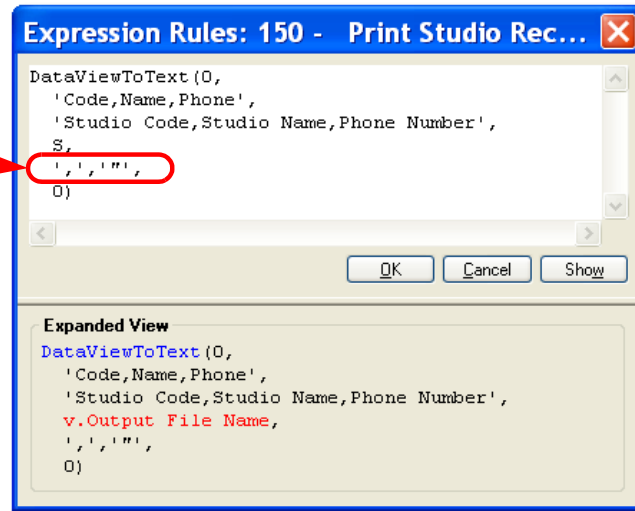
```
File Edit Format View Help
Studio Code      Studio Name      Phone Number
"S001"  "Twentieth Century Fox Home Video"  "408 496-7223"
"S002"  "Buena Vista Home Video"  "408 286-2428"
"S003"  "Universal Studios"  "415 935-4228"
"S004"  "Paramount"  "615 297-2723"
"S005"  "Warner Home Video"  "415 843-2991"
"S006"  "New Line Home Entertainment"  "415 836-7128"
```

**See also:** Chapter 5, “How do I Create Tasks that Dump Data Records Into Flat Text Files and Vice Versa?” on page 97.  
Chapter 22, “How do I Allow the End-user to Dynamically Export Data?” on page 564.

## How do I export Data Into a CSV File?

You can use the **DataViewToText()** function to export data from your program into a CSV file. All you need to do is use a comma for the delimiter, and double-quotes for the string identifier, as shown here.

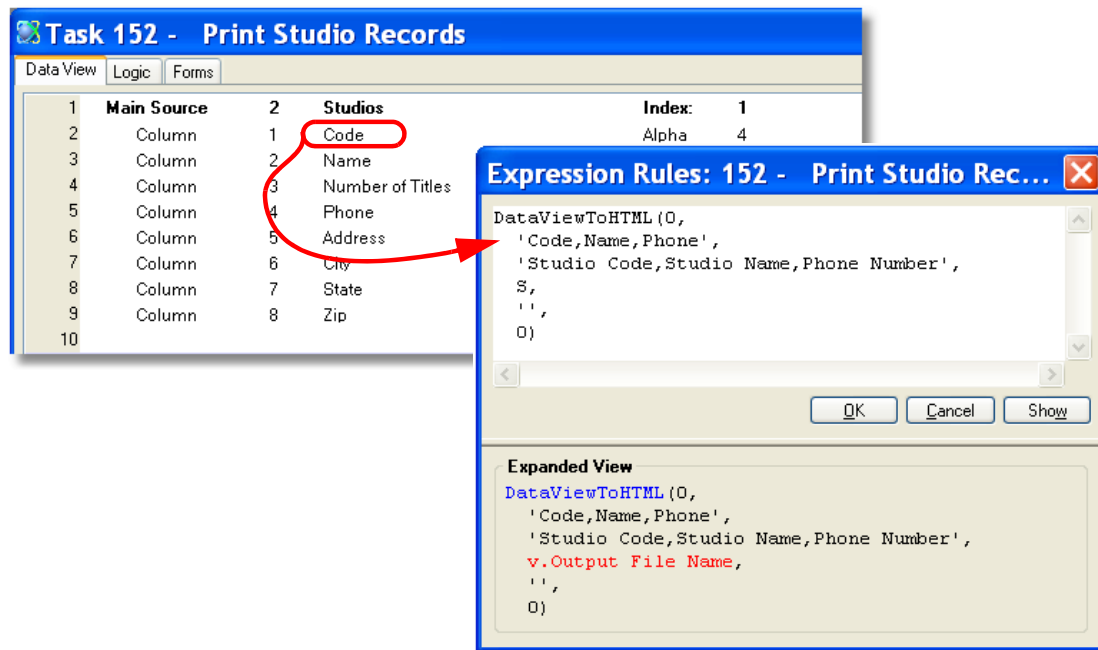
For details on using DataViewToText(), see Chapter 22, “How do I Export Data into a Text File?” on page 569.



## How do I Export Data Into an HTML File?

You can use the **DataViewToHTML()** function to export data from your program into a text file. This function gives you, as the programmer, the most control over the export, so you can, if you choose, limit which records can be exported.

### Using DataViewToText()



**DataViewToHTML()** will export the current data view, either in the current task or its ancestors. The syntax is:

```
DataViewToHTML(Generation, VariableList, HeaderList, Template-File, CharSet)
```

Where:

- **Generation** is the generation number. Zero for the current task, 1 for the task's parent, etc.
- **VariableList** is a list of the variable names you want to export. This is the text name of the variable, as it appears in the Data View section of the task. The names are separated by commas (no spaces between variable names, and case matters).
- **HeaderList** is a list of titles that will be used in the export file. In our example, we used "Studio Code" for the "Code" field, to make the output more readable. If the string is empty, no headers will be sent. If the string is '@', the variable list will be used as the header list.
- **TemplateFile** is the name of a template file that can be used to create the HTML file (optional).
- **CharSet** is a number that sets the character set to be used: 0=ANSI, 1=Unicode, 2=UTF-8.



The result from our sample is shown below.



The screenshot shows a web browser window titled "C:\Temp\DVDs.htm - Microsoft Internet Explorer". The browser displays a table with three columns: Studio Code, Studio Name, and Phone Number. The table contains six rows of data for various studios.

Studio Code	Studio Name	Phone Number
S001	Twentieth Century Fox Home Video	408 496-7223
S002	Buena Vista Home Video	408 286-2428
S003	Universal Studios	415 935-4228
S004	Paramount	615 297-2723
S005	Warner Home Video	415 843-2991
S006	New Line Home Entertainment	415 836-7128

### Using the HTML Template File

If you want, you can specify an HTML template file when you export your data. The HTML template file needs to follow the format of the minimal HTML file shown on the left. The `<MGTABLE>` tag will be replaced during the export with the HTML table text.

Minimal template	Customized template
<pre> 1 &lt;html&gt; 2 &lt;head&gt; 3 &lt;title&gt; &lt;/title&gt; 4 &lt;/head&gt; 5 &lt;MGTABLE&gt; 6 &lt;/html&gt; 7 </pre>	<pre> &lt;html&gt; &lt;head&gt;   &lt;title&gt;My DVD List&lt;/title&gt;   &lt;meta http-equiv="Content-Type"   content="text/html; charset=iso-8859-1"&gt;   &lt;!-- This page should not be cached --&gt;   &lt;meta http-equiv="Expires" content="Mon, 06 Jan 1990 00:00:01 GMT"&gt;   &lt;meta http-equiv="pragma" content="no-cache"&gt;   &lt;meta http-equiv="cache-control" content="0"&gt;   &lt;meta http-equiv="refresh" content="100"&gt;   &lt;link rel="stylesheet" href="DVDStyles.css" type="text/css"&gt; &lt;/head&gt; &lt;MGTABLE RowStyle=ALL ColumnStyle=ALL&gt; &lt;/html&gt; </pre>
<p>This is the minimum template. <code>&lt;MGTABLE&gt;</code> will be replaced at runtime with the actual table.</p>	<p>Here is our customized template. We've added some meta tags, and more importantly, a link to a stylesheet.</p>

### MGTABLE Styles

The `<MGTABLE>` tag has two style tags you can use in your template, `RowStyle` and `ColumnStyle`. These determine how the style tags are attached to the generated table.

**RowStyle:**

All	eDeveloper creates a specific style for each row and title.
EvenAndOdd	eDeveloper creates two styles, an MG_Even_Row and an MG_Odd_Row.
Equal	eDeveloper creates the same style for all rows and titles.

**Column Style:**

All	eDeveloper creates a specific style for each column.
Equal	eDeveloper creates the same style for all columns.

So in our example, we get an HTML table that looks like the one below. Note all the **class=** tags. You can use those to customize the look of the table.

```
<table border="1" width="100%" class="MG_TABLE">
<THEAD>
  <tr>
    <td width="33%" class="MG_TITLE1">Studio Code</td>
    <td width="33%" class="MG_TITLE2">Studio Name</td>
    <td width="33%" class="MG_TITLE3">Phone Number</td>
  </tr>
</THEAD>
<TBODY>
<tr class="MG_ROW1">
  <td width="33%" class="MG_DATA1">S001 </td>
  <td width="33%" class="MG_DATA2">Twentieth Century Fox Home Video </td>
  <td width="33%" class="MG_DATA3">408 496-7223 </td>
</tr>
<tr class="MG_ROW2">
  <td width="33%" class="MG_DATA1">S002 </td>
  <td width="33%" class="MG_DATA2">Buena Vista Home Video </td>
  <td width="33%" class="MG_DATA3">408 286-2428 </td>
</tr>
<tr class="MG_ROW3">
  <td width="33%" class="MG_DATA1">S003 </td>
  <td width="33%" class="MG_DATA2">Universal Studios </td>
  <td width="33%" class="MG_DATA3">415 935-4228 </td>
```

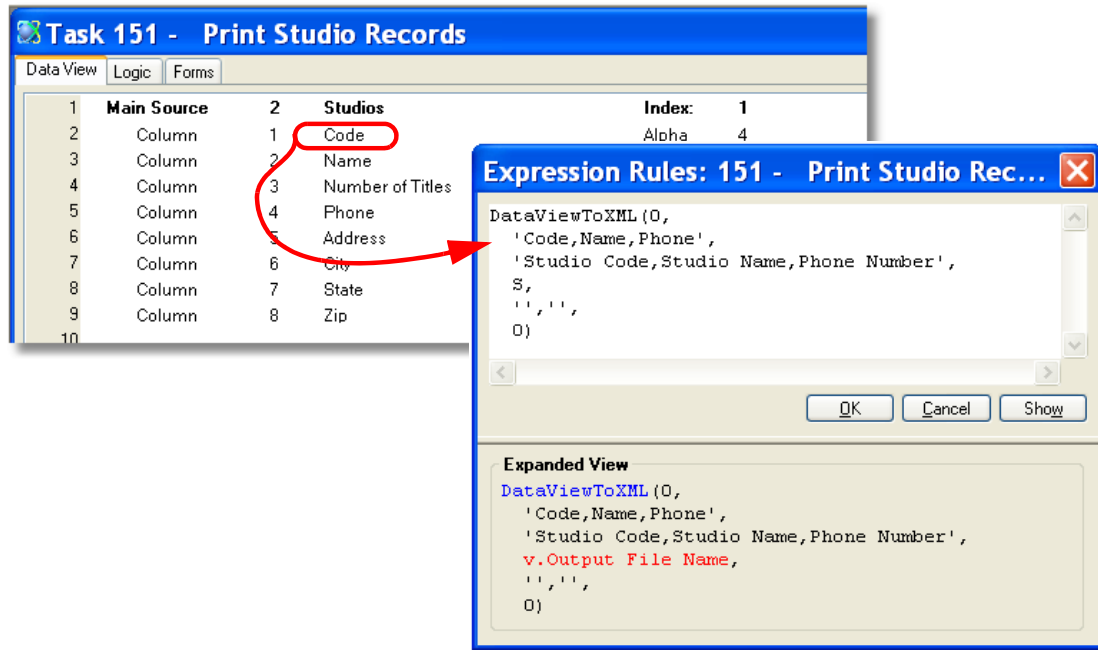
Here is a customized version of our DVD list in HTML.

<i>Studio Code</i>	<i>Studio Name</i>	<i>Phone Number</i>
<b>S001</b>	<a href="#">Twentieth Century Fox Home Video</a>	408 496-7223
<b>S002</b>	<a href="#">Buena Vista Home Video</a>	408 286-2428
<b>S003</b>	<a href="#">Universal Studios</a>	415 935-4228
<b>S004</b>	<a href="#">Paramount</a>	615 297-2723
<b>S005</b>	<a href="#">Warner Home Video</a>	415 843-2991
<b>S006</b>	<a href="#">New Line Home Entertainment</a>	415 836-7128

## How do I Export Data Into an XML File?

You can use the **DataViewToXML()** function to export data from your program into a text file. This function gives you, as the programmer, the most control over the export, so you can, if you choose, limit which records can be exported.

### Using DataViewToXML()



**DataViewToXML()** will export the current data view, either in the current task or its ancestors. The syntax is:

**DataViewToXML**(*Generation*, *VariableList*, *HeaderList*, *SchemaFileName*, *TemplateFileName*, *CharSet*)

Where:

- **Generation** is the generation number. Zero for the current task, 1 for the task's parent, etc.
- **VariableList** is a list of the variable names you want to export. This is the text name of the variable, as it appears in the Data View section of the task. The names are separated by commas (no spaces between variable names, and case matters).
- **HeaderList** is a list of titles that will be used in the export file. In our example, we used "Studio Code" for the "Code" field, to make the output more readable. If the string is empty, no headers will be sent. If the string is '@', the variable list will be used as the header list.
- **SchemaFileName** (optional) is the name of the schema file that will be created. If it is blank, no schema will be created.

- **TemplateFileName** (optional) is the name of the template file that will be used when creating the XML file.
- **CharSet** is a number that sets the character set to be used: 0=ANSI, 1=Unicode, 2=UTF-8.

The result from our sample is shown below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Print_data>
<Record><Studio_Code>S001</Studio_Code><Studio_Name>Twentieth Century Fox Home Video</Studio_Name><Phon
<Record><Studio_Code>S002</Studio_Code><Studio_Name>Buena Vista Home Video</Studio_Name><Phone_Number>4
<Record><Studio_Code>S003</Studio_Code><Studio_Name>Universal Studios</Studio_Name><Phone_Number>415 93
<Record><Studio_Code>S004</Studio_Code><Studio_Name>Paramount</Studio_Name><Phone_Number>615 297-2723</
<Record><Studio_Code>S005</Studio_Code><Studio_Name>Warner Home Video</Studio_Name><Phone_Number>415 84
<Record><Studio_Code>S006</Studio_Code><Studio_Name>New Line Home Entertainment</Studio_Name><Phone_Num
</Print_data>
```

### Creating a schema file

If you specify a schema file name in **DataViewToXMLQ**, a schema file will be generated.

### Using an XML Template file

You can, if you want, specify an XML template file to format the XML. The template file needs to be an XSLT file, and it is used to format the XML.

## How do I Create a Report?

eDeveloper makes it easy to export data to spreadsheets and report writers, using the automated data dumping tools (see Chapter 22, “How do I Dynamically Export the Dataview of a Task into an XML, HTML, Text, or CSV file?” on page 563). These are useful when you want to allow the end-users to work with the data dynamically.

However, eDeveloper also has excellent report writing capabilities build right into the Studio. Writing reports in the Studio is useful when you are creating standardized reports, such as a monthly billing summary, or forms, such as invoices or bills of lading.

Here we will go through the basics of creating a simple report. You would create a complex report using the same procedures, but probably with more fields and using some more advanced features, such as:

- Chapter 22, “How do I Set Repeating Captions for a Table in a Report?” on page 607
- Chapter 22, “How do I Produce PDF Documents?” on page 608
- Chapter 22, “How do I Define Page Header and Footer Information?” on page 588
- Chapter 22, “How do I Create Report Break Levels?” on page 600
- Chapter 22, “How do I Include All Data From a Multi-Line Control in My Report?” on page 605

## Creating a report in eDeveloper

### 1. Create a “Launch screen”

**DVDs in Stock**

Title Keyword

Studio

Sort by

Output Options

Run this report to print a list of desired titles.

It can be printed to the Windows printer or to an Excel spreadsheet.

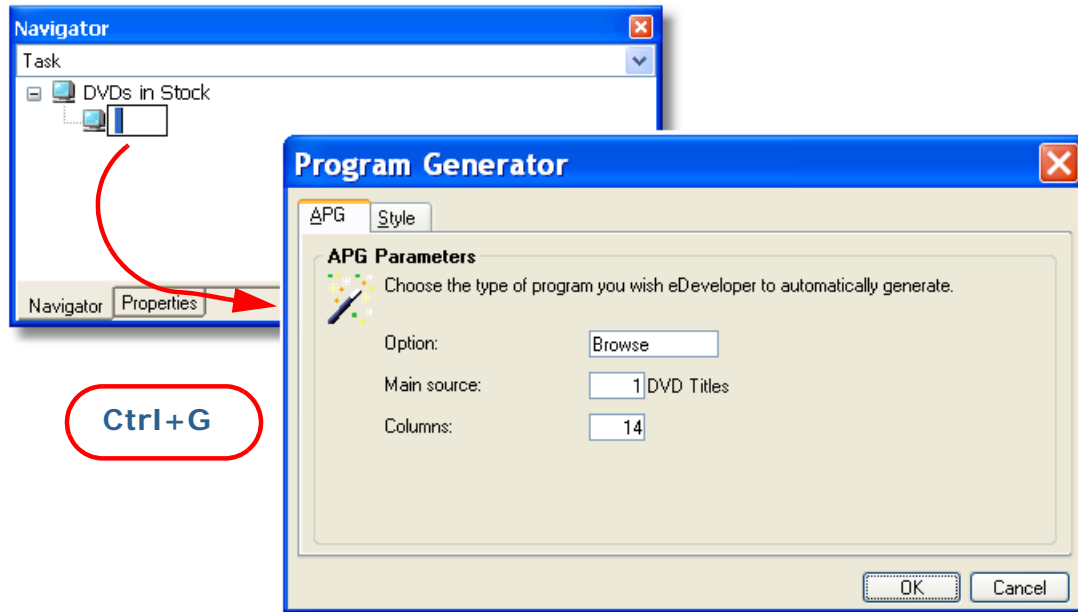
*Leave field blank to select all values*

While not absolutely required, it is a good idea to give the user some kind of screen to start the program, typically with some text that indicates the title of the report and what it is for, and whatever output choices you are giving the user. This is a very simple eDeveloper online task (for details about creating online tasks, see Chapter 5, “How do I Create a Simple Program?” on page 83).

Often the same screen can be used to output the same report in either a GUI report format, or as output to Excel. You can also give the user choices about print preview or sort order.

It is a good idea to come up with a company standard for these kinds of launch tasks. This helps in training users, and it also means you can copy one task to create the next one.

## 2. Create a simple text export program



Use the *Generate program* utility to create a simple browse task. (See Chapter 22, “How do I Create a Quick Browse Program?” on page 567).

Set your Start button to call this task. If the start button works correctly, you should see the browser on the screen.

### 3. Check your sort order and range

**DVDs in Stock**

Title Keyword:  Run this report to print a list of desired titles.

**Browse - DVD Titles**

SN	Title	List Price	Discount	Release date
0784012717	The Boys From Brazil	\$ 69.98	35%	21/09/2004
0790736500	The Postman	\$ 12.98	0%	09/06/1998
B00000K19E	The Matrix	\$ 19.96	25%	21/09/1999
B00003CwLF	Anna and the King	\$ 14.98	20%	04/02/2003
B00003CwT6	The Lord of the Rings - The Fellowship of the Ring (Widescreen Edition)	\$ 19.97	25%	06/08/2002
B00003CX11	Harry Potter and the Sorcerer's Stone (Special Widescreen Edition)	\$ 19.97	25%	01/01/1901
B000052210	The X-Files - Fight the Future	\$ 9.98	30%	23/01/2001

Output

One of the bigger headaches in creating reports is making sure the ranges and sorts work correctly. This is far easier to debug with an online task, which is why we are starting out this way.

If you have multiple sort or output options, you may need to call different report tasks depending on which options the user chose. But, once you get one report finished, you can just copy the subtask to serve as a template for the next version.

### 4. Set up the batch task

**DVDs in Stock**

Title Keyword:  Run this report to print a list of desired titles.

Studio:

Sort by:

Output Options:  It can be printed to the Windows printer or to an Excel spreadsheet.

File Name:

**Print DVD Report**

Report is printing...

Title:

Page:

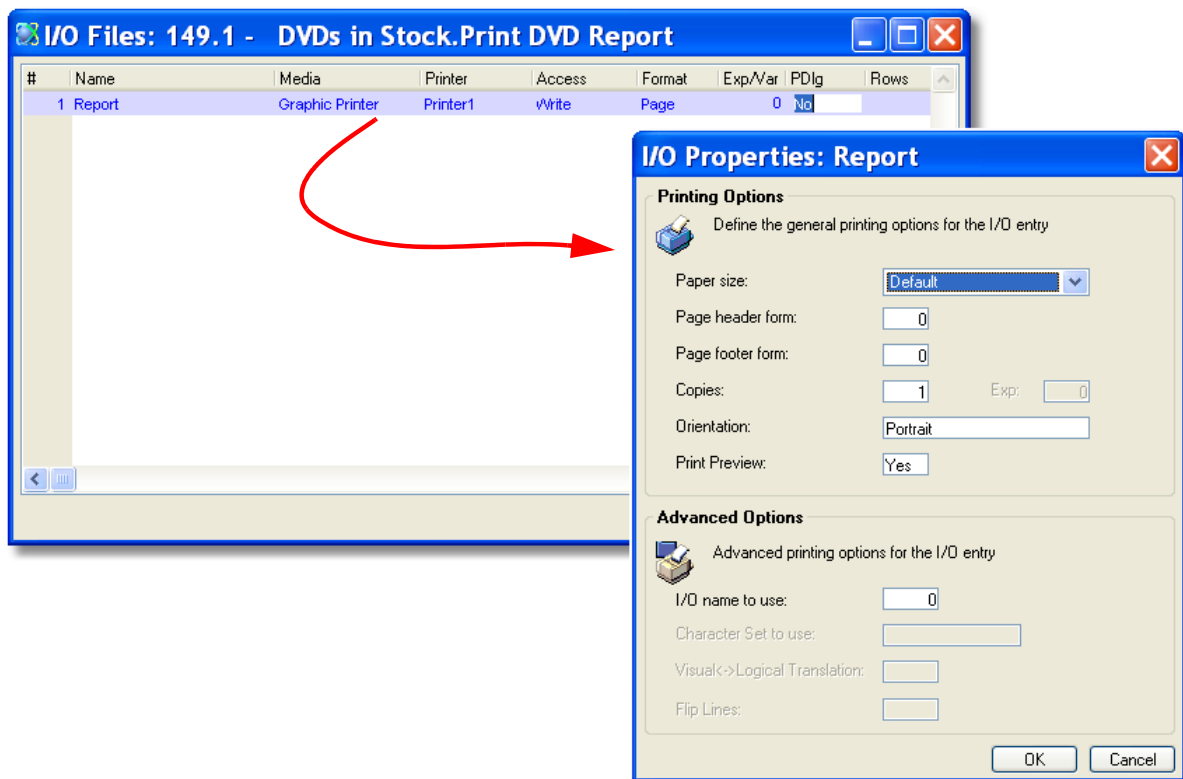
When you are done debugging, change your subtask into a batch task by changing **Task Properties->General->Task type** to **Batch**.



Also, unless you are going to be updating records, make sure that **Task Properties->Data->Transaction Mode** is **Physical** and **Transaction Begin** is **None** (transactions can slow down reports). Similarly, check that the **Access** property of the Main source is **Read**.

Last, change the form that shows to the user so it is very simple and not a table. Delete the generated table, then drop a field or two on the form so the user gets some message to see something is running. You don't want a lot of activity going on in the form though. Repainting the screen is another thing that can slow down the report, and some reports are very big.

## 5. Set up the I/O device



Next, you need to set up an I/O file for the report. The I/O file definition does a lot of the formatting for the report automatically, and the type of the I/O file determines how it prints.

1. Go to your report subtask.
2. Press **Ctrl+I**, or select **Task->I/O Devices**.
3. Press **F4** to open up a line on the list of devices.
4. Set **Media** to **Graphic Printer**.
5. Go to the **I/O Properties** (**Alt+Enter**). Set **Print Preview** to **Yes**. Print preview will help you in your testing.

As you can see, there are plenty of other options you can set here. If you position the cursor on any of them and press **F1**, you can view the Help file for them.

When testing, Print preview is very useful. In production however, you might want to use print dialog (let the user route it directly to the printer). If you want a print dialog, you would enter **Yes** in the **PDlg** column.

Or, you might want to force the report to go to a specific printer, as would usually be the case for printing documents on specific forms, like checks. In that case, you would set **PDlg** to **No**, and also set **Print preview** to **No**.

Most of these options can also be set to expressions too.

## Create your forms

The screenshot shows the 'Task 149.1 - DVDs in Stock.Print DVD' window with the 'Forms' tab selected. The 'Forms' list on the left has the following items:

#	Name	Class	Area	Interface T...
1	Main Program	0		GUI Display
2	DVDs in Stock	0		GUI Display
3	Print DVD Report	0		GUI Display
4	Report Header	1	Header	GUI Output
5	Report Footer	1	Footer	GUI Output
6	Report Detail	1	Detail	GUI Output

The 'Report Detail' form is selected, and the 'Form Properties GUI Output - Report Detail' dialog box is open. The 'Model' tab is selected, and the 'Details' section is expanded. The 'Form units' are set to 'Inches'. The 'Grid X' and 'Grid Y' are both set to '0.030'. The 'Form name' is 'Report Detail'. The 'Input' section shows 'Area' set to 'Detail'. The 'Appearance' section shows 'Font' set to '1' and 'Color' set to '1'. The 'Navigation' section shows 'Width' set to '8.000' and 'Height' set to '0.190'.

Next you want to create your forms. The basic steps are:

1. Go to the **Form** tab.
2. Press **F4** to open up a line.
3. Give your form a **Name**, and a **Class** that is not zero.
4. Set the **Area** depending on the function of the form. For this example, make one **Page Header**, one **Page Footer**, and one **Detail** form.

5. Set the **Interface Type** to **GUI Output**.

In the Form Properties, it is important that the **Form units** are set to **Inches** or **Centimeters**. If you use Dialog Units, the form will resize itself, which you don't want. Make sure the **Width** is the size of your paper. The **Height** will vary for each form, and is easily resized with the mouse.

It is best to use a Model for the report form. That way the width, font, and form units are all set up automatically and consistently.

**Hint:** If the forms are way too big to edit easily, set the size manually before you zoom into it. Just type in a reasonable size for the width and height in form properties.

### Editing your forms

**Report form: class 1**

Report Header

Date: ####/####/####      DVDs in Stock      Page: 4

Time: HH:MM PM

Report Detail

Studio	Title	Starring	List Price
4	100	100	####.##

Report Footer

Total Titles: 4      Total Price: 6.22 +\$;

When you zoom on any of the class 1 forms, you will see all three of the forms together. The header will appear at the top, and the footer at the bottom, even though they are not listed that way in the form list. Sometimes it is convenient to edit just one or two forms at the same time. In that case, just set the class number temporarily to something else, and you can edit those forms separately.

At this point, you edit the form as you would an online form, selecting fields and dropping them onto the form. It is very helpful to have standard models for the various types of fields.

The Detail form is basically a table. It is shown here in the default table format to make this clear, though likely for most of your reports you will turn off the dividing lines. It is created like any other table: drop a table control onto the form, then drop the fields onto it.

When a table control is included on a report, the options are slightly different than when a table is on an online form. There is a property, **Title on every page**, which controls whether or not the title row automatically is printed. Usually, you want this to be Yes.

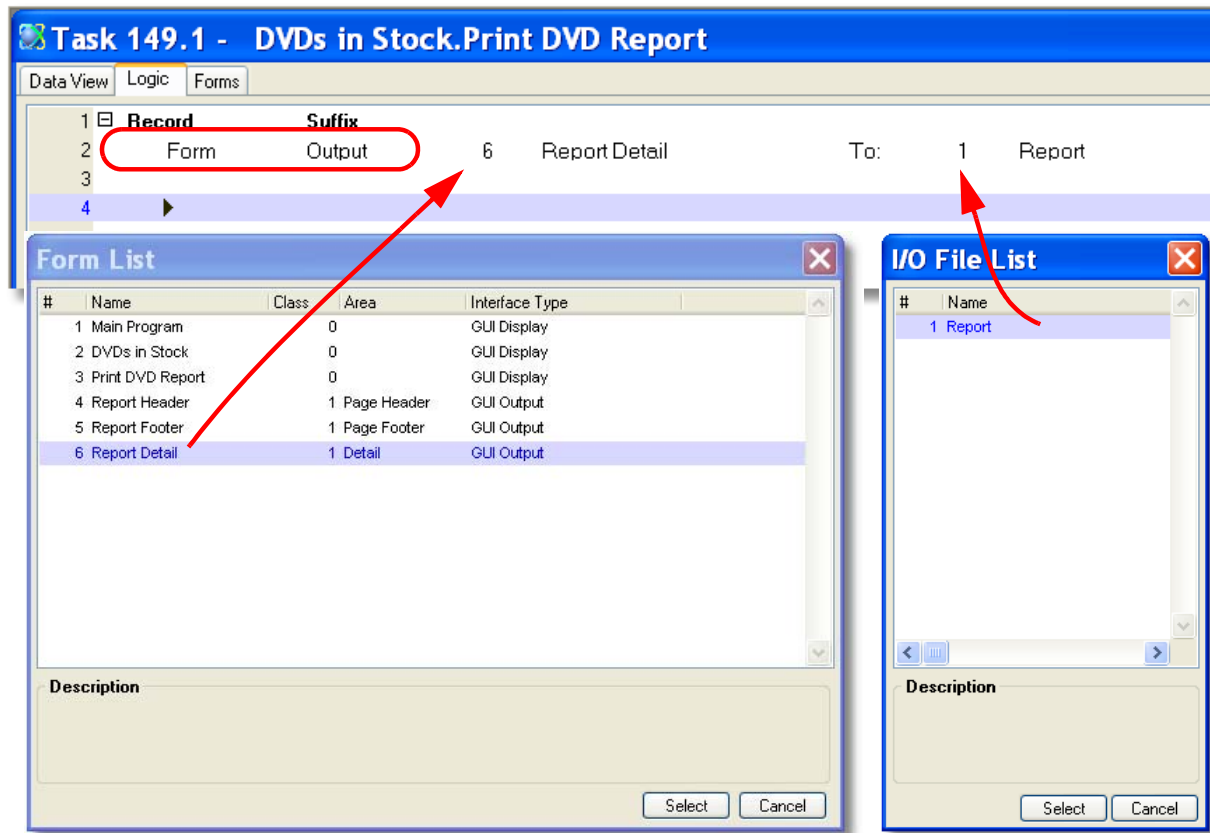
Another property is **Fix size table**. If **Fix size table = Yes**, then, if there are more records than will fit, the table repeats.

If **Fix size table = No**, then the table will resize to fit the data. It will repeat on the next page, if necessary.

The amount of white space around the table on the form is retained, just as it is on online forms.

Control Properties : Table	
Categorized   Alphabetic	
Model [default]	
Appearance	
Column divider	Yes
Last Divider	No
Color	1 0
Visible	0
Style	2-D
Border style	No Border
Line divider	No
Title height	0.250
Row height	0.260
Columns	4
Fix size table	No
Title on every page	Yes
Navigation	
Left	0.150
Top	0.170
Width	7.700
Height	1.580
Column divider Defines whether the table displays a divider between its columns.	
Properties   Checker result	

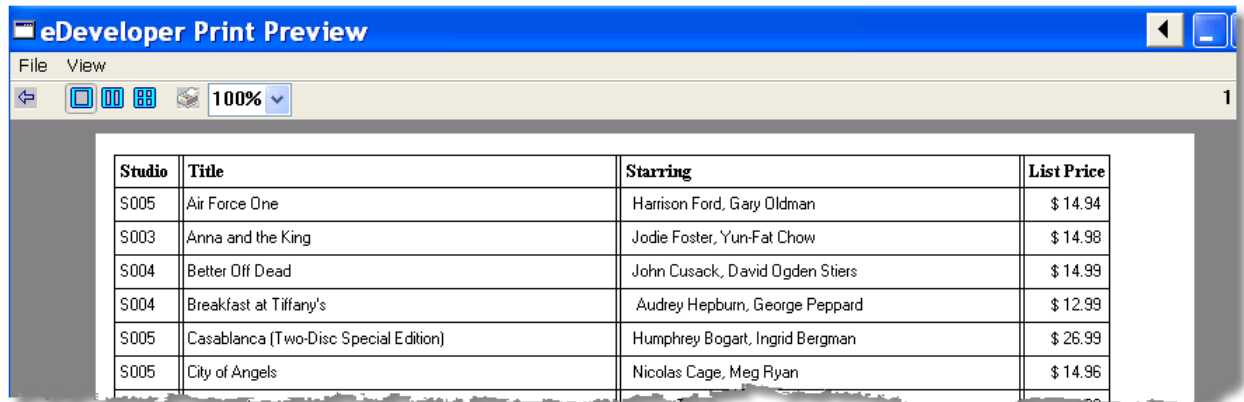
## Making the detail form print



Now, you just need to make the report actually print. Usually the detail line is printed once per record, so we print it in a Record Suffix event, as shown in this example. However, if it is a summary report, it would be in a Variable change event. You can print any of the forms in any event you want.

1. Go to the event where you want to print the form (Record suffix, in our example).
2. Press **F4** to open up a line.
3. Select the **Form** operation by typing F. The next field will default to **Output**, which is what you want.
4. Zoom to select the form you want to print.
5. The **To:** field will default to I/O device 1, which is usually what you want. If there is more than one I/O device though, you can zoom to select a different one here.

Now when you print the report you will see in the print previewer:

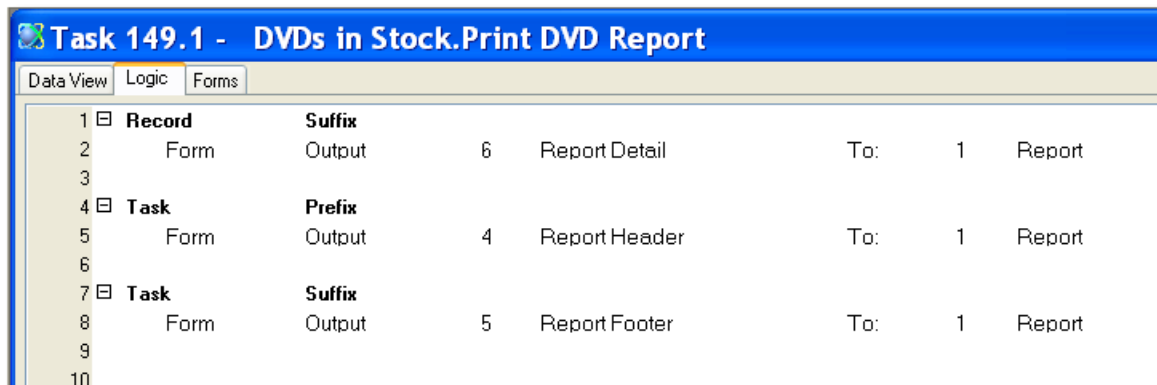


The image shows a screenshot of the 'eDeveloper Print Preview' window. The window has a blue title bar and a menu bar with 'File' and 'View'. Below the menu bar is a toolbar with icons for navigation and zooming, and a '100%' zoom level dropdown. The main content area displays a table with the following data:

Studio	Title	Starring	List Price
S005	Air Force One	Harrison Ford, Gary Oldman	\$ 14.94
S003	Anna and the King	Jodie Foster, Yun-Fat Chow	\$ 14.98
S004	Better Off Dead	John Cusack, David Ogden Stiers	\$ 14.99
S004	Breakfast at Tiffany's	Audrey Hepburn, George Peppard	\$ 12.99
S005	Casablanca (Two-Disc Special Edition)	Humphrey Bogart, Ingrid Bergman	\$ 26.99
S005	City of Angels	Nicolas Cage, Meg Ryan	\$ 14.96

Now all we need to do is add the header and footer.

## Printing the header



The image shows a screenshot of the 'Task 149.1 - DVDs in Stock.Print DVD Report' window. The window has a blue title bar and a menu bar with 'Data View', 'Logic', and 'Forms'. Below the menu bar is a table with the following data:

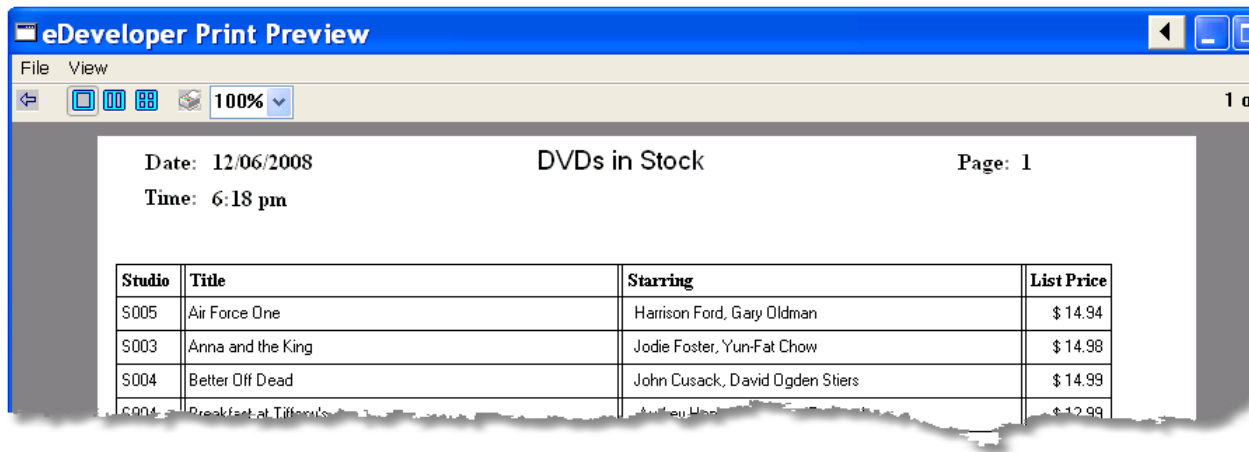
Record	Suffix	Form	Output	Report Detail	To:	1	Report
1	Record	Suffix	6	Report Detail	To:	1	Report
2	Form	Output	6	Report Detail	To:	1	Report
3							
4	Task	Prefix	4	Report Header	To:	1	Report
5	Form	Output	4	Report Header	To:	1	Report
6							
7	Task	Suffix	5	Report Footer	To:	1	Report
8	Form	Output	5	Report Footer	To:	1	Report
9							
10							

To print the report header, repeat the procedure for printing the detail line. However, the header is usually printed for the first time in Task prefix. From then on, it will be automatically reprinted whenever a form is first printed on a new page. There can be more than one form defined as a “header,” and all of them will print when a new page is started.

Alternatively, you can define a Form as a Page Header, in which case it will automatically print at the top of any page, and it does not need to be manually output in Task prefix. (See Chapter 22, “How do I Define Page Header and Footer Information?” on page 588).

## Printing the footer

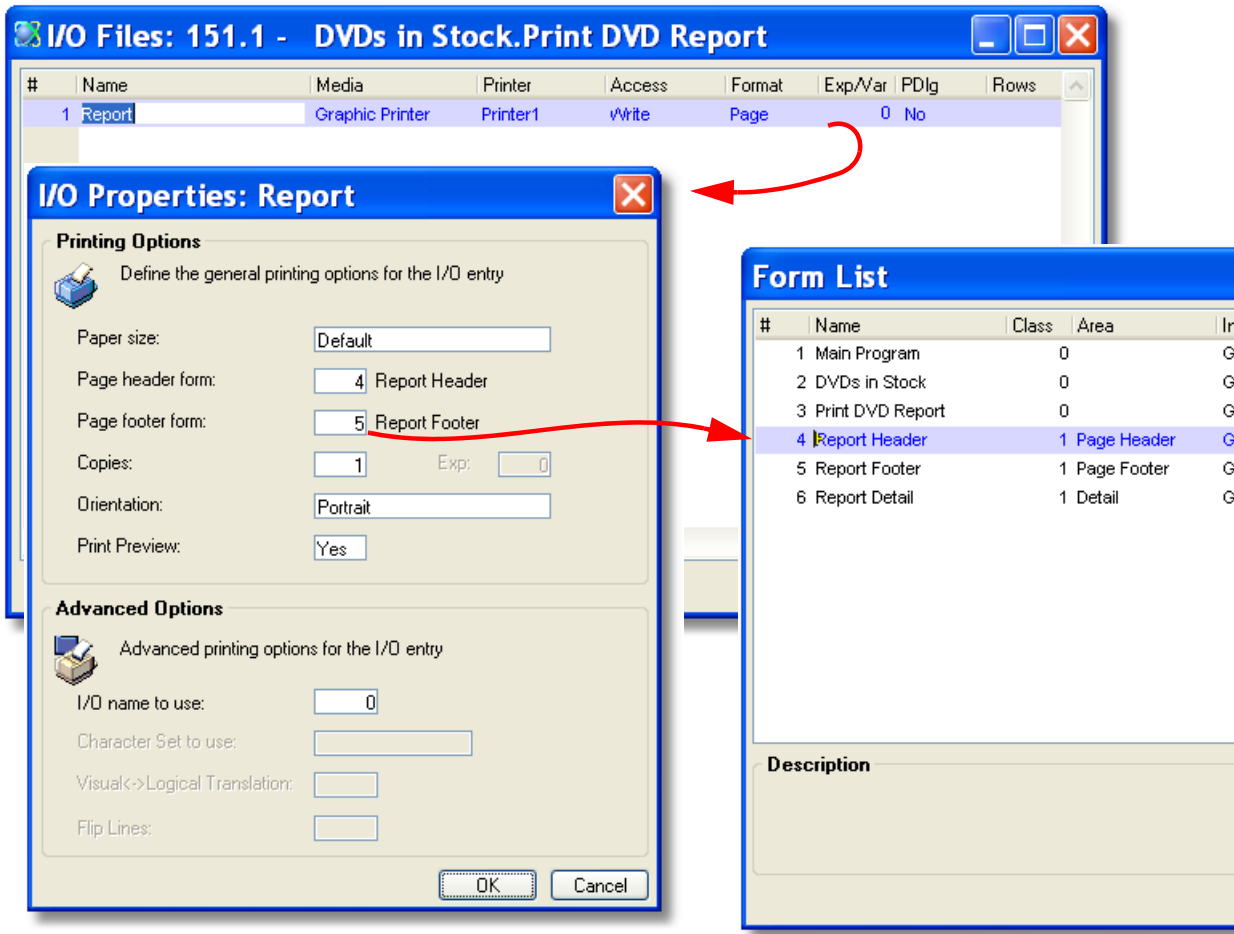
The report footer is printed in Task suffix, after the rest of the report has printed. It only prints once, for this report -- after the last record has printed -- so putting it in Task Suffix makes sense. Footers often print at break levels too, to print subtotals. These are explained in Chapter 22, “How do I Define Aggregates per Break Level?” on page 603.



Now the report prints with headers and footers.

- See also:** Chapter 22, "How do I Create Report Break Levels?" on page 600  
Chapter 22, "How do I Define Aggregates per Break Level?" on page 603  
Chapter 22, "How do I Include All Data From a Multi-Line Control in My Report?" on page 605  
Chapter 22, "How do I Set Repeating Captions for a Table in a Report?" on page 607

## How do I Define Page Header and Footer Information?



Page headers and footers are specified in the **I/O Properties** of the I/O file. The positioning is handled automatically: the Page Header will be at the top of the page, and the Page Footer at the very bottom.

Note that a *page* header or footer is not necessarily the same thing as a *report* header and footer. Page headers and footers print on every single page, and typically look identical on every page. A typical *page* footer, for instance, would print in the same spot, at the bottom of each page, and might be just a line and a page number. But a *report* footer would have the totals for the report, and would print just after the last line of the report.



The forms used for the Page Header and Page Footer do not need to be located in the current task. For Page Header especially, it can be convenient to keep the form in the Main Program and use it in all your reports. See Chapter 22, “How do I Define a Global Page Header or Footer?” on page 590 for how to do this.

Date: 12/06/2008

DVDs in Stock  
That contain "Harry"

Time: 6:54 pm

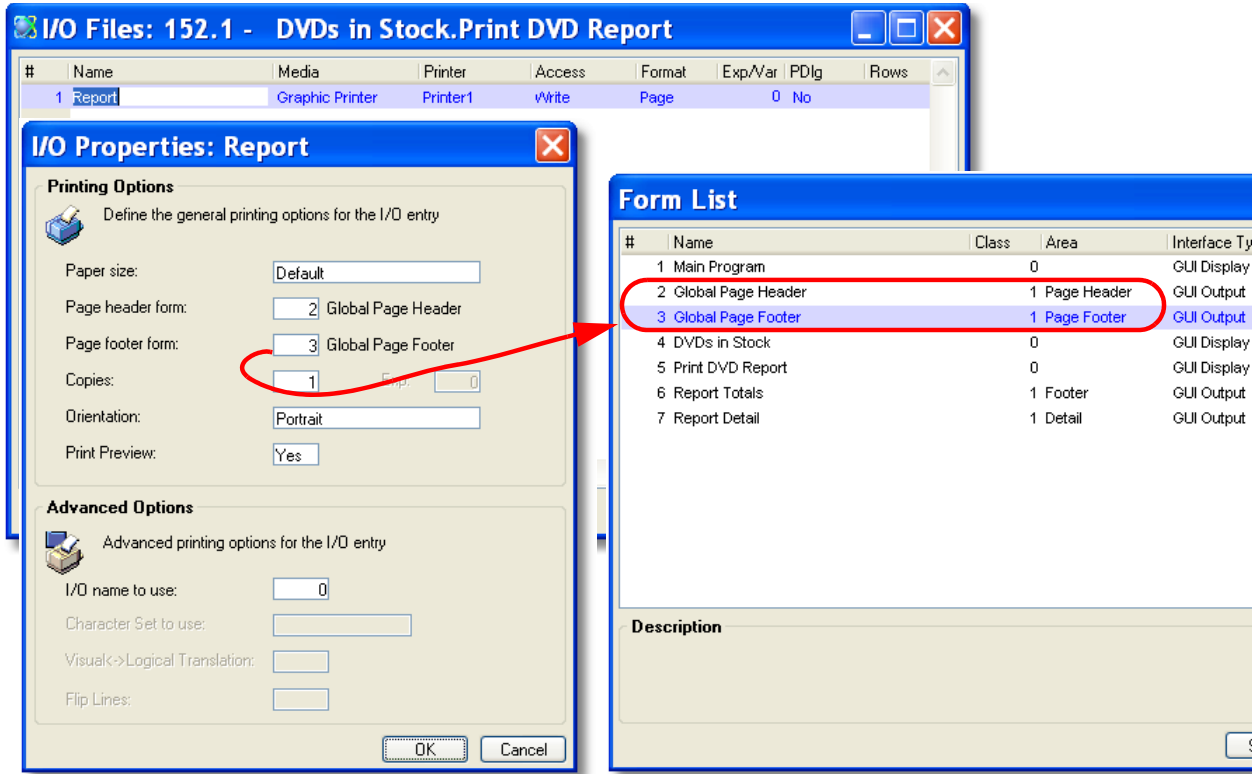
Studio	Title	Starring	List Price
S005	Harry Potter and the Chamber of Secrets (Widescreen Edition)	Daniel Radcliffe, Rupert Grint	\$ 19.97
S005	Harry Potter and the Prisoner of Azkaban (2-Disc Widescreen E	Daniel Radcliffe, Rupert Grint	\$ 19.97
S005	Harry Potter and the Sorcerer's Stone (Special Widescreen Edi	Daniel Radcliffe	\$ 19.97

- Page 1 -

**Hint:** You can use the *visibility*

*property to change the look of a header or footer form based on the type of page that is printing or other criteria. Just make one set of fields visible under one condition, and another set of fields visible under different conditions. You can also store the text in variables that change based on conditions.*

## How do I Define a Global Page Header or Footer?



It is often useful to have standard page headers that print on reports. Besides making the reports more consistent, it saves a lot of work when creating new reports. It also avoids the problem of having a hard-coded “Company name” that has to be changed when the company name changes.

Since the page header and footer are specified in the I/O device properties, it is a simple matter to point the task to the global form rather than a local one.

### Creating and using a global form

You create a global form simply by creating the form in the Main Program. The process is exactly the same as creating the form in any task.

Once the form is created, it will show on the form list, above the forms for that task. In the example above, you can see our two global forms, which are forms #2 and form #3. Our local forms for this task are forms #6 and #7.

## Handling variable data on the global page header or footer

**Task 153.1 - DVDs in Stock.Print DVD Report**

Data View Logic Forms

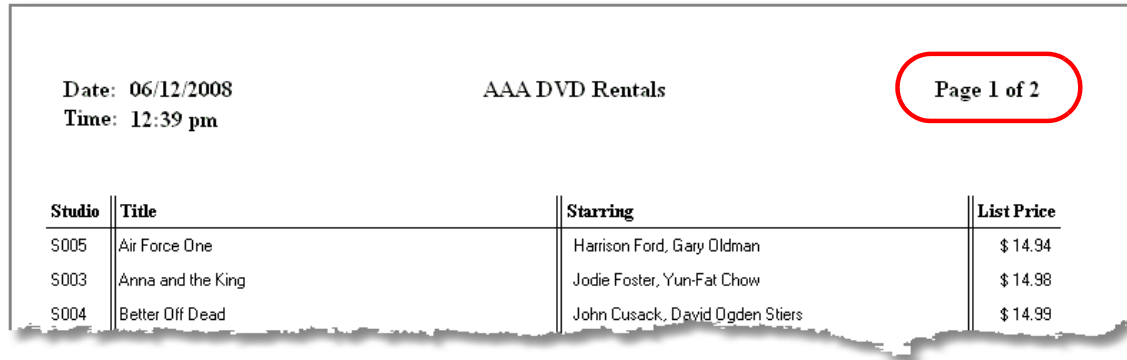
	Record	Suffix			To:		
1	Form	Output	7	Report Detail	To:	1	DVDS
2	Update	Variable	BF	Total Price	With:	9	Total Price+List Price
3							
4							
5	Task	Prefix					
6	Update	Variable	F	q.Report Name	With:	2	'DVDs in Stock'
7	Update	Variable	G	q.Report subheader	With:	3	IF(f.Title Keyword=","','Contai
8							
9	Event	Page Header					Scope: SubTree
10	Update	Variable	C	q.Report Page#	With:	5	Page(0,1)
11							
12							

There are some considerations when creating global headers and footers.

First, the report name typically prints in the header, along with perhaps some words about what filters were used. You can handle this easily by adding global variables to the Main Program, and updating them in Task Prefix of your reporting task.

Second, the Main Program doesn't have access to the page number via the **Page(n,n)** function which is typically used for reports, so you need to update a global page number variable from your reporting task. This is done by setting a Page Header event. When the Page Header event is triggered, it updates the global page number. In our example, it is updated to **Page(0,1)** because the I/O device is the first I/O device in the current task.

## How do I Enumerate Report Pages?



Date: 06/12/2008 Time: 12:39 pm		AAA DVD Rentals	Page 1 of 2
Studio	Title	Starring	List Price
S005	Air Force One	Harrison Ford, Gary Oldman	\$ 14.94
S003	Anna and the King	Jodie Foster, Yun-Fat Chow	\$ 14.98
S004	Better Off Dead	John Cusack, David Ogden Stiers	\$ 14.99

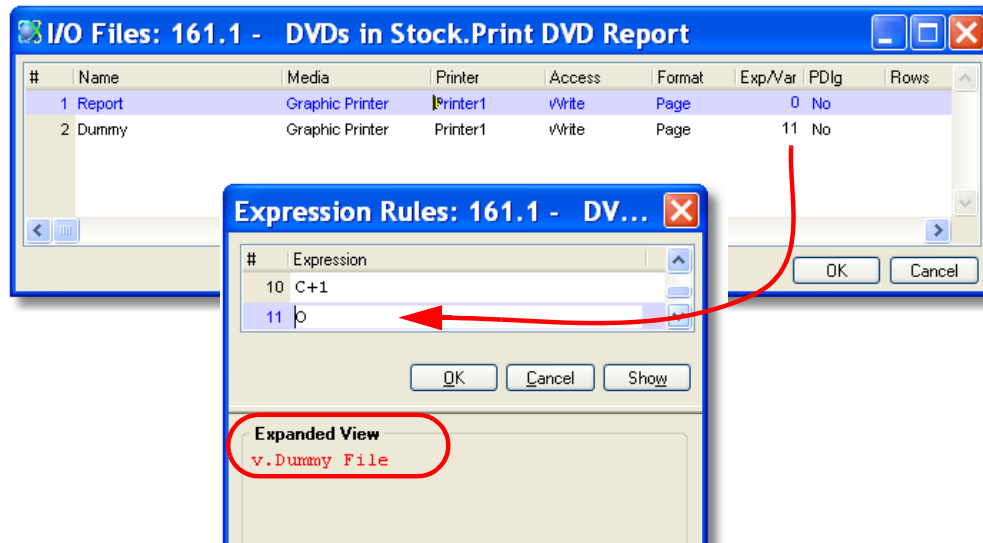
Users often like to have reports with the numbering system “Page n of nn”.

However, it is not possible to know in advance how many pages will be printed in a report, for a given set of data on a particular printer. Page sizes can vary, as can margins and even fonts. For this reason, you need to cycle the report twice: first, to get the total number of pages, and second, to actually do the printing.

Fortunately, this is easy to do in eDeveloper. Here we will show you how.

In our example we are storing the page numbers in our [Main Program](#), which also prints all the headers and footers. So the variable `g.Report Page#` holds the current page number, and `g.Report Total Pages` holds the total number of pages that will print. For more information about using global page headers and footers, see Chapter 22, “How do I Define a Global Page Header or Footer?” on page 590.

## 1. Set up two I/O Devices



In your report subtask, set up two I/O devices. The first one is your actual report. The second one is set up to go to a file, such as `%TEMP%Dummy.txt`. We used a variable in the parent task to hold the file name, so we can easily use **FileDelete()** to delete it after we are done.

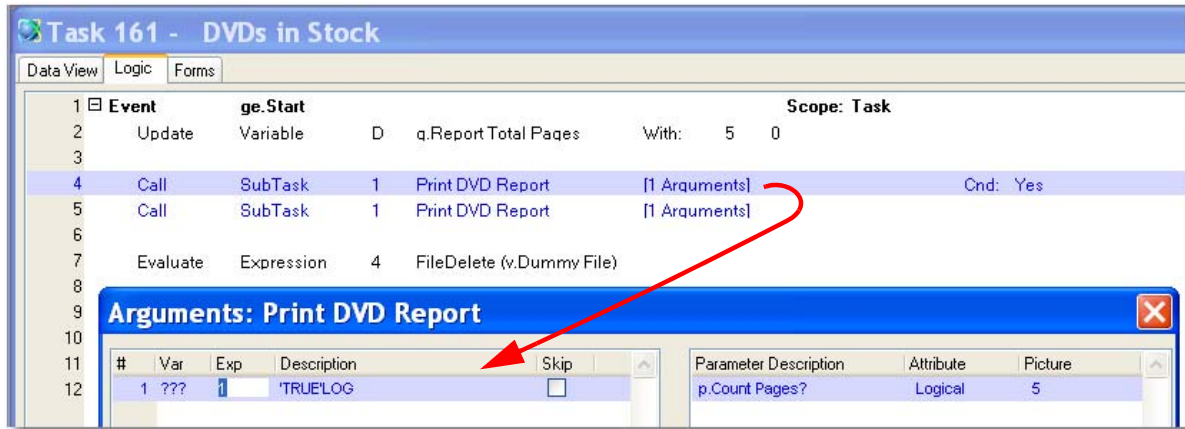
The presence of this filename keeps the output from actually printing. The output is formatted *as if* it were going to print, but the output is actually stored in a file.

It is important that each of these devices is set up the same. For instance, you want to use the same headers and footers, or the page count will not be the same on the two devices.

Also in our report subtask, we create a parameter, “`p.Count Pages?`”. When this is TRUE, the task will only print to Dummy, and no real output will be produced.

**Note:** Usually you will want to use the same printer for both I/O devices. However, there are a multitude of printer setups in Windows, and there are cases where the printer driver is set up to do something like prompt the user for a filename, which you don’t want to happen twice in the same job. In that case, just set up a “DummyPrinter” in **Options->Settings ->Printers**.

## 2. Set up the calling task



The calling task is going to do four things:

1. Zero out the global variable `g.Report Total Pages`.
2. Call the printing subtask with `p.Count Pages? = TRUE`. This will count the total number of pages.
3. Call the printing subtask with `p.Count Pages? = FALSE`. This will actually print the report.
4. Delete the temporary file.

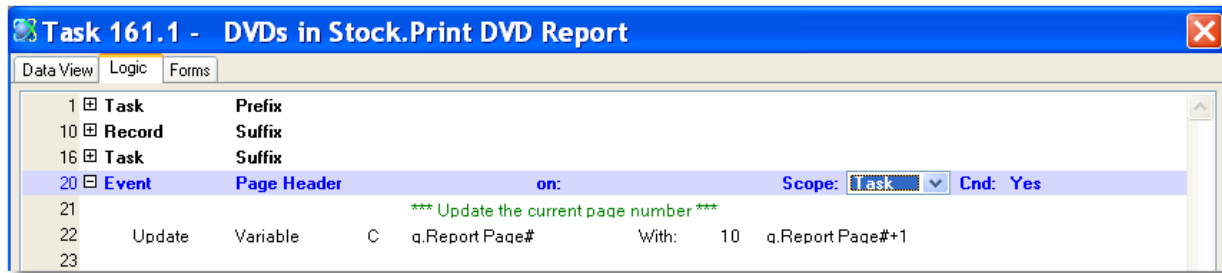
## 3. In Task Prefix, zero out report variables



Now, in your subtask, zero out `g.Report Page#`, and any other variables that are totalled for the entire report.

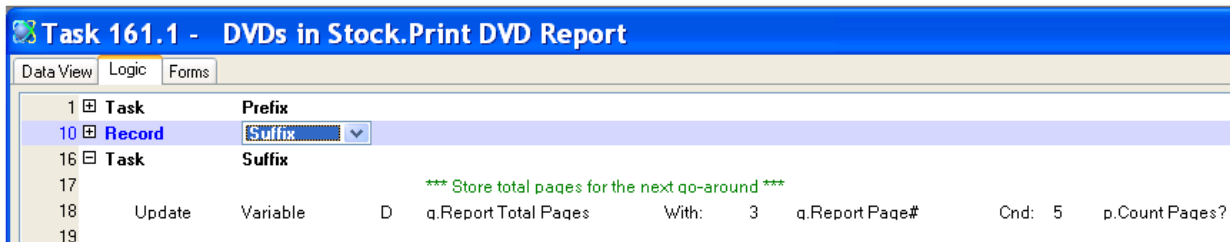
**Note:** It is not required that variables, like `Total Price`, to be zeroed out, because eDeveloper automatically initializes them to zero when the task starts. However, it is a good practice to zero them out anyway, because often tasks end up getting set to “resident” for efficiency. In that case, then the totals will suddenly be off if the user runs the report twice.

#### 4. Create a Page Header event to count the pages



Create a Page Header event to update the global page number. This will keep track of the current page number for either I/O device.

#### 5. Store the total number of pages on the first go-around



Now, in Task Suffix, update your global total number of pages, `g.Report Total pages`, with the current page number. Note we have a condition on this so it only happens on the first iteration. That isn't strictly necessary, since the results should be the same for each iteration, but it makes the code a little more clear.

#### 6. Output to two I/O devices



Now here is the part that makes this all work. Replicate each Form Output operation (in most reports there won't be very many: only one in this example). The first Form Output will go to the Dummy I/O device, if `p.Count Pages` is TRUE. The second Form Output will go to the Report I/O device, if `p.Count Pages` is FALSE.

Now, when the report runs, it will cycle twice, and you will get correct “Page n of nn” headers.



## How do I Print a Report from Several Programs to the Same I/O Device?

Date: 12/06/2008

AAA DVD Rentals

Page: 2

Time: 7:17 pm

Studio	Title	Starring	List Price
S005	The Postman	Kevin Costner, Will Patton	\$ 12.98
S001	The X-Files - Fight the Future	David Duchovny, Gillian Anderson, John Neville, Will	\$ 9.98
S005	Three Kings	George Clooney, Mark Wahlberg	\$ 12.98

### Studio Cross-Reference

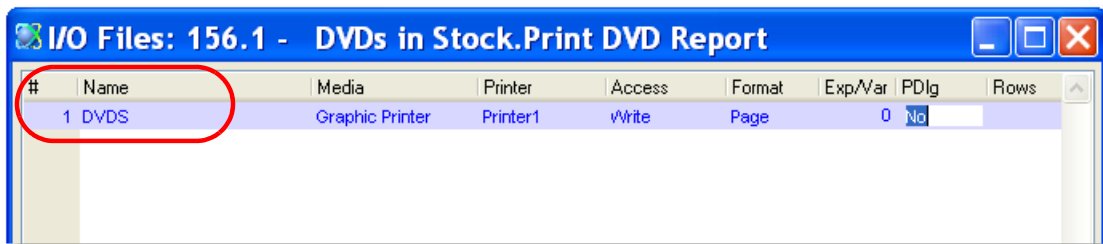
Code	Name
S001	Twentieth Century Fox Home Video
S002	Buena Vista Home Video
S003	Universal Studios
S004	Paramount
S005	Warner Home Video
S006	New Line Home Entertainment

The scope of an I/O Device is much the same as the scope of variables or forms. That is, each task can see and use the I/O Devices of its ancestors.

However, it is often useful to call a totally unrelated program to do some printing. For instance, you might want to have a generic program to print some summary information or format some text. You can do this in eDeveloper by using a facility called **I/O Name to Use**.

In this example, we want to create a “Studio Cross-Reference” that will print at the end of several reports, so new users can figure out the studio codes. We have a program that prints out the codes, but how do we attach it to several different reports? Below we will show you how.

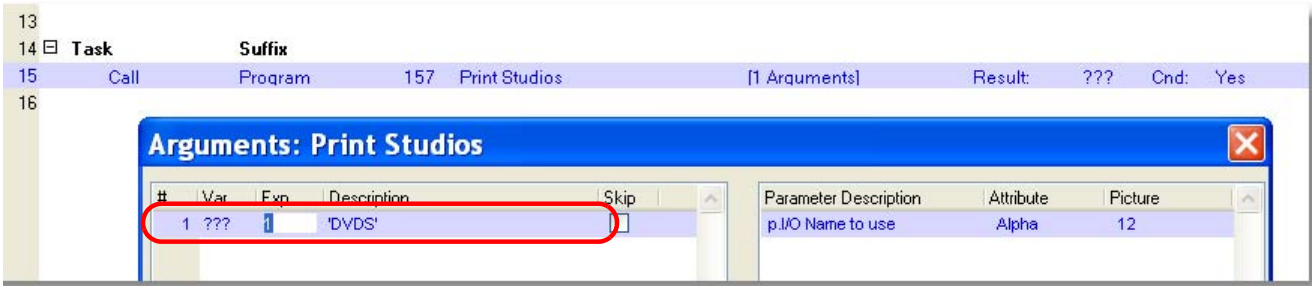
Set up the calling program



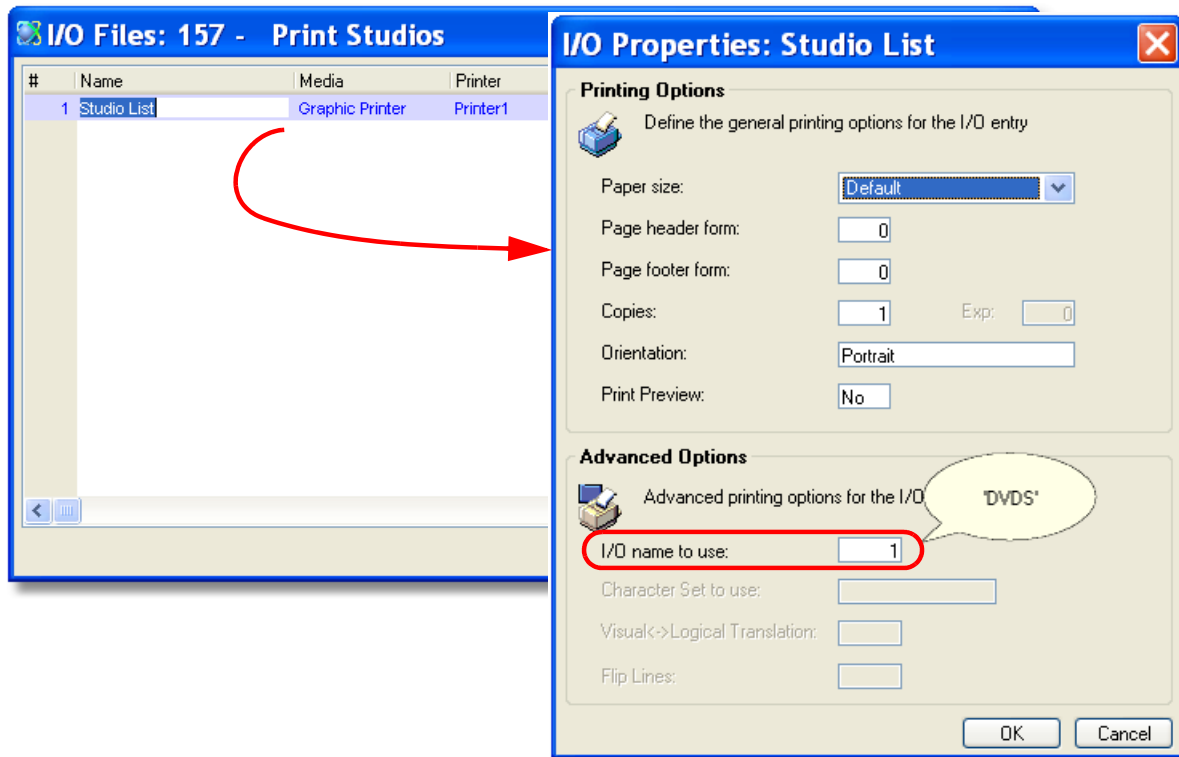
First, you need to set up the program that opens the I/O Device. There isn't very much to set up here, just two things:

- 1. Set the I/O Device name. It's good to make it something simple, with no spaces.
- 2. Pass the I/O Device name to the called program. You don't *have* to do this, but if you hard-code the name it wouldn't make the called program very generic.

In our example, we typed 'DVDS' for I/O device name. Then we created an expression with the string 'DVDS' and passed that to the called program.



## Setting up the called program



Now, in our called program, we just have to set up the I/O Device. In the **I/O Properties**, set the **I/O Name to use** to the passed parameter, which in this case will evaluate to 'DVDS'.

That's all there is to it. Now the output of the called program will be routed to the I/O device opened in the first program.

**Hint:** In some cases, you might want to have a "controlling task" which calls two or more separate programs for output. In this case, open the I/O Device in the controlling task. If that task is the online task, or if there might not be any output, set **Settings->Environment->Preferences->IO device Open timing** to *On Demand* to prevent blank pages from being printed.

## How do I Create Report Break Levels?

Date: 06/12/2008

AAA DVD Rentals

Page: 1

Time: 2:41 pm

Studio: S001 Twentieth Century Fox Home Video

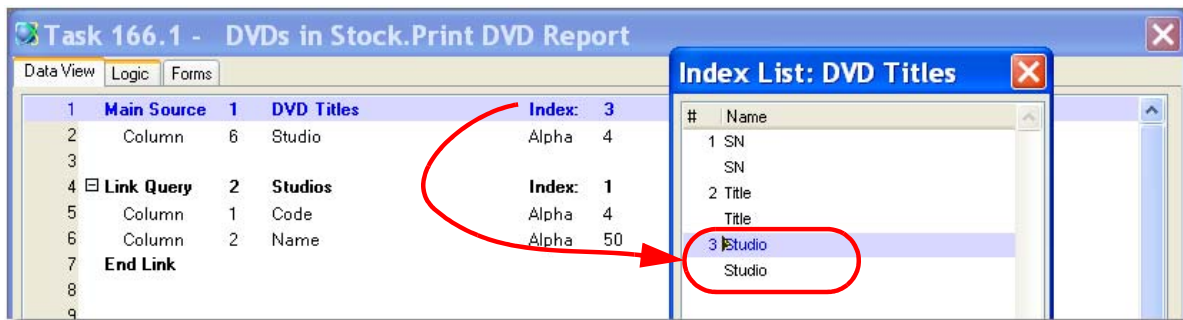
Title	Starring	List Price
The Boys From Brazil	Gregory Peck, Laurence Olivier, James Mason, Lilli P	\$ 69.98
Moulin Rouge (Single Disc Edition)	Nicole Kidman, Ewan McGregor, John Leguizamo, Jir	\$ 19.98
Mystic Pizza	Annabeth Gish, Julia Roberts, Lili Taylor, Vincent D'O	\$ 14.99
The X-Files - Fight the Future	David Duchovny, Gillian Anderson, John Neville, Will	\$ 9.98
Star Wars Trilogy (Widescreen Edition)	Harrison Ford	\$ 69.98
Total titles for S001: 5		Total Price: \$184.91

Reports often have *break levels*, where data is grouped together and subtotals are printed. This can be one of the most challenging things to do in a lower-level language, but eDeveloper has built-in functionality that makes it easy to create even very complex break level reports. In this section we will show you how. Here is a summary of the steps:

1. Make sure your record order matches the break level
2. Set up a variable that will change at the right time
3. Set up your Group levels
4. Sum the totals

Now let's go through them step by step.

### 1. Make sure your record order matches the break level



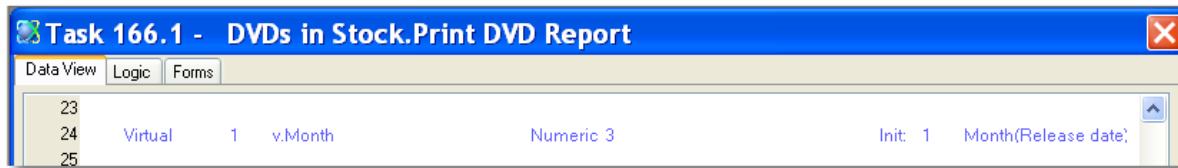
One of the most common errors on break level reports is using the wrong record order. In our example, we are going to group records by “Studio”. So, we also have to be sure we use the “Studio” index.

If there is no index that will work for the report you want, you can use **Task->Sort** to create a customized record order for this task. See Chapter , “Creating an index on the fly” on page 477.

## 2. Set up a variable that will change at the right time

Usually, you will have a variable in the DB Source that you can use for a break level. In our example, since the records are ordered by “Studio”, we will use the Studio column to break on.

In some cases, however, there will not be a variable handy. For instance, if you want to create a report that summarizes data by month, but only have a date field in the data. The records would be ordered correctly, by date, but there is no field that changes just when the month changes.



In that event, you would set up a variable called, for instance, v.Month, and initialize it to the month part of the date, using the **Month()** function, as shown here. Or, you could concatenate several fields together to cause a break only when any one of those fields change. Virtuals can be used for break levels just like Columns can.

## 3. Set up your Group levels



Next, you want to set up your Group levels. The Group levels will point to the variable you are using for the break, which in this example is the Studio column.

In **Group Prefix**, you will print out the header (if you are using one) and zero out the totals that are used in this break.

In **Group Suffix**, you will print out the footer (if you are using one).

Alternatively, you can zero out the break level in Group Suffix, after you print the form. Either method works, but it's good to be consistent.

#### 4. Sum the totals



Last, you need to update your totals whenever you print a record. Usually this is done in Record Suffix, with simple Update operations.

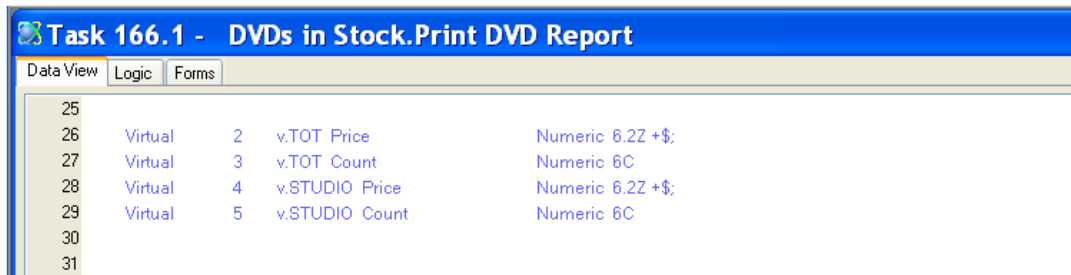
## How do I Define Aggregates per Break Level?

To define your break level aggregates, there are 3 steps:

1. Define the aggregate variables
2. Update the aggregates
3. Zero out the aggregates

Let's take each of these step by step.

### 1. Define the aggregate variables



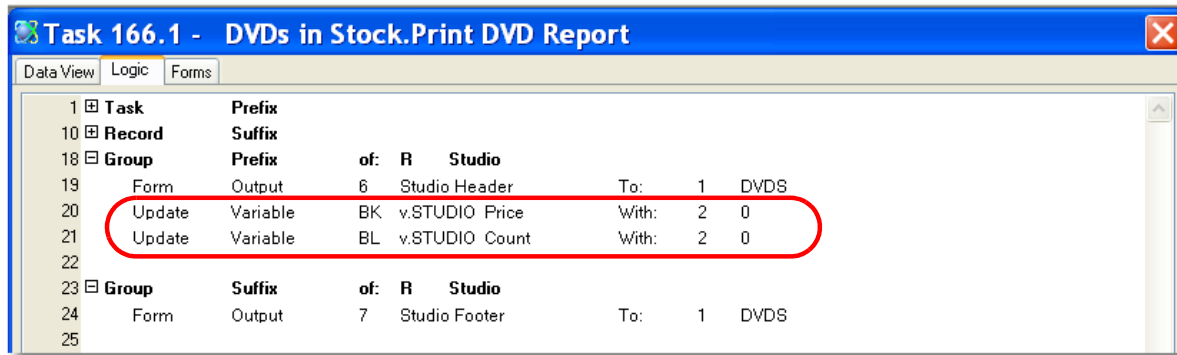
The variables used for aggregates are usually just virtuals, and they are defined in the **Data View** as you would any virtual. However, with aggregates it is a good idea to have some kind of standard naming convention to make them easy to spot, because it is easy to use the incorrect variable. In this example, we added a capitalized prefix to specify which variable goes with which break level.

### 2. Update the aggregates



The easiest way to update the aggregates is to increment them all as each record is read. That way all your updating logic is in one place, which makes debugging faster.

### 3. Zero out the aggregates



Last, you need to zero out the aggregates. This can be done in the Group Prefix break level, or in Group Suffix after the aggregates have printed.



## How do I Include All Data From a Multi-Line Control in My Report?

Date: 12/06/2008

DVDs in Stock

Page: 1

Time: 4:39 pm

Studio	Title	Starring	List Price	Release date
S001	The Boys From Brazil	Gregory Peck, Laurence Olivier, James Mason, Lilli Palmer, Uta Hagen	\$ 69.98	09/21/2004
S001	Moulin Rouge (Single Disc Edition)	Nicole Kidman, Ewan McGregor, John Leguizamo, Jim Broadbent, Richard Roxburgh	\$ 19.98	01/14/2003
S001	Mystic Pizza	Annabeth Gish, Julia Roberts, Lili Taylor, Vincent D'Onofrio, William R. Moses	\$ 14.99	10/21/1988
S001	The X-Files - Fight the Future	David Duchovny, Gillian Anderson, John Neville, William B. Davis, Martin Landau	\$ 9.98	01/23/2001
S001	Star Wars Trilogy (Widescreen Edition)	Harrison Ford	\$ 69.98	09/21/2004

Sometimes the data on a report will be rather variable. For instance, in our sample report the Title and Starring fields can be very short, or very long. In this sort of instance, it works well to have the data wrap inside one table column.

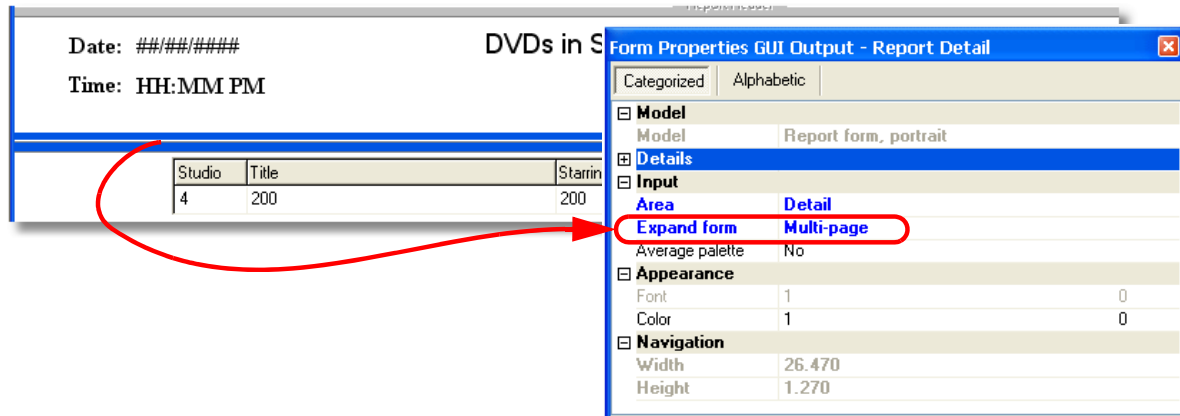
In eDeveloper, you do this by changing the properties on the field and form to allow them to expand with the data.

### 1. Change the field Property

The screenshot shows a report preview on the left and the 'Control Properties : Edit' dialog on the right. The report preview displays a table with columns 'Studio', 'Title', and 'Starring'. The 'Title' column is highlighted with a pink box, and a red arrow points from it to the 'Multi-line edit' property in the dialog. The dialog has tabs for 'Categorized' and 'Alphabetic'. The 'Input' section is expanded, showing 'Multi-line edit' set to 'Yes'. Other sections like 'Model', 'Details', and 'Appearance' are also visible.

For the fields you want to expand, change **Control Property->Multi-line edit** to **Yes**.

## 2. Change the Form Property

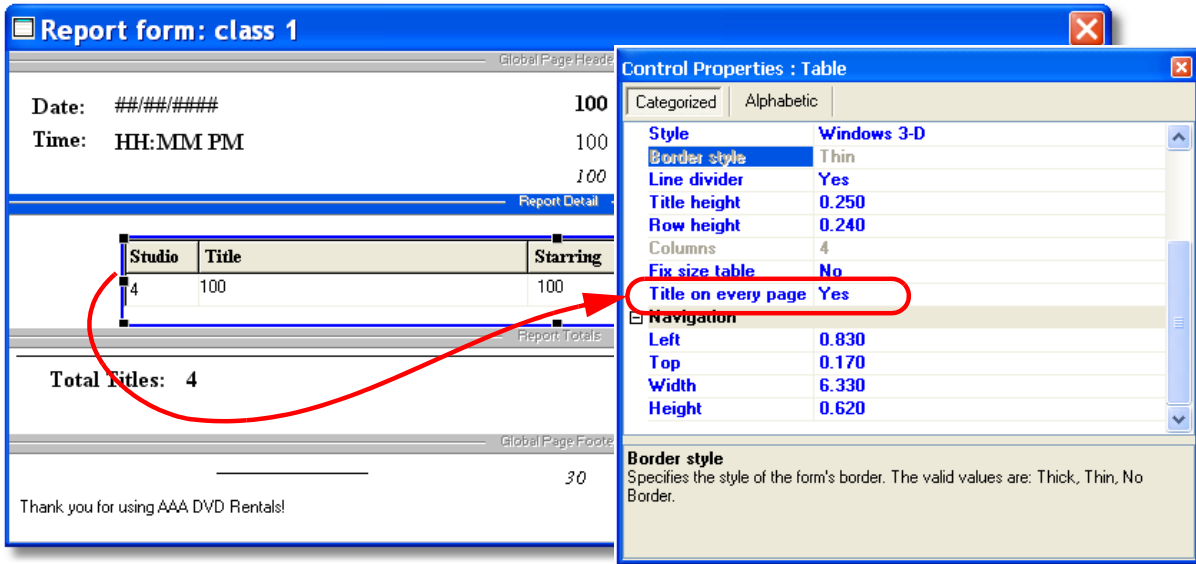


For the Form that contains the expanding field, change **Form Properties->Expand form** to **Multi-page**.

Now the table row will expand when the data will not fit on one row.

**Note:** Sometimes in the Print Previewer, the last line of the wrapped field will not show correctly. This is an issue with the Print Previewer, but the report will work correctly when it goes to a printer.

## How do I Set Repeating Captions for a Table in a Report?



The title at the top of the table prints, by default, once at the top of every page. It also prints when the table repeats, if you have control breaks, for instance.

This is controlled by the Control Property **Title on every Page**. If you want the headers to print on every page, set it to **Yes**. If you want headers on the first page only, set it to **No**.

## How do I Produce PDF Documents?

In the old days, output from programs always was routed to paper. This generated a lot of paper that no one ever used. These days, it's easy to route output to the current standard, a PDF document.

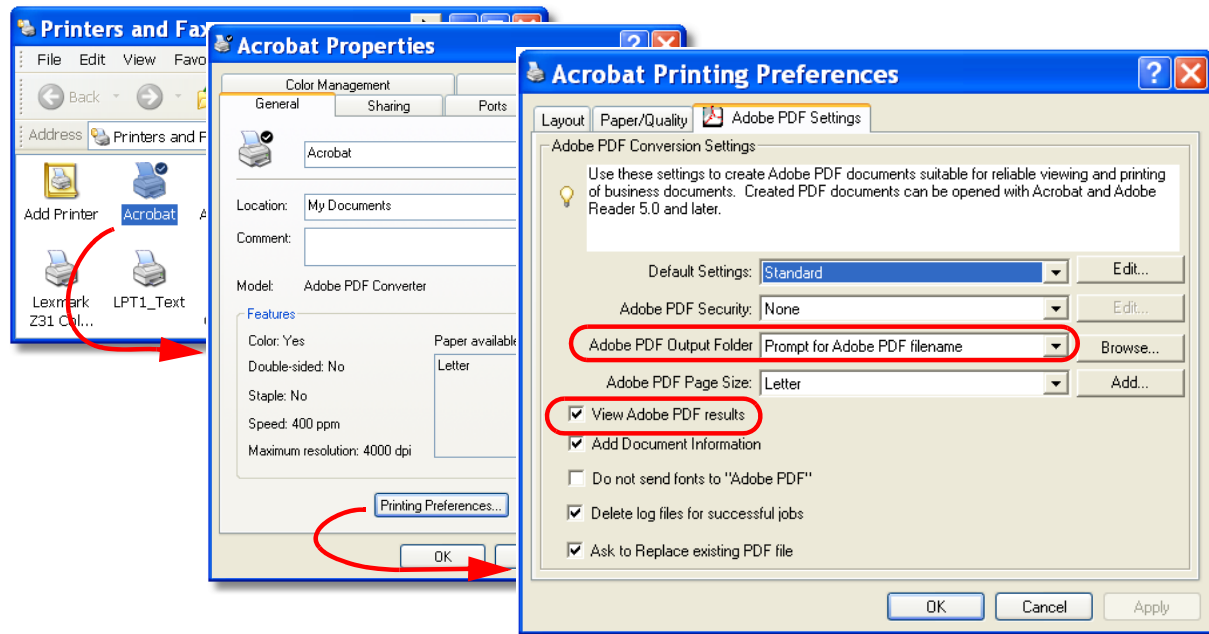
In this section, we will just cover basic PDF production. For more involved PDF manipulation, we suggest you read the documentation that comes with the very useful PDF products on the market.

There are several basic situations where PDFs are useful, and each case is handled a little differently:

- 1) The user is choosing the printer at runtime.
- 2) Setting up PDFs as a default print preview
- 3) The output is a batch report that needs to route to a PDF without user intervention.

Let's go through each of these cases.

### Setting up for PDF when the user chooses the output



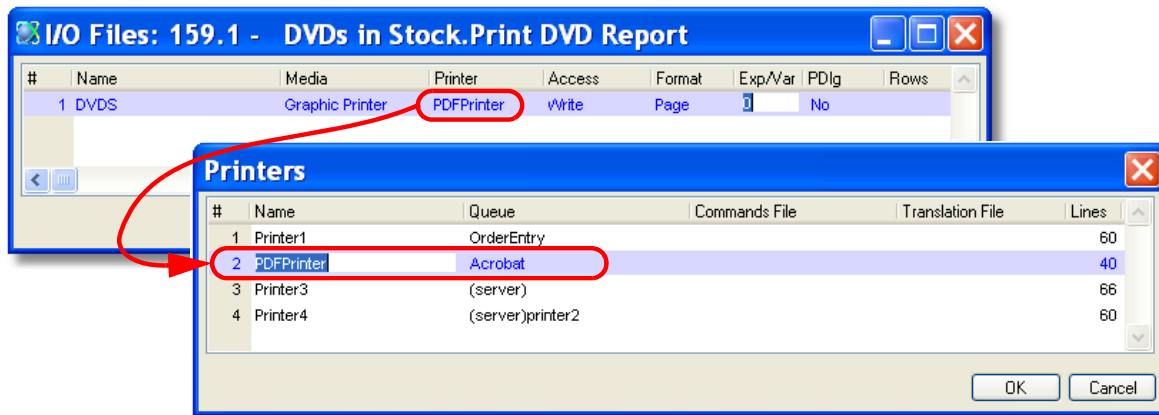
In the first case, the setup is easy.

1. You simply create a Windows printer that routes to the PDF driver. This is generally done automatically when the PDF product is installed, and the details vary per product. Set the PDF driver to automatically open when the PDF is produced.
2. Set **I/O Devices->PDlg** to **Yes**.
3. Set **I/O Devices->Properties->Print Preview** to **No**

Now, the user can choose to route the output to PDF. This doesn't force the user to choose a PDF printer. In fact, they could also choose a fax driver or anything else they have in Windows. What it does is shift the responsibility for choosing the output device to the user, which in most cases is what the users want.

Inside the Windows printer definition, you can choose settings such as whether or not to prompt for a file-name, resolution, and default location. The driver for Acrobat is shown here, but there are several different good products on the market.

### Setting up PDF as a default print preview



1. Create the Windows printer driver, as above. Give the Windows printer driver an easy name (no spaces, easy to spell). The driver in our example is called 'Acrobat'.
2. In **Options->Settings->Printers**, create a printer that you will use for PDF printing. Here we chose 'PDFPrinter', but you can use any printer name you like.

In the Queue field, type in the name of the Windows printer driver. It has to be spelled exactly.

3. Go to your report task, **I/O Devices**. Choose the PDF printer you set up.

Now, the output will route directly to the PDF. If the Windows printer driver is set to automatically open after printing, then the user will have in effect a PDF print preview. This can be very useful, as the users can then add notes or rename the PDF and save it, or email or fax the PDF to other people.

**Note:** The **Commands File** and **Translation File** fields should usually be empty, as these are only used in very specific instances (mostly for code that was imported for older HPL5 printers or some kinds of language translations).

**Hint:** This method is useful if there is a company standard regarding printing. It requires that all the computers all have the a printer driver of the same name and setup installed, so installation needs to be carefully coordinated. The users need to know that the specially-named printer driver cannot be renamed; so it may be useful to name the driver something like "OrderEntry", so they realize that all output from their "Order Entry" system will automatically route to that printer or PDF setup.

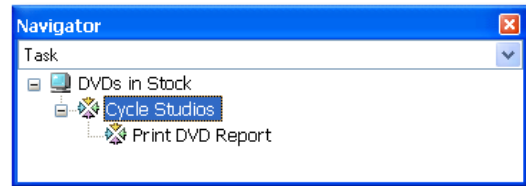
## Setting up a PDF for batch jobs

It is often the case that you will want a report to route automatically to a PDF. For instance, you might have a report that creates PDF output for 50 different salespeople, then emails each salesperson a copy of their report. While a user might launch the original job, it would be tedious to choose a printer and rename the PDF for each of the 50 salespeople.

Working with PDFs in batch is a bit more complex, because of the differences in interacting with the different PDF makers. This example uses Acrobat.

### 1. Create a job that opens the I/O device once per PDF

In our example, we are going to run our DVD report again, but we are going to create one PDF file for each studio.



To do this, we need to open and close the I/O file every time a new studio is encountered. We do that by creating a “driver” job that cycles through the studios, then calls a subtask to print the DVDs for that studio.

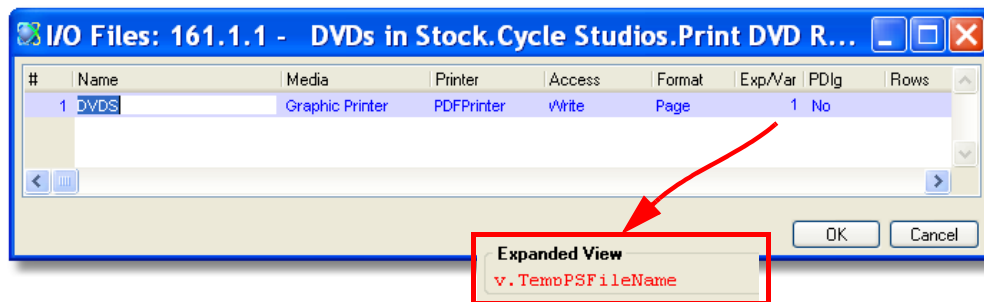
In our new task, we create a new file name for each new PDF that is going to be created, including the studio name. So for instance, the name for studio S001 might evaluate to:

```
C:\Temp\S001DVDS.PS
```

And then we also create a name for the final PDF report:

```
C:\PDFReports\S001DVDS.PDF
```

### 2. Creating the Postscript file



First, you need to create the Postscript file. This is done automatically, via the printer specified in the I/O Device. The I/O device shown here is exactly like the I/O devices we set up in the previous section, except that it also specifies a filename. When the job runs, Acrobat will write the output to this file. In our example, that means the report for studio S001 will be called `C:\Temp\S001DVDS.PS`.

### 3. Converting the Postscript to PDF

Next, you need to convert the Postscript to a PDF file. If you have a Postscript to PDF converter running, such as Adobe Distiller running on the host computer, and write the Postscript file to a watched directory, then you don't have to do any more work. Distiller will automatically convert the output. One problem with this approach, however, is that it relies on the user to have started Distiller and to have it set up correctly. It also means that you lose control of the flow, so your program doesn't know when the PDF is available for other processing, such as sending as an attachment in an email.

You can solve the waiting issue by using a block loop with a **Delay()** function, waiting until the PDF appears. But a more elegant solution is to use a COM object, such as the PdfDistiller COM object (in the library "Acrobat Distiller (Ver 1.0)" ), as shown below.

**Task 161.1 - DVDs in Stock.Cycle Studios**

	Record	Prefix	Suffix
1	Record		
2	Record		
3	Call	SubTask	1 Print DVD Report
4	Invoke	COM	Method PDF Distiller 2.FileToPDF [3 Arguments] Return :

**Properties of : Invoke Operation**

Object: T  
Option: Invoke Method  
Element: FileToPDF  
Arguments: 3  
Value: [empty]  
Return value: [empty]  
Return code: ???  
Condition: Yes

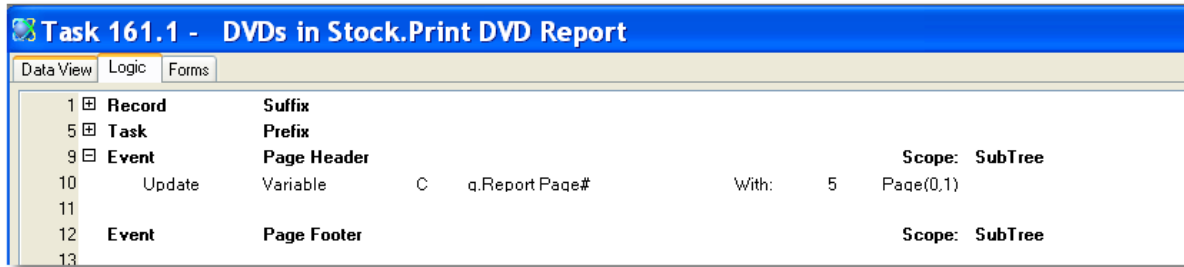
**Invoke FileToPDF Method Arguments**

#	Name	Var	Exp	Internal Type	External Type	Direction	Opt
1	strInputPostScript	???	2	Unicode	VT_BSTR	In	No
2	strOutputPDF	???	3	Unicode	VT_BSTR	In	No
3	strJobOptions	???	1	Unicode	VT_BSTR	In	No

You can control the distillation with other methods, but to get the job done you only need to specify the input file (first parm) and the output file (second parm), and send an empty string for the third parm.

In this example, each time "Cycle Studios" cycles, it will produce one PDF file. You can then add another step to email the PDF or do other processing.

## How do I Implement Page-Based Calculations?



Sometimes you may need to have totals or other calculations done on a per-page basis. For instance, you might want to have the “total cost for this page” or increment the page number. However, for GUI reports, it is very difficult to manually figure out when a page is about to print, because the number of lines that print on a page can vary. So eDeveloper provides you with two internal events: **Page Header** and **Page Footer**, which allow you to do operations just before the Page Header or Page Footer print.

Note that you do not need to use these events to actually *output* the page header or footer, because these can be printed automatically.

**See also:** Chapter 22, “How do I Define Aggregates per Break Level?” on page 603.  
 Chapter 22, “How do I Define Page Header and Footer Information?” on page 588  
 Chapter 22, “How do I Define a Global Page Header or Footer?” on page 590.



## Chapter 23: Merge

---

### How do I Merge Data Into a Text File?

One very useful and versatile facility within eDeveloper is the Merge functionality. While this is often used for creating web applications, it can be used to merge data into any text-type file, including XML, RTF, or HTML.

The task you use for doing a Merge will look a lot like a reporting task, looping through data and using **Form Output** operations to export the data. To find out more about creating a reporting task, see Chapter 22, “How do I Create a Report?” on page 578.

In this section we will go through the basics of the Merge function. More detailed information is found in the other questions in this chapter.

The basic steps for Merging data are as follows:

1. Create your text template
2. Specify your output file
3. Create your Merge form
4. Select the Tags
5. Associate each Tag with data
6. Output the merge form

Let's look at each of these in detail.

## 1. Create your text template

```

1
2 Dear <!--$MG_Cust>:
3
4 Here is a list of DVDs we have in stock<!--$MGIF_KeywordSpecified> that contain the
  . keyword <!--$MG_Keyword><!--$MGENDIF>.
5
6 If you tell me which ones you would like to order, I'll get them in the mail
  . immediately.
7 Thank you for calling AAA DVD Rentals!
8
9 <!--$MG_Salesrep> (<!--$MG_Salesphone>)
10
11 =====
12   ID Number      Title                          Price
13 =====
14 <!--$MGREPEAT>
15   <!--$MG_SN>    <!--$MG_MovieTitle>            <!--$MG_Cost>
16 <!--$MGENDREPEAT>
17 =====
18

```

The first step in creating a Merge is to create your template. The template file can be any editable file, but in this example we are using just plain text for simplicity.

To create this template we just typed it into the text editor, named it “DVDMerge.txt” and stored it in the working directory. However, we can also edit it from within eDeveloper (see Chapter 23, “How do I Set the Preferred HTML Editor of My Choice?” on page 628).

Prefix = ‘<!--\$MG\_’  
 Tag name = ‘Cost’  
 Suffix = ‘>’

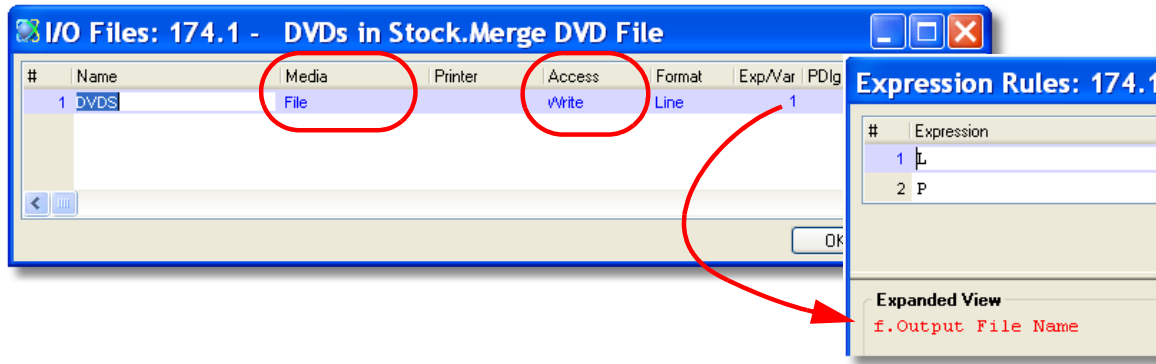


In the places where we have variable data, we use *tags* to tell eDeveloper that we want something replaced with data at runtime. Each tag consists of a prefix (**<!--\$MG\_**), a tag name, and a suffix (**>**). These tags are case-sensitive.

Inside the template, we can also specify repeating data using the **<!--\$MGREPEAT>** tag, and we can specify conditional text using **<!--\$MGIF>**.

**See also:** Chapter 23, “How do I Set Up Replaceable Tokens in a Predefined Template?” on page 626  
 Chapter 23, “How do I Condition Inserting Data into a Predefined Template?” on page 625  
 Chapter 23, “How do I Insert Repeatable Data into a Table Format in a Predefined HTML Template?” on page 624

## 2. Specify your output file



The task you use for creating your output file will look a lot like a reporting task, looping through data and using Form Output operations to export the data. However, there are some differences in how you specify the I/O Device and Forms.

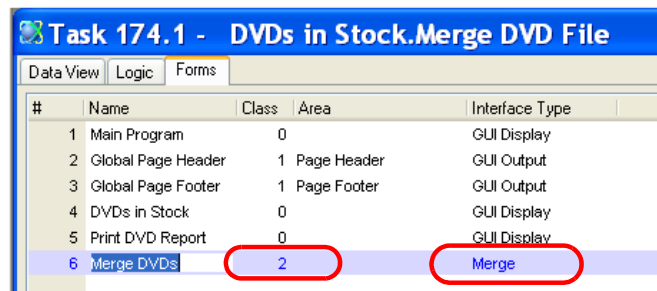
The **Media** type needs to be **File**, and the **Access** needs to be **Write**. Specify the file name in the **Exp/Var** column.

**Note:** If you are doing a Merge for a web application, the **Media** will be **Requester**, and in some instances, where the output will be stored in a BLOB, the **Media** type will be **Variable**. However, that doesn't affect the rest of the instructions here.

## 3. Create your Merge form

Next, you need to create a Merge form. This form is created in the Forms tab, as you would any other Form. However, in this case the **Interface Type** is **Merge**.

Also, you should set the Class to some number that is different from any other Form on the list. In this example, we have a global page header and footer that appear here, which are Class 1. So we made our Merge form Class 2.



#### 4. Set up the Merge form Properties

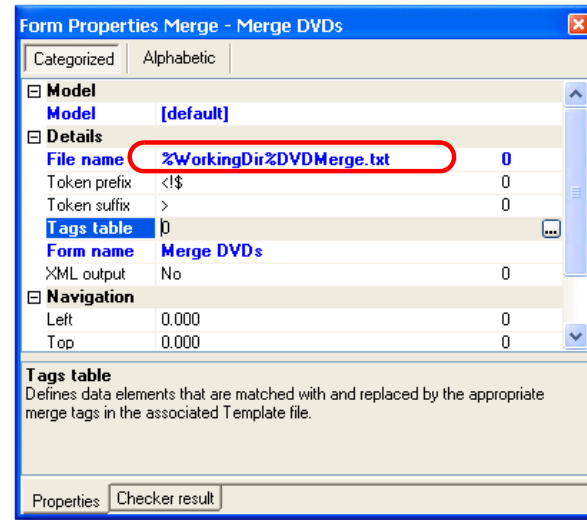
Next, go to the Properties pane of the Merge form (**Alt+Enter**).

The main thing you need to set up here is the File name. This needs to point to your template file. It is a very good idea to use a logical name here, to point to the spot where you keep your templates.

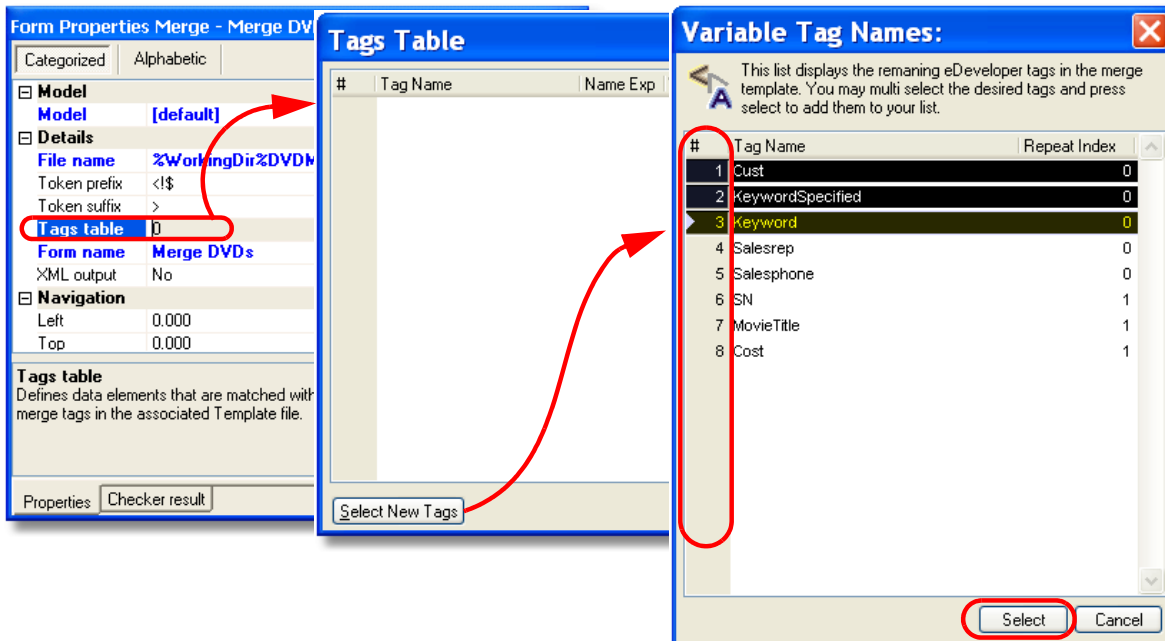
Once you have the template name entered here, you can edit the template by zooming on the Form name in the Form list. That is a good test to make sure you entered the template name correctly!

Also note that you can change the **Token prefix** and **Token suffix** here. If you do that, then your tags will look different. For instance, if you change the Token prefix from “< !\$” to “< # %”, then the Cost tag would need to change from < !\$MG\_Cost> to < # %MG\_Cost>. From a maintenance point of view, it is best to not change the Token prefix or suffix unless it is really necessary.

Once you have specified the file name, **Zoom** from the **Tags table** field to continue.



#### 5. Select the Tags

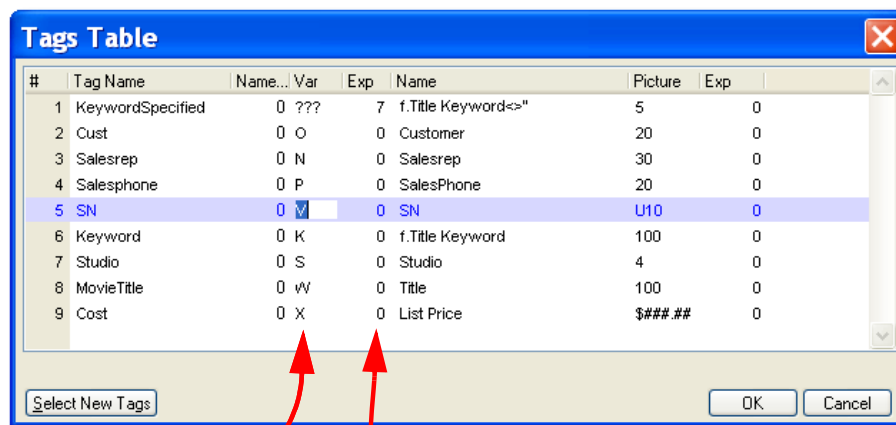


Next, you need to select your Tags. You do this by:

1. Go to **Form Properties->Tags Table**. Zoom (F5 or double click). If it is a new Form, you will see an empty Tags table.
2. Click on the **Select New Tags** button. You should see a list of Tag names from your template.  
If you don't see a list of Tags, then there are a few possibilities:
  - You have already selected all the Tags. If that is the case, then you should hear a beep and see a message at the bottom of the screen, "eDeveloper couldn't find new input tags in Template file".
  - The Tag names are mis-typed (the prefix and suffix don't match what eDeveloper expects).
  - The Template file name is mis-typed. You can check this by zooming from the Form name to see that it comes up in the editor.
3. Select each Tag that you want by clicking in the first column (the one marked "#"). The Tag should turn black when clicked. You can select multiple Tags by using Ctrl+Click. When you are done, press the Select button.
4. Now all the tags you selected should appear in the Tags Table, and you can proceed to the next step.

**Note:** You can also type in the Tag Names, but selecting them is faster and ensures that they are spelled correctly.

## 6. Associate each Tag with data



If the preceding step went correctly, the Tag Names should be filled in on your Tags Table. Now, you need to associate each Tag with a value. The value can be either a **variable** or an **expression**.

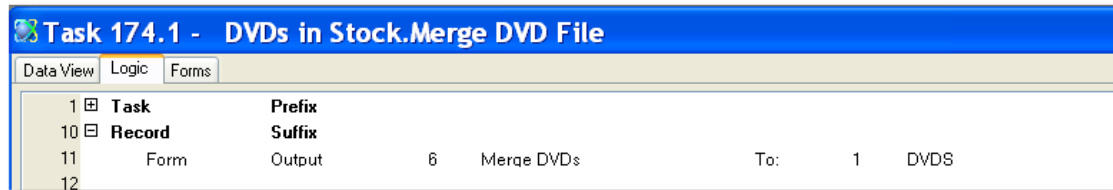
To enter a variable, zoom from the **Var** column. A list of all the variables available will appear. Move the the variable you want and press Enter to select it.

To enter an expression, zoom from the **Exp** column that is next to the **Var** column.

While this process is much the same as selecting data for any report, there are some differences. One major difference is that the data is automatically trimmed. Therefore, even though "MovieTitle" is 100 characters long, it does not print as 100 characters when it is merged.

Another difference is that for a Merge, there typically is no header or footer form. All the data is merged with one template. The MGREPEAT tags determine where the repeating elements are.

## 7. Output the merge form



Last, you need to specify where the form is output. Usually this will be in Record Suffix, as you will be outputting data once from each record processed.

Here is our result:

```

1
2 Dear Frank Smith:
3
4 Here is a list of DVDs we have in stock.
5
6 If you tell me which ones you would like to order, I'll get them in the mail immediately.
7 Thank you for calling AAA DVD Rentals!
8
9 Jill (555.555.1212)
10
11 =====
12 ID Number Title Price
13 =====
14 0784012717 The Boys From Brazil $ 69.98
15 B000077VR3 Moulin Rouge (Single Disc Edition) $ 19.98
16 B000053VB4 Mystic Pizza $ 14.99
17 B000052210 The X-Files - Fight the Future $ 9.98
18 B00003CXCT Star Wars Trilogy (Widescreen Edition) $ 69.98
19 B0000DZ3GQ Midnight Madness $ 14.99
20 B00003CWLK Anna and the King $ 14.98
21 B0006H32DY The Palm Beach Story $ 12.99
22 B0003JAONG Cloak & Dagger $ 9.99
23 B00009A0BK Gotcha! $ 14.99

```

Note how the fields trimmed automatically. The customer name variable was 30 characters long, but the colon after the name is right next to the trimmed name, because the colon was right next to the tag.

However, when the Title field was trimmed, that made the Price column not line up. To get the Price column to line up, we would have to use a Tab character in the template, or use a different kind of format (such as HTML).

## How do I Create a Dynamic Word Document?

Merge

Dear qqqMG\_Custq:

Here is a list of DVDs we have in stockqqqMGIF\_KeywordSpecifiedq that contain the keyword qqqMG\_Keywordq qqqMGENDIFq.

If you tell me which ones you would like to order, I'll get them in the mail immediately.  
Thank you for calling AAA DVD Rentals!

Sincerely:

qqqSalesrepq (qqqMG\_Salesphoneq)

ID Number	Title	Price
<u>qqqMG_SNq</u>	<u>qqqMG_MovieTitleq</u>	<u>qqqMG_Costq</u>

1. Create a document in Word that looks about like what you want.
2. Where you want your tags, add some special string, such as qqqMG\_Custq. Do not use the `< !$MG_Cust>` format, because the special characters will be converted by Word in the next step.
3. Save the document as HTML, then close it in Word.

```

175 </td>
176 <td colspan=1><!--$MGREPEAT-->
177 <tr style=mso-yfti-irow:1;mso-yfti-lastrow:yes;height:17.25pt'>
178 <td width=132 valign=top style='width:99.0pt;border:solid windowtext 1.0pt;
179 border-top:none;mso-border-top-alt:solid windowtext .5pt;mso-border-alt:solid windowtext
180 padding:0pt 5.4pt 0pt 5.4pt;height:17.25pt'>
181 <p class=MsoNormal><span class=SpellE<!--$MG_SN--></span></p>
182 </td>
183 <td width=276 valign=top style='width:207.0pt;border-top:none;border-left:
184 none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
185 mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
186 mso-border-alt:solid windowtext .5pt;padding:0pt 5.4pt 0pt 5.4pt;height:17.25pt'>
187 <p class=MsoNormal><span class=SpellE<!--$MG_MovieTitle--></span></p>

```

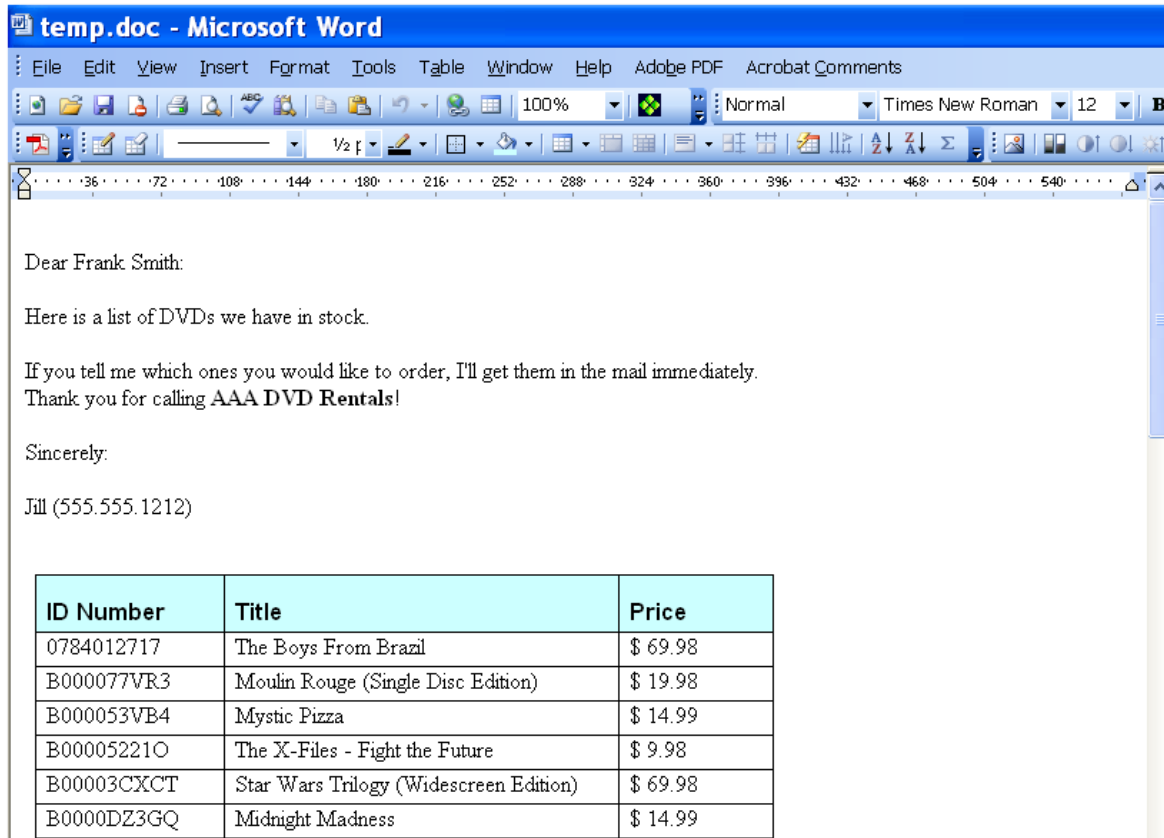
4. Edit the HTML with a text editor. Replace your special tags, such as qqqMG\_Custq, with the eDeveloper tags, such as `< !$MG_Cust>`. Add your MGREPEATs and MGIFs as needed.

**Hint:** If you use a unique string, like qqq, you can let the Find/Replace command do most of the work.

5. Use this HTML file as your Merge template.
6. For your output file name, use the DOC extension.

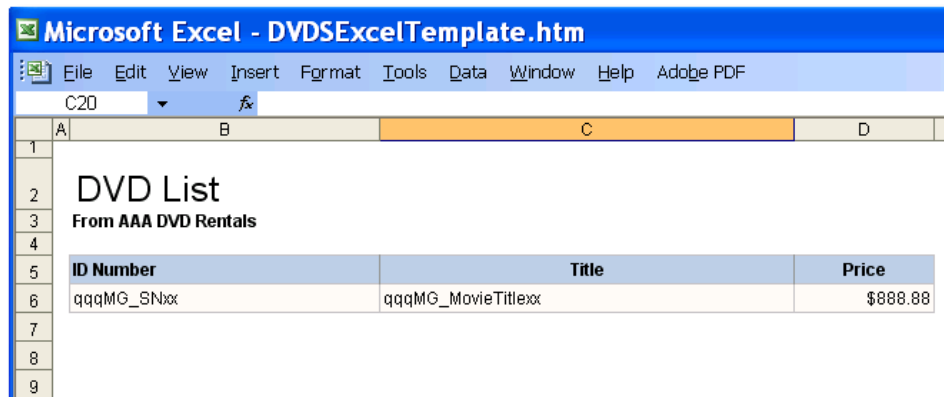
Now, when the document is created, it will open in Word and look like a Word document.

**Note:** This method also works for the RTF format.



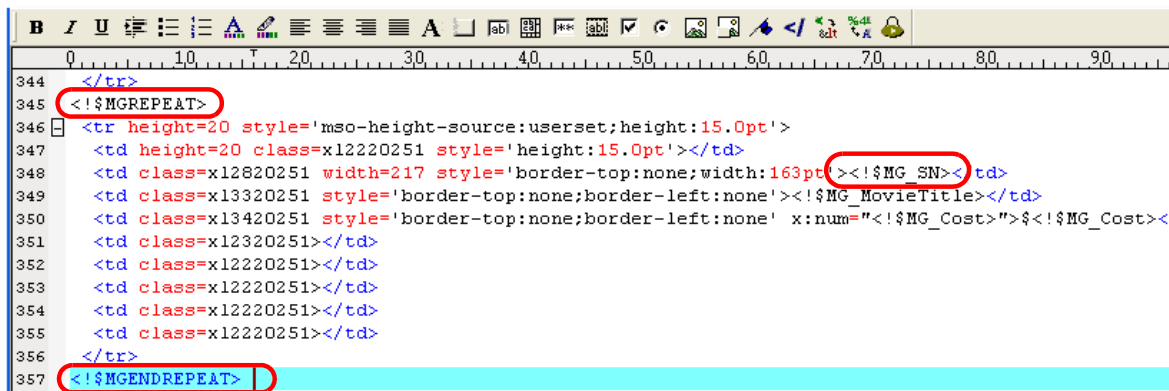


## How do I Create a Dynamic Excel Document?



ID Number	Title	Price
qqqMG_SNxx	qqqMG_MovieTitlexx	\$888.88

1. Create a document in Excel that looks about like what you want.
2. Where you want your tags, add some special string, such as `qqqMG_Custxx`. Do not use the `< !$MG_Cust>` format, because the special characters will be converted by Word in the next step.
3. Save the document as HTML, then close it in Excel.



```

344 </tr>
345 < !$MGREPEAT>
346 <tr height=20 style='mso-height-source:userset;height:15.0pt'>
347 <td height=20 class=x12220251 style='height:15.0pt'></td>
348 <td class=x12820251 width=217 style='border-top:none;width:163pt'>< !$MG SN></td>
349 <td class=x13320251 style='border-top:none;border-left:none'>< !$MG_MovieTitle></td>
350 <td class=x13420251 style='border-top:none;border-left:none' x:num=" < !$MG_Cost>">< !$MG_Cost></td>
351 <td class=x12320251></td>
352 <td class=x12220251></td>
353 <td class=x12220251></td>
354 <td class=x12220251></td>
355 <td class=x12220251></td>
356 </tr>
357 < !$MGENDREPEAT>

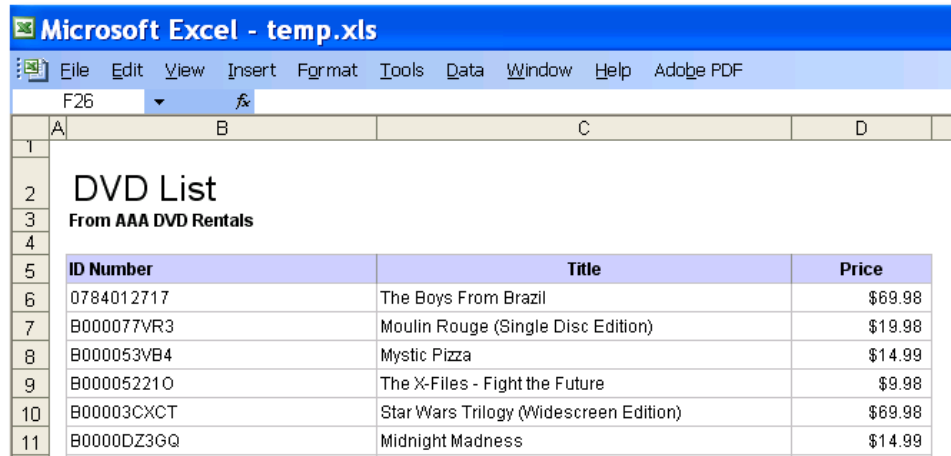
```

4. Edit the HTML with a text editor. Replace your special tags, such as `qqqMG_SNxxx`, with the eDeveloper tags, such as `< !$MG_ SN>`. Add your MGREPEATs and MGIFs as needed.

**Hint:** If you use a unique string, like `qqq`, you can let the Find/Replace command do most of the work.

5. Use this HTM file as your Merge template.
6. For your output file name, use the XLS extension.

Now, when the document is created, it will open in Excel and look like a Excel document.



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - temp.xls". The menu bar includes File, Edit, View, Insert, Format, Tools, Data, Window, Help, and Adobe PDF. The active cell is F26. The worksheet contains a table with the following data:

ID Number	Title	Price
0784012717	The Boys From Brazil	\$69.98
B000077VR3	Moulin Rouge (Single Disc Edition)	\$19.98
B000053VB4	Mystic Pizza	\$14.99
B00005221O	The X-Files - Fight the Future	\$9.98
B00003CXCT	Star Wars Trilogy (Widescreen Edition)	\$69.98
B0000DZ3GQ	Midnight Madness	\$14.99

## How do I Insert Repeatable Data into a Predefined Template?

Merge

```

10
11 =====
12 ID Number Title Price
13 =====
14 <!--MGREPEAT>
15 <!--MG_SN> <!--MG_MovieTitle> <!--MG_Cost>
16 <!--MGENDREPEAT>
17 =====
18
19

```

ID Number	Title	Price
0784012717	The Boys From Brazil	\$ 69.99
B000077VR3	Moulin Rouge (Single Disc Edition)	\$ 14.99
B000053VB4	Mystic Pizza	\$ 14.99
B000052210	The X-Files - Fight the Future	\$ 14.99
B00003CXCT	Star Wars Trilogy (Widescreen Edition)	\$ 14.99
B0000DZ3GQ	Midnight Madness	\$ 14.99
B00003CWLF	Anna and the King	\$ 14.98
B0006H32DY	The Palm Beach Story	\$ 12.99
B0003JAONG	Cloak & Dagger	\$ 9.99
B00009AOBK	Gotcha!	\$ 14.99

Repeatable data is handled with the `<!--MGREPEAT>` tags. Whatever is between `<!--MGREPEAT>` and `<!--MGENDREPEAT>` will be repeated each time you do a Form Output operation that involves the tags between them. So, in our example, we do a Form Output operation in Record Suffix. The Form updates the tags SN, Movie Title, and Cost, so those will repeat. This is somewhat similar to how the Table control works on a GUI output form.

## How do I Insert Repeatable Data into a Table Format in a Predefined HTML Template?



```

174 </td>
175 </tr>
176 <!--$MGREPEAT-->
177 <tr style='mso-yfti-irow:1;mso-yfti-lastrow:yes;height:17.25pt'>
178 <td width=132 valign=top style='width:99.0pt;border:solid windowtext 1.0pt;
179 border-top:none;mso-border-top-alt:solid windowtext .5pt;mso-border-alt:solid windowtext .5pt;
180 padding:0pt 5.4pt 0pt 5.4pt;height:17.25pt'>
181 <p class=MsoNormal><span class=SpellE><!--$MG_SN--></span></p>
182 </td>
183 <td width=276 valign=top style='width:207.0pt;border-top:none;border-left:
184 none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
185 mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
186 mso-border-alt:solid windowtext .5pt;padding:0pt 5.4pt 0pt 5.4pt;height:17.25pt'>
187 <p class=MsoNormal><span class=SpellE><!--$MG_MovieTitle--></span></p>
188 </td>
189 <td width=108 valign=top style='width:81.0pt;border-top:none;border-left:
190 none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
191 mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
192 mso-border-alt:solid windowtext .5pt;padding:0pt 5.4pt 0pt 5.4pt;height:17.25pt'>
193 <p class=MsoNormal><span class=SpellE><!--$MG_Cost--></span></p>
194 </td>
195 </tr>
196 <!--$MGENDREPEAT-->
197 </table>
198

```

In HTML, the basic format of a table is:

```

<table>
  <tr >
    <td> ..Header stuff ..... </td>
  </tr>
  <tr >
    <td> ..Detail line stuff ... </td>
  </tr>
</table>

```

The trick to converting the table into a Merge template is to have only two rows: one for the header, and one for the MGREPEAT area.

The easiest way to do this is to edit your HTML in whatever tool you are using (Dreamweaver, Excel, Word) and delete all but those two rows.

Then, using a text editor (or the source-code editor in Dreamweaver), surround the last table row with <!--\$MGREPEAT--> and <!--\$MGENDREPEAT-->. So you will get:

```

<table>
  <tr >
    <td> ..Header stuff ..... </td>
  </tr>
<!--$MGREPEAT-->
  <tr >
    <td> ..Detail line stuff ... </td>
  </tr>
<!--$MGENDREPEAT-->
</table>

```

## How do I Condition Inserting Data into a Predefined Template?

Sometimes you will want data to be inserted only if a certain condition is TRUE. This is done using the MGIF tag. The MGIF tag surrounds the conditional data. For instance:

```
Here is a list of DVDs we have in
stock<!$MGIF_KeywordSpecified> that contain the key-
word <!$MG_Keyword><!$MGENDIF>.
```

If the tag **KeywordSpecified** is TRUE, then the text

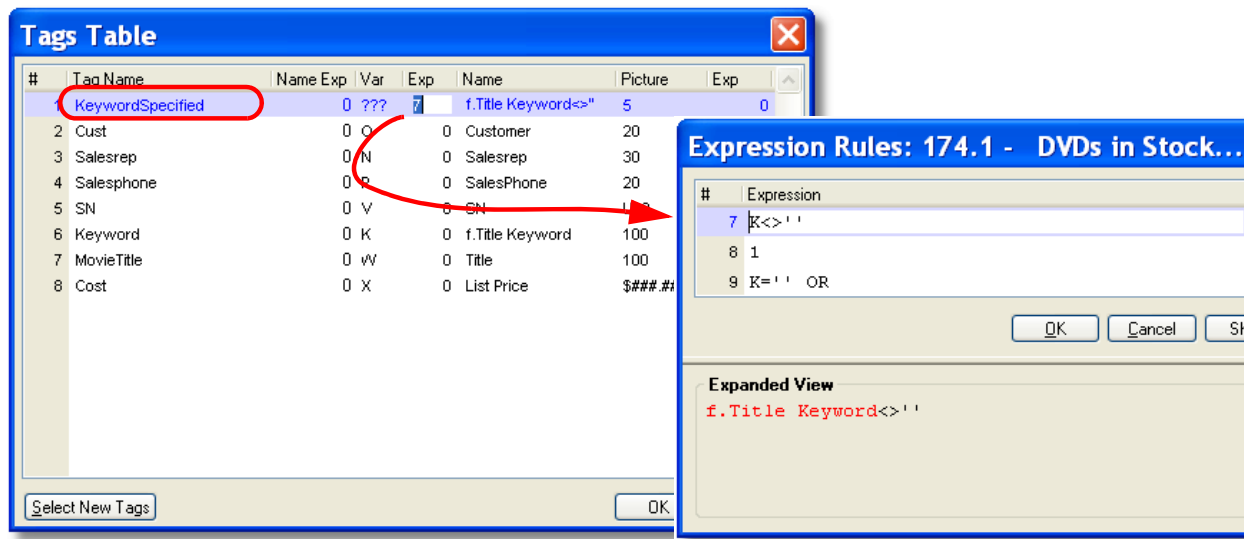
```
that contain the keyword <!$MG_Keyword>
```

will be included. Otherwise, the entire phrase will be omitted, and the sentence will read:

```
Here is a list of DVDs we have in stock.
```

### Specifying a condition

Within eDeveloper, you will need to specify the data that determines if **KeywordSpecified** is TRUE. This is done in the **Tags Table**.



The **Tag Name** needs to match the MGIF tag. The MGIF tag would read:

```
<!$MGIF_KeywordSpecified>
```

The Expression needs to evaluate to a Boolean. In this case, we are checking to see if the user entered any filtering criteria for the list. If such criteria exist, then variable **K** will not be blank, and the expression will evaluate to TRUE.

## How do I Set Up Replaceable Tokens in a Predefined Template?

The easiest way to set up replaceable tokens is:

1. Edit your template using the tool of your choice (Dreamweaver, Excel, or Word, for instance).
2. Where you want replaceable tokens, enter an easily findable string, such as `qqqMG_Custxx`.
3. If you are using Excel or Word, do a **Save As** to save the results in HTML.
4. Change your findable string into a token, such as `< !$MG_Cust>`.

You can see an example of this in Chapter 23, “How do I Create a Dynamic Word Document?” on page 619.

## How do I Differentiate HTML Tags and Merge Tokens?

Merge

```

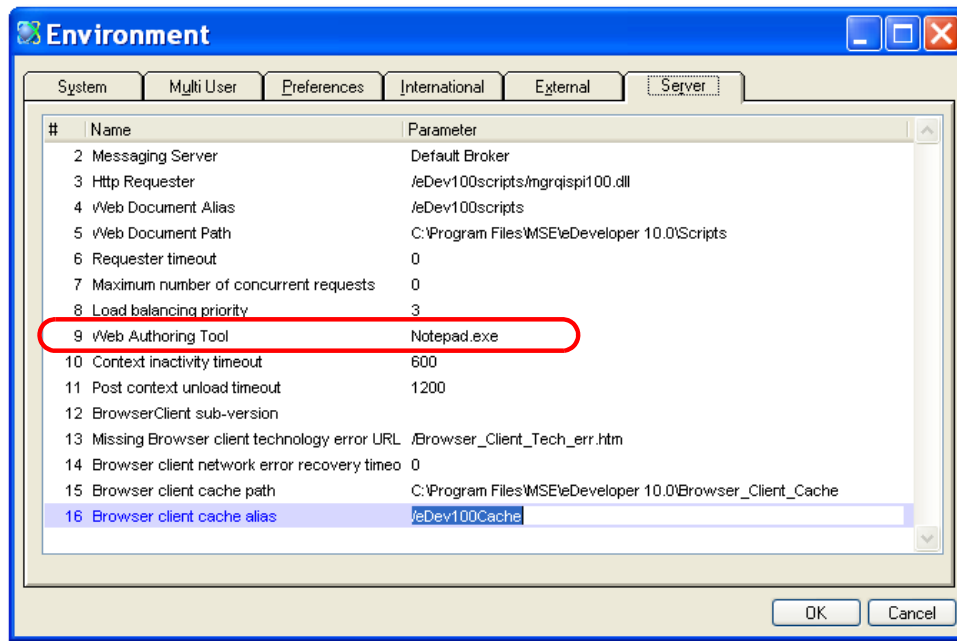
345 < !$MGREPEAT>
346 <tr height=20 style='mso-height-source:userset;height:15.0pt'>
347   <td height=20 class=x12220251 style='height:15.0pt'></td>
348   <td class=x12820251 width=217 style='border-top:none;width:163pt'>< !$MG_SN></td>
349   <td class=x13320251 style='border-top:none;border-left:none'>< !$MG_MovieTitle></td>
350   <td class=x13420251 style='border-top:none;border-left:none'
    x:num="< !$MG_Cost>">< !$MG_Cost></td>
351   <td class=x12320251></td>
352   <td class=x12220251></td>
353   <td class=x12220251></td>
354   <td class=x12220251></td>
355   <td class=x12220251></td>
356 </tr>
357 < !$MGENDREPEAT>

```

HTML Tags and Merge tokens are similar in that both are surrounded by `< >`. However, Merge tokens begin with a specific string, usually `< !$MG_`. You can specify some other first three characters, by changing the entry in Form properties, but the `MG_` is set by eDeveloper.

So, in the code snippet above, you can differentiate between the HTML tags such as `<tr>` and `<td>` and the Merge tokens, because the Merge tokens all start with `< !$MG`.

## How do I Set the Preferred HTML Editor of My Choice?



When you zoom on the name column in a Merge form, eDeveloper will bring up the specified form template. This is very useful, because it allows you to edit the template while you are specifying the tags.

However, there are several different authoring tools available. You can specify which one you want to use by entering it in **Options->Settings->Environment->Web Authoring Tool**. In our example it is set to *Notepad.exe*.

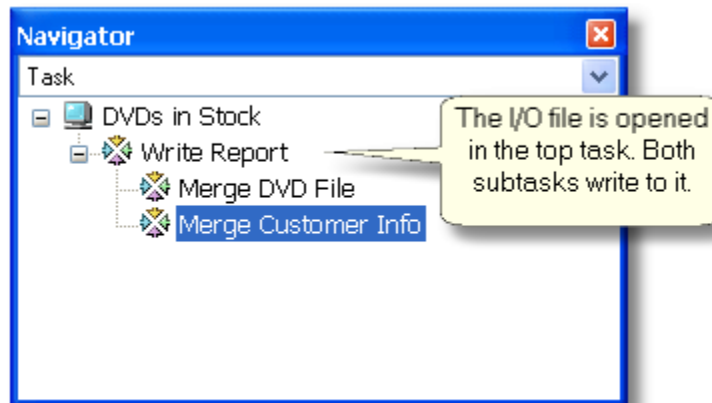


## How do I Merge Data into One Document From Several Tasks?

This question occurs in two different scenarios. The first one is where the separate tasks are part of the same program and share a common ancestor. The second scenario is where the tasks are part of two different programs. The methods are slightly different in each case; let's look at both of them.

Note that the methodology here is the same as that used for printing reports, and is covered in detail in Chapter 22, "How do I Print a Report from Several Programs to the Same I/O Device?" on page 597.

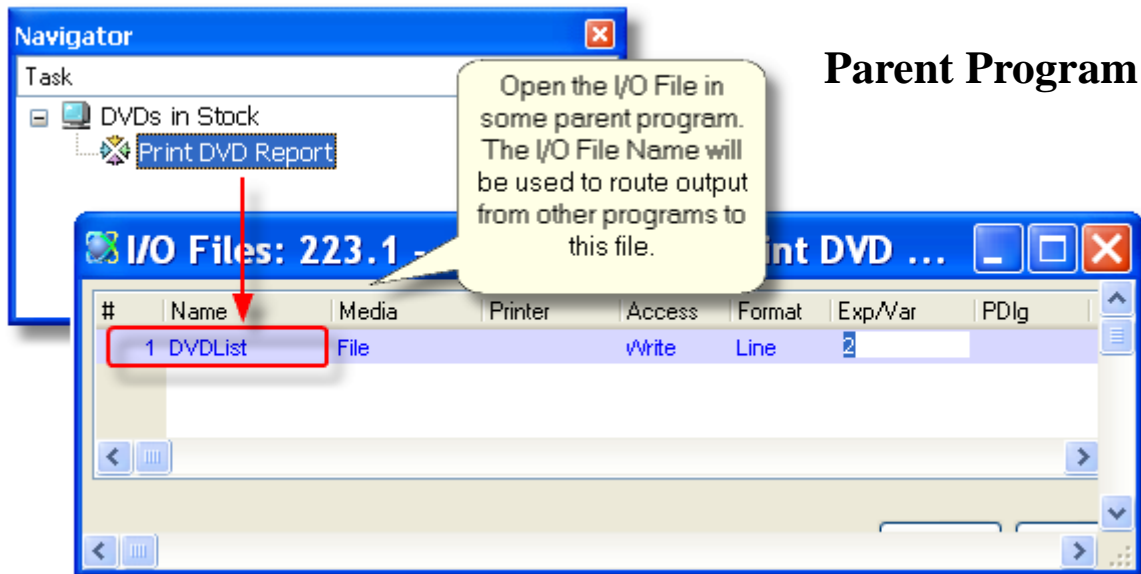
### Merging data from two tasks in the same program



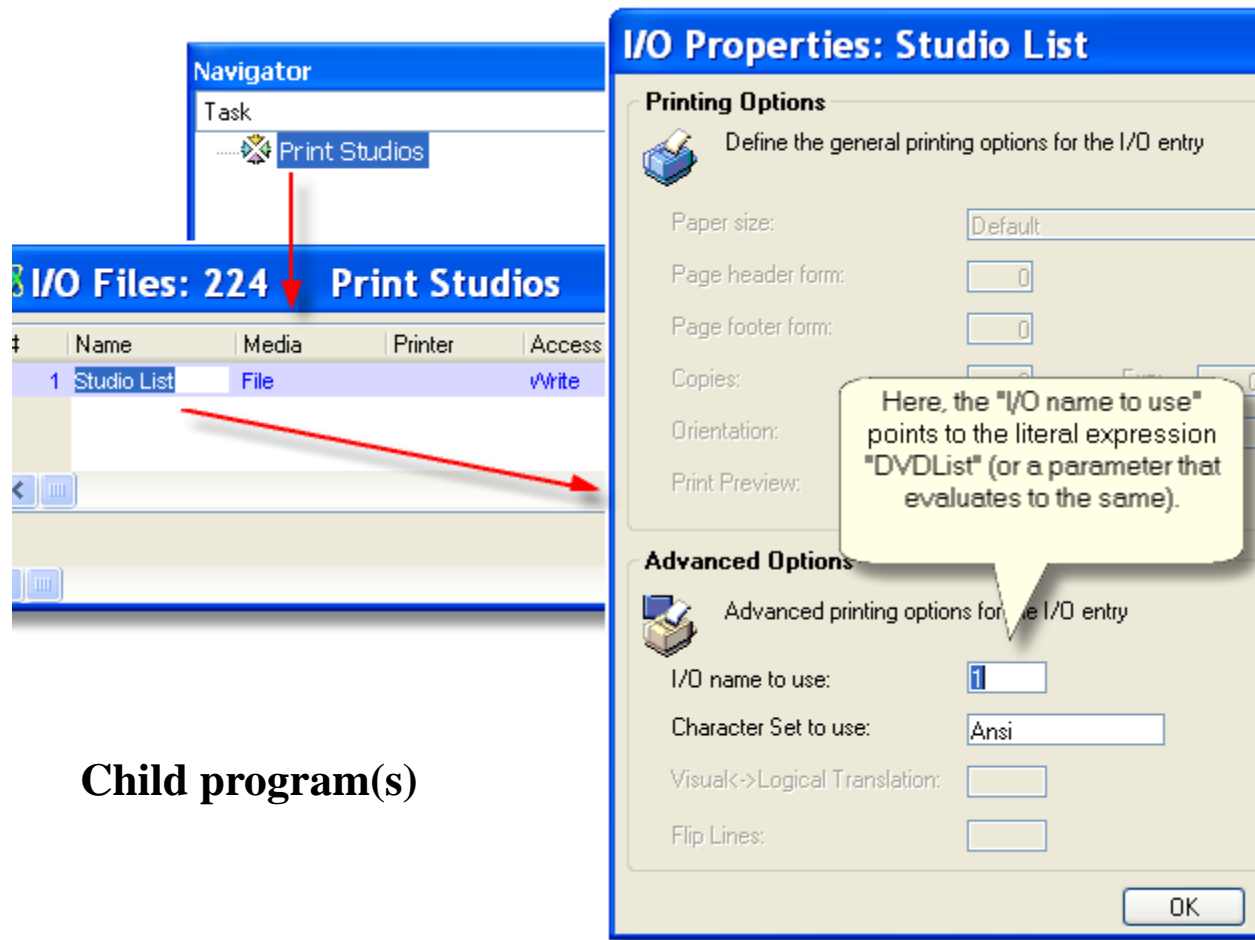
When you have two tasks in the same program that need to merge data into one document, you can handle this as you would for a report: open the I/O File in a parent task that is shared between the two tasks.

### Merging data from two different programs

When you are merging data in two different programs, the I/O Name to Use property routes the output for you. The parent task is set up with an I/O Name, as shown below:



Now, in any program that will have output routed to the same file or requester, this same name is used in the I/O Name to use property:



The child program will route the output to the file that was opened in the parent, even if a different file name was specified in the child.

**Note:** Be sure to use the same Media type in both programs. If one program uses a File and one uses a Printer, it will pass syntax check but will give inconsistent results at runtime.

## How do I Present Data as Grouped, Within a Predefined HTML Template?

If you are writing data to an HTML template and want to group data, the best way to do that is with an HTML table. You can have multiple HTML tables in one file, and use styling to format them nicely.

For instance, here are two separate lists, grouped together into one HTML form:

Item Code	Description
0790736500	The Postman
B000052210	The X-Files - Fight the Future
B00003CX74	Three Kings

Studio Code	Studio Name
S001	Twentieth Century Fox Home Video
S002	Buena Vista Home Video
S003	Universal Studios
S004	Paramount
S005	Warner Home Video
S006	New Line Home Entertainment

Two different tasks were used to route these to the same file (as shown in Chapter 23, “How do I Merge Data into One Document From Several Tasks?” on page 629).

The template used uses MGREPEAT tags to actually create the HTML table, as shown here.

The first task writes DVD records, and only refers to the first two tags, MG\_SN and MG\_Title.

The second task only writes Studio records, and so writes the MG\_Studio and MG\_StudioName lines.

```

17
18 <TABLE class="MGW_TableControl">
19   <TR class="MGW_TableHeader">
20     <TD>Item Code</TD>
21     <TD>Description</TD>
22   </TR>
23   <!--MGREPEAT-->
24   <TR class="MGW_TableRow">
25     <TD><!--MG_SN--></TD>
26     <TD><!--MG_Title--></TD>
27   </TR>
28   <!--MGENDREPEAT-->
29 </TABLE>
30 <BR>
31
32 <TABLE class="MGW_TableControl">
33   <TR class="MGW_TableHeader">
34     <TD>Studio Code</TD>
35     <TD>Studio Name</TD>
36   </TR>
37   <!--MGREPEAT-->
38   <TR class="MGW_TableRow">
39     <TD><!--MG_Studio--></TD>
40     <TD><!--MG_StudioName--></TD>
41   </TR>
42   <!--MGENDREPEAT-->
43 </TABLE>
44

```

**Studio: S003 Universal Studios**

Item Code	Description
B00003CWLF	Anna and the King
B0006H32DY	The Palm Beach Story
B0003JAONG	Cloak & Dagger
B00009AOBK	Gotcha!

**Studio: S004 Paramount**

Item Code	Description
6305537321	Breakfast at Tiffany's
B00021832C	Murder on the Orient Express
B00005JKFA	Better Off Dead
B00005ALMI	Paris When It Sizzles
B00003CXCG	Sabrina

You can do something similar to group data that is within one data source also. In this example, we grouped the DVD's according to their Studio code.

Here, a separate HTML table is created for each group.

Now, let's see how this was done.

## 1. Setting up the template

```

13 <FORM Name="Studios">
14 <!--$MGREPEAT-->
15 <H3> Studio: <!--$MG_Studio--> <!--$MG_StudioName--></H3>
16 <TABLE class="MGW_TableControl">
17 <TR class="MGW_TableHeader">
18 <TD>Item Code</TD>
19 <TD>Description</TD>
20 </TR>
21 <!--$MGREPEAT-->
22 <TR class="MGW_TableRow">
23 <TD><!--$MG_SN--></TD>
24 <TD><!--$MG_Title--></TD>
25 </TR>
26 <!--$MGENDREPEAT-->
27 </TABLE>
28 <BR>
29 <!--$MGENDREPEAT-->

```

The template uses nested MGREPEAT tags. The inner MGREPEAT creates the “detail” rows, as it would in any report. The tags MG\_SN and MG\_Title will be output in Record suffix, so there will be one line per record.

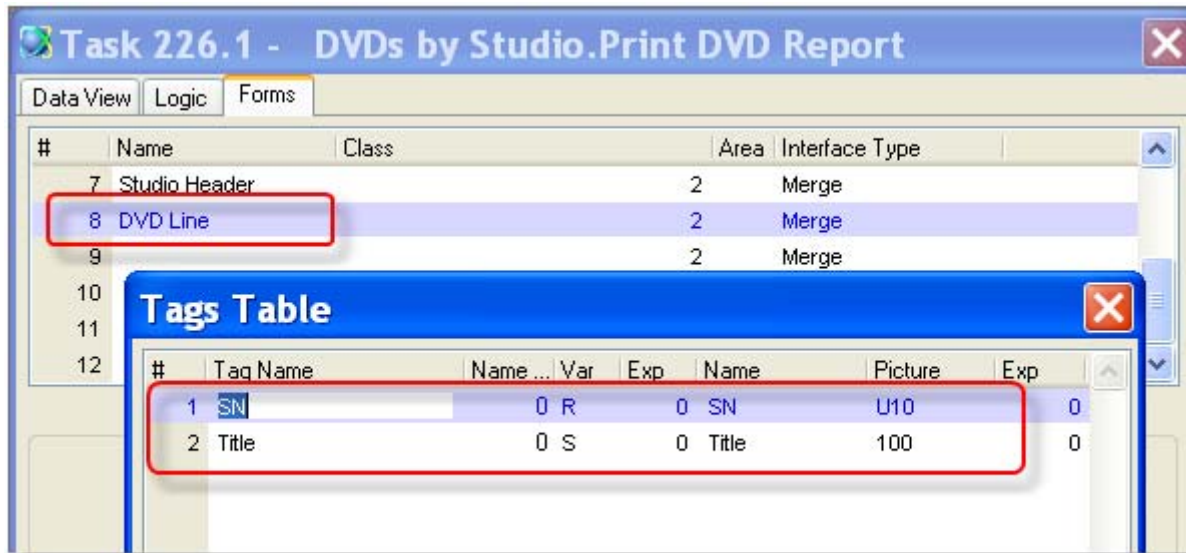
The outer MGREPEAT writes the Studio name in big letters, and it also handles the declaration of the HTML table and the table headers.

## 2. Writing the detail line

Task 226.1 - DVDs by Studio.Print DVD Report							
Data View Logic Forms							
1	Record	Suffix					
2	Form	Output	8	DVD Line	To:	1	DVDList
3							
4	Group	Prefix	of: Q	Studio			
5	Form	Output	7	Studio Header	To:	1	DVDList
6							
7							

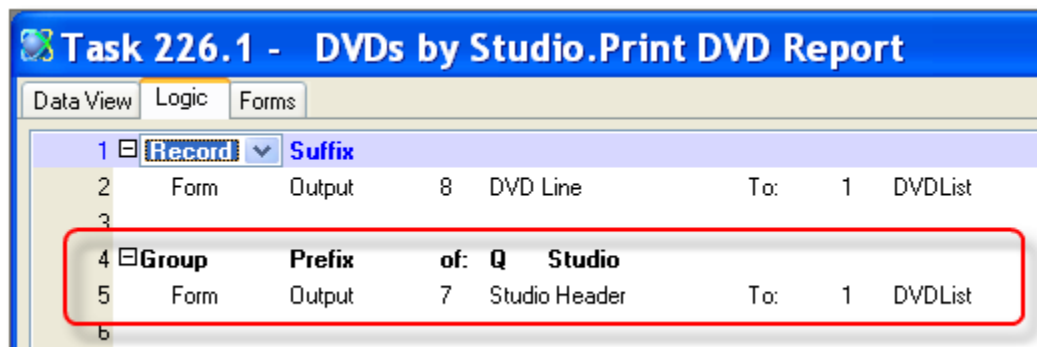
The detail line is written to in Record Suffix, as you would expect in any report.

However, the Form that is being output, DVD Line, only refers to the tags within the inner MGREPEAT.



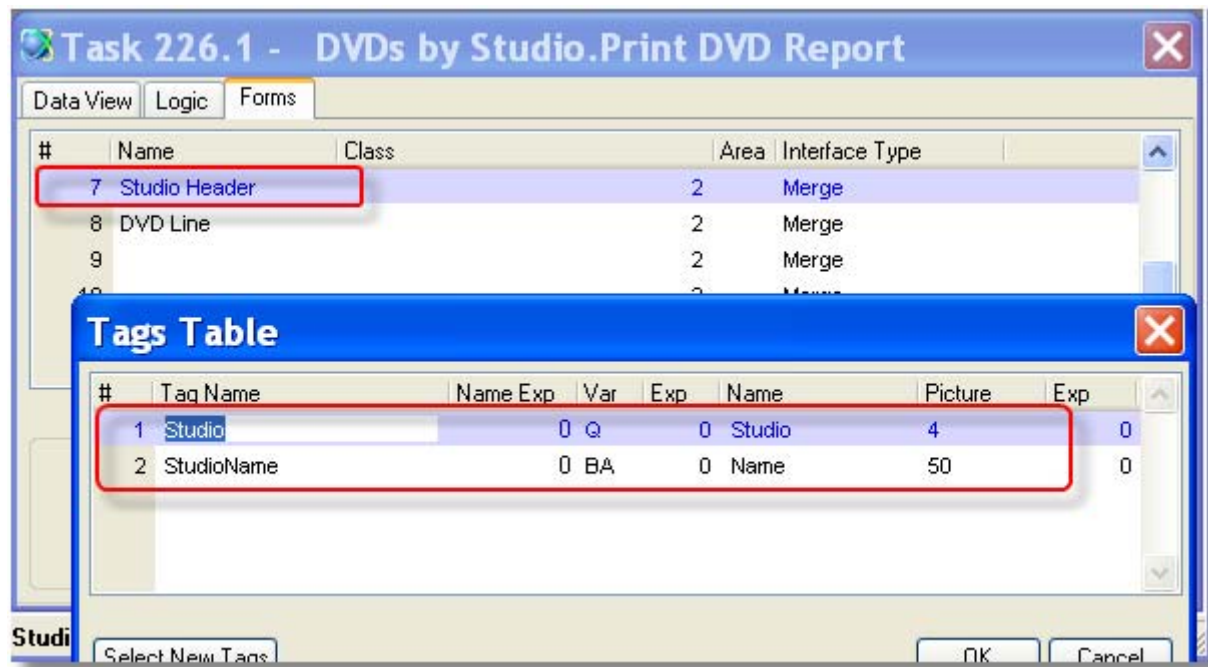
So, the Output Form only causes one more DVD line to be written.

### 3. Writing the header



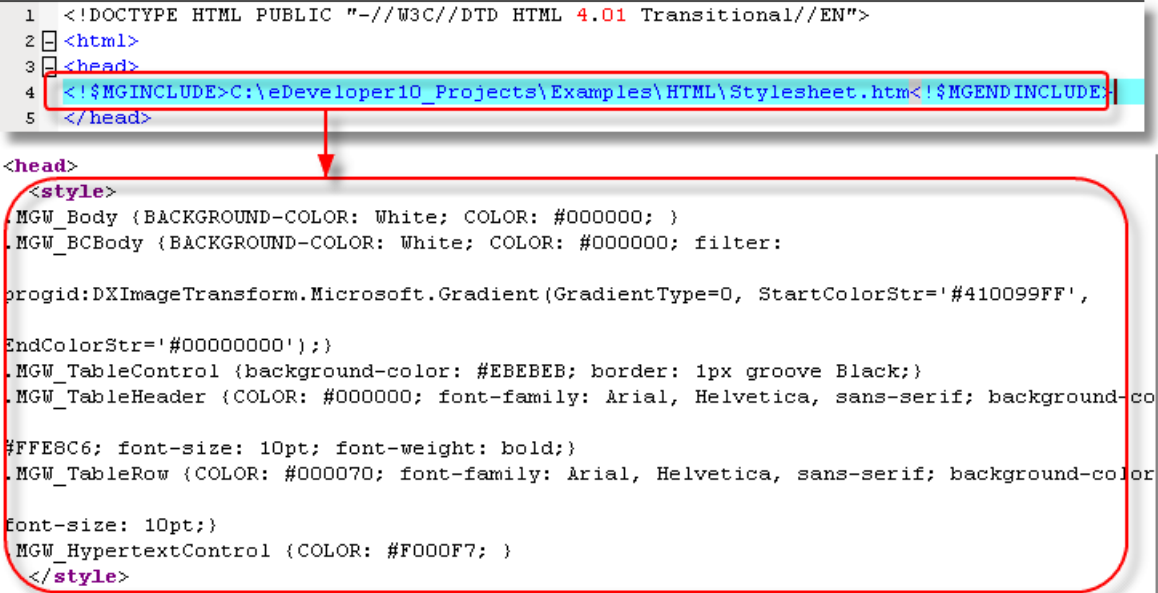
The header is written in a Group Prefix Logic unit. This kind of Logic unit will only execute when the variable it refers to changes. Since the Data View uses the Studio index, the Studio Header will only be written once per studio. This works the same as it would for a GUI report.





The Studio Header form only refers to the tags in the outer MGREPEAT tag. So, when the Form Output is executed, a new <H3> header is written, and a new HTML table is written.

## How do I Embed a File into a Predefined Template?



You can embed entire files, if you like, using the `<!--$MGINCLUDE-->` tag. In our example, we used `<!--$MGINCLUDE-->` to embed our stylesheet into the file at runtime.

### Syntax of `<!--$MGINCLUDE-->`

The syntax of `<!--$MGINCLUDE-->` is:

```
<!--$MGINCLUDE--><file name><!--$MGENDINCLUDE-->
```

Where `<file name>` is the name of the file to include. You can use a tag for the file name; the embedding happens after the file is fully merged.

## Chapter 24: Messaging

---

### How do I send a Message to MSMQ?

You can send messages to MSMQ easily in eDeveloper, using the MSMQ component. You have two basic options for sending messages:

- You can use the use three different programs, to open, send messages, and close the queue. This is the most efficient if you have many messages to send.
- Or, use the quick send program, which does all three steps in one.

We'll cover both methods here.

**Prerequisite:** You need to have MSMQ installed on your computer, and have the eDeveloper MSMQ component loaded into your application. See Chapter 24, “How do I Set Up My Computer for MSMQ?” on page 656.

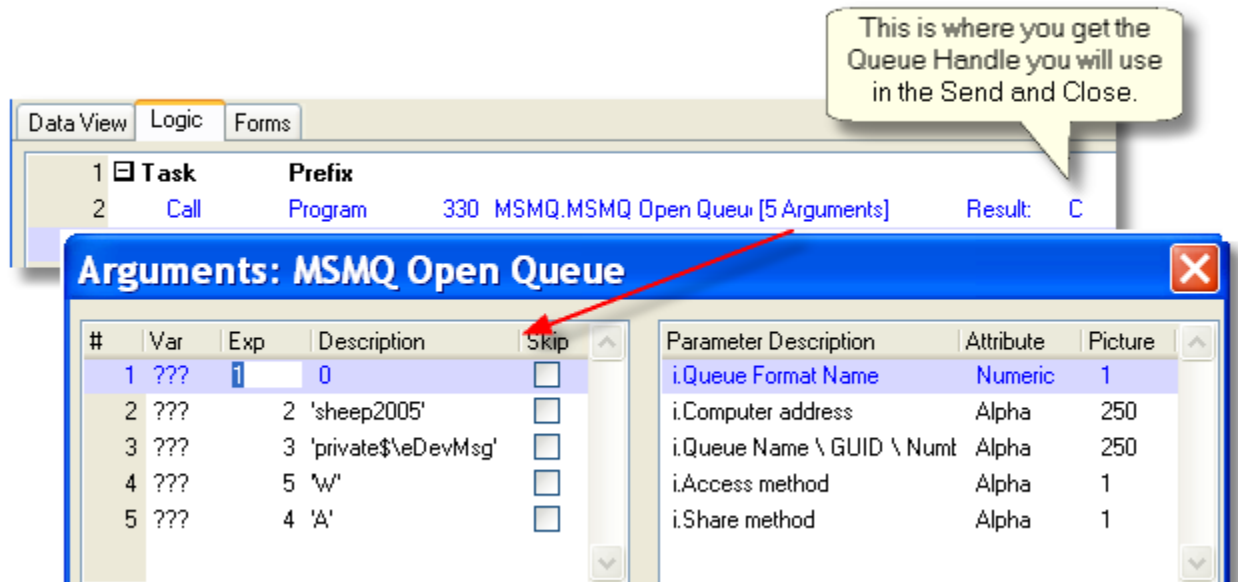
### Using Open, Send, and Close

The first method uses three MSMQ programs:

- *Open Queue:* Opens the Send queue, and returns the queue handle that will be used for sending messages.
- *Send Message:* Send as many messages as needed, on the same queue handle.
- *Close Queue:* closes the queue handle.

Detailed information about these three programs is found in eDeveloper Help. Here we will show a simple example.

## 1. Open Queue



Before you use a messaging queue, you need to open it. When you open the queue, you are determining if the queue is going to be used for reading or writing; if you are doing both at the same time, you will need two different queues.

Similarly, you need to open a unique queue for each destination. You can open a queue that is addressed to one particular computer, as shown in our example, or one that is addressed to a specific IP address, or use a public or private queue.

The first three parameters specify the queue. The queue can be of various types, and the computer address and queue name depend on what kind of queue you are accessing. For instance:

Queue Format Name	Computer Address	Queue Name
0: A computer name	The computer name as set up in Windows	The queue name as set up in Windows components
1: A TCP Address	The IP address	
2: Public	Not required	The queue GUID
3: Private	The queue number	The queue number

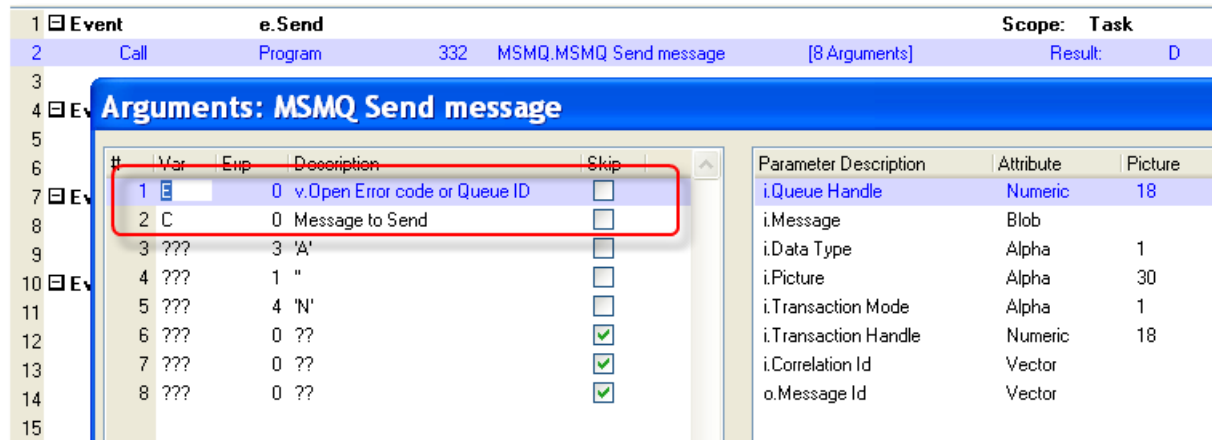
The fourth parameter sets the Access method, which needs to be 'W' for 'Write' if it is a write queue. We set the Share method to 'A' because we are allowing other programs to write to the queue.

If the *Open Queue* program opened the queue successfully, it will return a positive integer, which is your queue handle. You need to store this number, because you will use it to send messages and to close the queue.

If the *Open Queue* program did not work successfully, it will return a negative integer. You can use this to give error messages, but it is easier to just use the *MSMQ.PublicError* event to give the error (see Chapter 24, “How do I Trap Messaging Errors?” on page 652).

**Hint:** For this example, we hard-coded the computer name for readability. You should use the function *OSEnvGet ('COMPUTERNAME')* to fetch the name of the computer you are working on, or store the computer name or IP address in a table, rather than hard-coding it.

## 2. Send Message

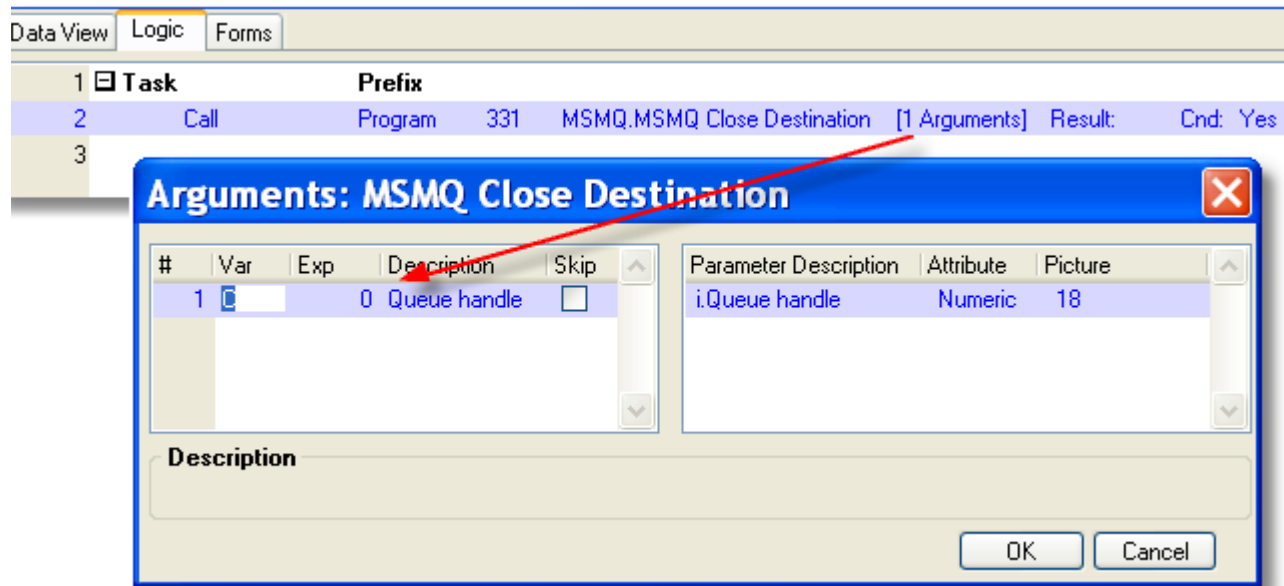


To send a simple message, you will use the following parameters:

1. *Queue handle*: the queue handle you obtained from opening the queue.
2. *Message*: The message you want to send. This should be a BLOB.
3. *Data type*: For a simple text message, use 'A'.
4. *Picture*: This is only needed for numeric data.
5. *Transaction mode*: we used 'N', because we aren't using transactions.
6. *Transaction handle*: not needed, because we aren't using transactions.

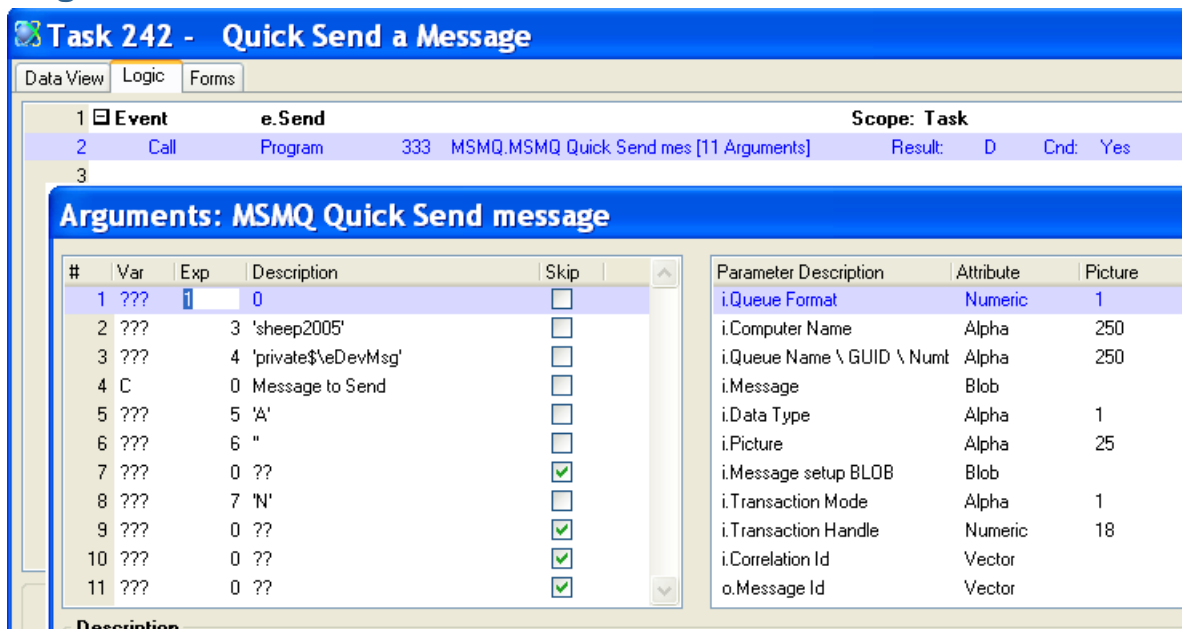
You can send multiple messages without re-opening the queue.

### 3. Close Queue



After you are finished sending messages, you close the queue. Again, use the Queue handle you obtained in the Open Queue step.

### Using Quick Send



Instead of using the Open-Send-Close sequence, you can use the Quick Send program, which does all three in one step. If you are only sending one message, this is an easier method.

Quick Send uses many of the same parameters as the three programs described above, but does them all in one step. For sending a simple text message, only a few of the parameters need to be specified, as shown.

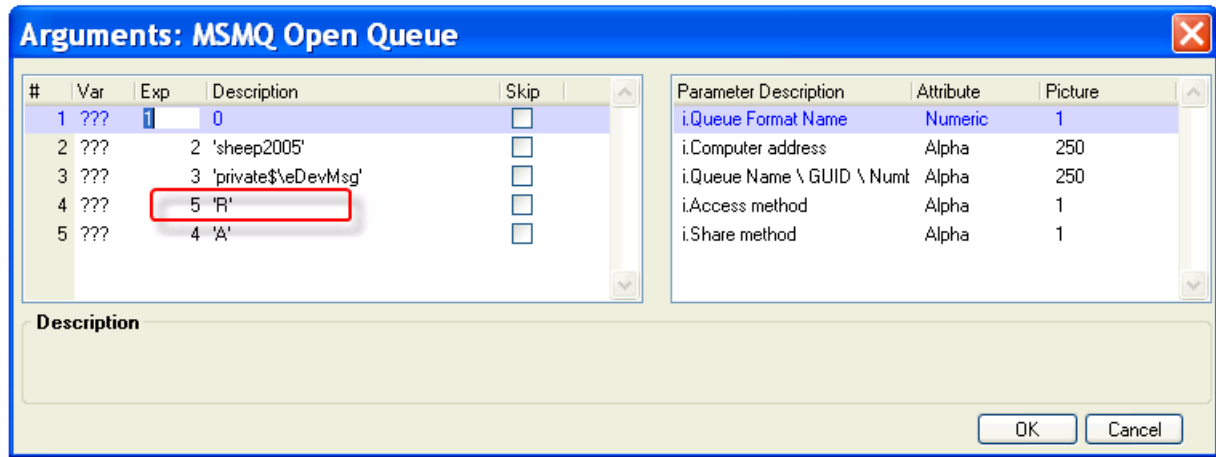
## How do I Receive an MSMQ message?

To receive an MSMQ message, you need to call three MSMQ programs from the component.

- *Open Queue*: Opens the Send queue, and returns the queue handle that will be used for sending messages.
- *Read Message*: Receive as many messages as needed, on the same queue handle.
- *Close Queue*: closes the queue handle.

Detailed information about these three programs is found in eDeveloper Help. Here we will show a simple example.

### 1. Open the Message Queue



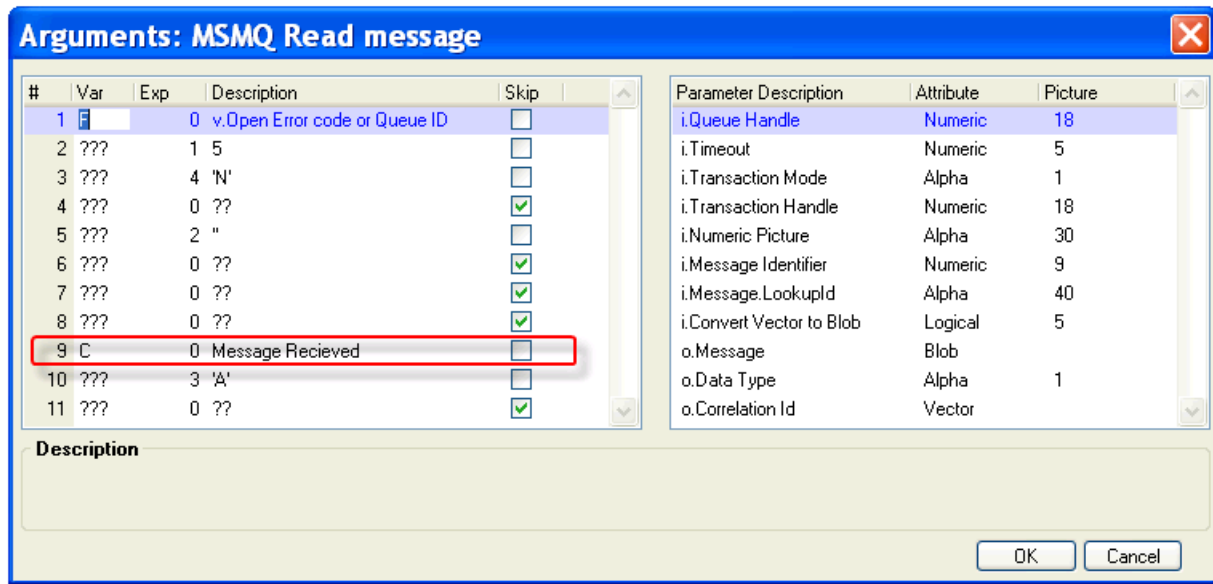
To open the Message queue, call the program *MSMQ Open Queue*, using **'R'** for the fourth parameter. “R”, for “Read”, signifies that you will be reading the message from the queue, and the message will then be removed from the queue. If you use a “V”, for “View”, then your program will read the messages, but they will be left on the queue.

The rest of the parameters are described in Chapter 24, “1. Open Queue” on page 640.

When the queue is successfully opened, it will return a numeric Queue handle. This handle is used as the first parameter in the *Receive* and *Close* programs



## 2. Read messages

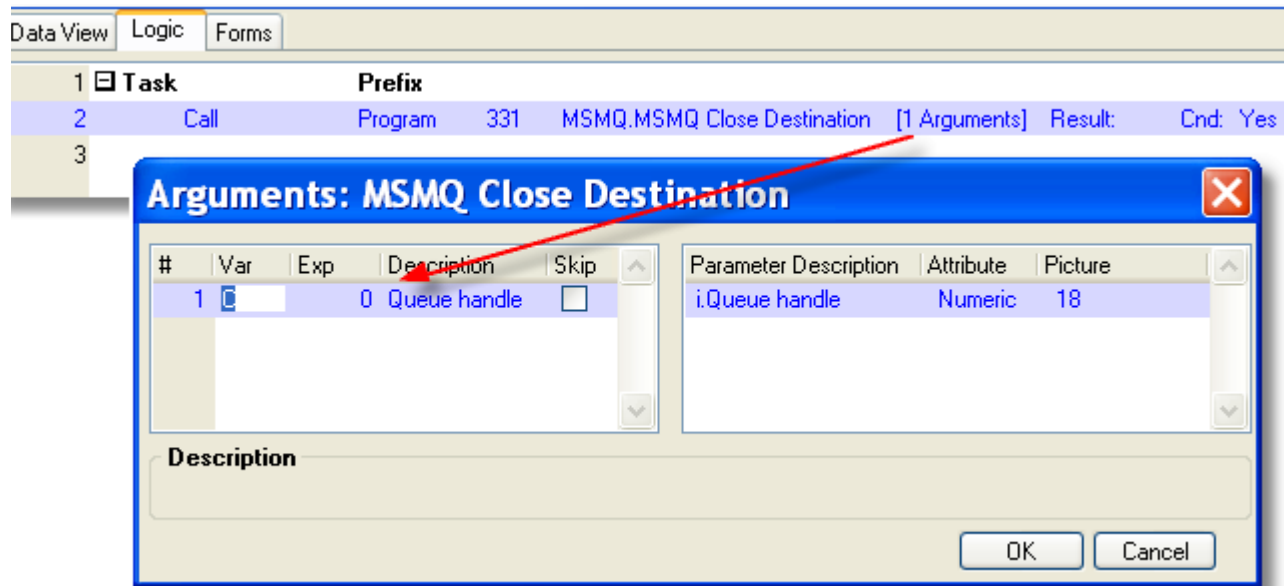


Once the queue is open, you can receive messages using the *MSMQ Read messages* program. Every time the program is called, one message will be read from the message queue, and copied into the BLOB variable which is the 9th parameter.

For simple text messages that don't use transactions, as in our example, you can leave most of the parameters blank.

The Timeout parameter indicates the amount of time to wait for a message in the queue. If you want the task to wait forever until a message shows up, use **-1** for the timeout parameter, and the task will halt until a message arrives. This is how to implement a "listener" task.

### 3. Close queue



After you are finished reading messages, you need to close the queue. Again, use the Queue handle you obtained in the Open Queue step.

## How do I Set a Label for a Message Sent to MSMQ?

<input checked="" type="checkbox"/> Link Write	26	MSMQ.MSMQ External Message Set	Index: 1	Direction: Default
Column 1	Property	[66] Alpha	30	Locate: 1 To: 1 Init: 1
Column 2	Datatype	[67] Alpha	1	
Column 3	Data	Blob		
End Link				

The label goes here.

The datatype of the label ("A" for Alpha)

An MSMQ message has a series of properties defined by Microsoft. The eDeveloper MSMQ component will set these properties for you when it sends the message. All you need to do is to set the property within the component. This is done using the *MSMQ External Message Set* table.

### Setting a the MSMQ Label property

1. Create a *Link Write* operation in the Data view.
2. Link on the *Property* column, using the literal text '**Label**', and init the column to the same value.
3. Update the *Datatype* column to the datatype of the label you are setting (usually '**A**', for Alpha).
4. Update the *Data* column to the value of the label you are trying to set.
5. **Send the table to the component.**

### Send the table to the component

Next, you need to send the table with all your properties to the component. See Chapter 24, "How do I send the MSMQ External Message Table to a the Message Setup Program?" on page 651 for how to do this.

# How do I Set the Priority for a Message Sent to MSMQ?

An MSMQ message has a series of properties defined by Microsoft. The eDeveloper MSMQ component will set these properties for you when it sends the message. All you need to do is to set the property within the component. This is done using the *MSMQ External Message Set* table.

## Setting a the MSMQ Priority property

Link Write

26

MSMQ.MSMQ External Message Set

Index: 1

Direction: Default

Column	1	Property	[66]	Alpha	30	Locate: 1	To: 1	Init: 1	Priority
Column	2	Datatype	[67]	Alpha	1				
Column	3	Data		Blob					

End Link

String version of the numeric priority code.

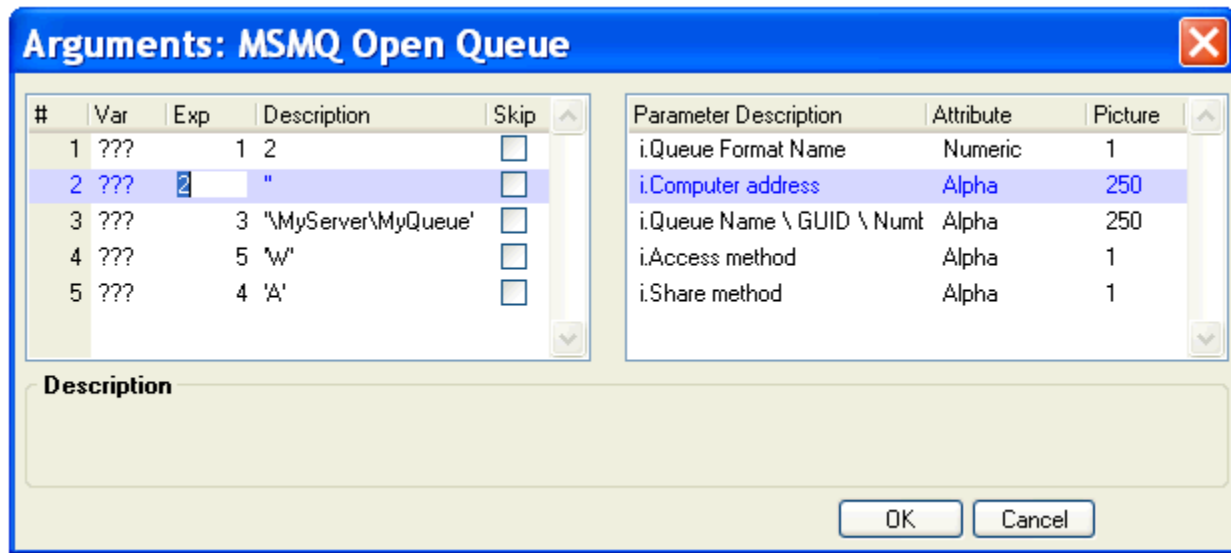
'N' for Numeric

- 1. Create a *Link Write* operation in the Data view.
- 2. Link on the *Property* column, using the literal text '**Priority**', and init the column to the same value.
- 3. Update the *Datatype* column to '**N**'.
- 4. Update the *Data* column to the priority code, in string format. Use the **Str()** function to convert the code to an alpha string, if needed.

## Send the table to the component

Next, you need to send the table with all your properties to the component. See Chapter 24, "How do I send the MSMQ External Message Table to a the Message Setup Program?" on page 651 for how to do this.

## How do I Access a Public MSMQ Queue?



If you want to access an MSMQ Public Queue, you need to open the Queue using:

- *Queue Format Name*: **2** (Public)
- *Computer Address*: " (blank)
- *Queue Name*: the Queue GUID.

The message will then go into the Public Queue, where it can be picked up by other computers that may not be online currently.

## How do I Verify That a Message Sent to MSMQ, Was Read?

The Microsoft MSMQ object includes a system that allows you to verify if, in fact, the message was received by the message queue.

An MSMQ message has a series of properties defined by Microsoft. The eDeveloper MSMQ component will set these properties for you when it sends the message. All you need to do is to set the property within the component. This is done using the *MSMQ External Message Set* table.

### Setting a the MSMQ Ack property

Link Write	26	MSMQ.MSMQ External Message Set				Index: 0	Direction: Default			
Column	1	Property	[66]	Alpha	30	Locate: 1	To: 1	Init: 1	'Ack'	
Column	2	Datatype	[67]	Alpha	1	Locate: 0	To: 0	Init: 0	'N'	
Column	3	Data		Blob						
End Link										

1. Create a *Link Write* operation in the Data view.
2. Link on the *Property* column, using the literal text '**Ack**', and init the column to the same value.
3. Update the *Datatype* column to '**N**'.
4. Update the *Data* column to the acknowledgement code, in string format. Use the **Str()** function to convert the code to an alpha string, if needed.

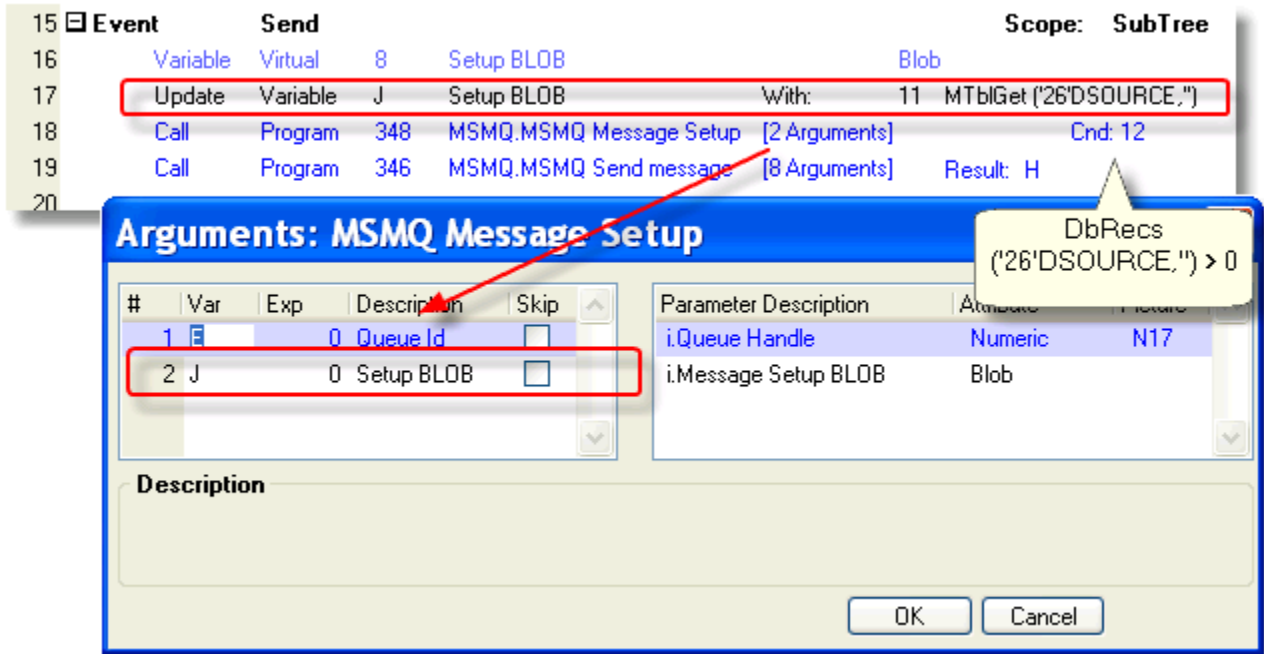
There are several different MSMQ acknowledgement codes, which are listed in the eDeveloper Help. In our example we used 14, which posts an acknowledgment in every case: a positive one if it was received within the time limit, and a negative one otherwise.

### Send the table to the component

Next, you need to send the table with all your properties to the component. See Chapter 24, "How do I send the MSMQ External Message Table to a the Message Setup Program?" on page 651 for how to do this.

Now, when the message is posted, MSMQ will post a response, according to the type of acknowledgement you requested, to the administration queue. You will need to read that queue and respond according to what the application needs.

## How do I send the MSMQ External Message Table to a the Message Setup Program?



Next, you need to package the table into a BLOB and send it to the component. This is done as follows:

1. Package the table into a BLOB using **MTblGet()**.
2. Call the program *MSMQ Message Setup*, sending the BLOB as the second parameter.
3. If you don't always set up the table, condition the *MSMQ Message Setup* call based on whether or not there are records in the setup table.
4. Now, when the message is sent, the Label property will be set to the value you chose.

Now, when the message is sent, the Priority will be set to the value you chose.

## How do I Trap Messaging Errors?

4	Event	MSMQ.Public Error				Scope:	SubTree
5	Variable	Parameter	1	pO.Messaging System	Alpha	1	
6	Variable	Parameter	2	pO.Error code [63]	Numeric	N17	
7	Variable	Parameter	3	pO.Validation error?	Logical	5	
8	Variable	Parameter	4	pO.Error message	Alpha	400	
9							
10	Verify	Warning	8	pO.Error message	Display in	Box	
11							

Whenever you use one of the MSMQ programs, there is the possibility that it will raise an error. Each of the programs has a return code that you can query, however, you can also use an Event handler to automatically trap MSMQ errors. To add this event:

1. Open the task you want to handle the MSMQ error. This can be in three different locations:
  - In the Main Program, with a *Scope* of **Global**.
  - In the top task of the task tree, with a *Scope* of **Subtree**.
  - In the task that is calling the MSMQ program, with a *Scope* of Task.
2. Click on the Logic tab.
3. Press **Ctrl+H (Edit->Create Header Line)** to create a new Logic header line.
4. Type E to create an Event. The Event dialog will appear.
5. Choose *Event type* of **User**.
6. Zoom from the *Event* field to choose **MSMQ.Public error**.
7. Press OK. You will get a dialog box that says “Create Parameter variables to match parameters in the event?”. Select Yes.
8. The Event handler will be created, complete with parameters.
9. Choose the *Scope* that applies in this case.
10. Use the parameter variables as needed to handle the error. In our example we used the message text to create a user error message.

The Event has four parameters:

pO.Messaging system	<ul style="list-style-type: none"> <li>• M - MSMQ</li> <li>• J - JMS</li> <li>• W - WebSphere MQ</li> </ul>
pO.Error code	The error code.
pO.Validation Error?	TRUE if the error is from the MSMQ component's validation check. FALSE if there was an error during execution.
pO.Error message	The error text.



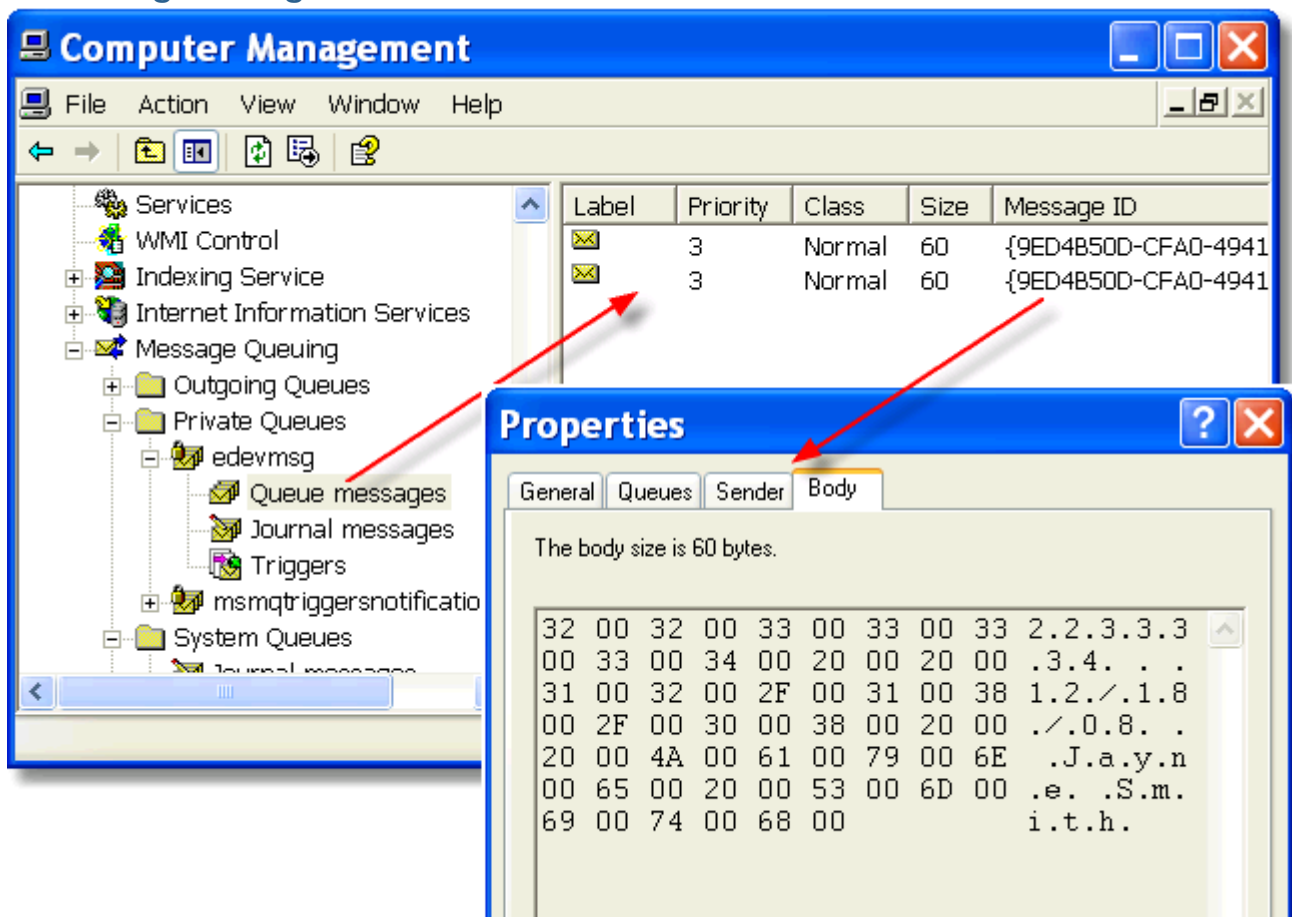
**Note:** While you can trap all the error messages here, remember that they are also written to the message error log. The location of this log is specified under **MessagingComponentDir** in the Logical Names section.

## How do I Debug MSMQ Applications?

Because messaging can involve multiple applications and servers, debugging an MSMQ might be more involved than an eDeveloper-only application. However, you have some good tools for debugging MSMQ at your disposal.

- The Debugger (see Chapter 29, “How do I Debug my Application Using the Debugger?” on page 711) is very useful to see what is going on internally.
- MSMQ errors are written to a log file. The location and name of that log file are set in the logical name **MessagingErrorLogFile**. By default it is set to **%MessagingComponentDir%MG\_message\_err.log**.
- You can view the MSMQ messages from within Windows. We’ll cover this below.

### Viewing Messages in Windows



1. **Start->Control Panel->Administrative Tools->Computer Management**
2. Open the tree node *Services and Applications*
3. Open *Message Queueing*. Now you will see the message queues you have available.
4. Go to the queue you are writing to. In the Queue messages section, you will see the messages that haven't been read off the queue yet.

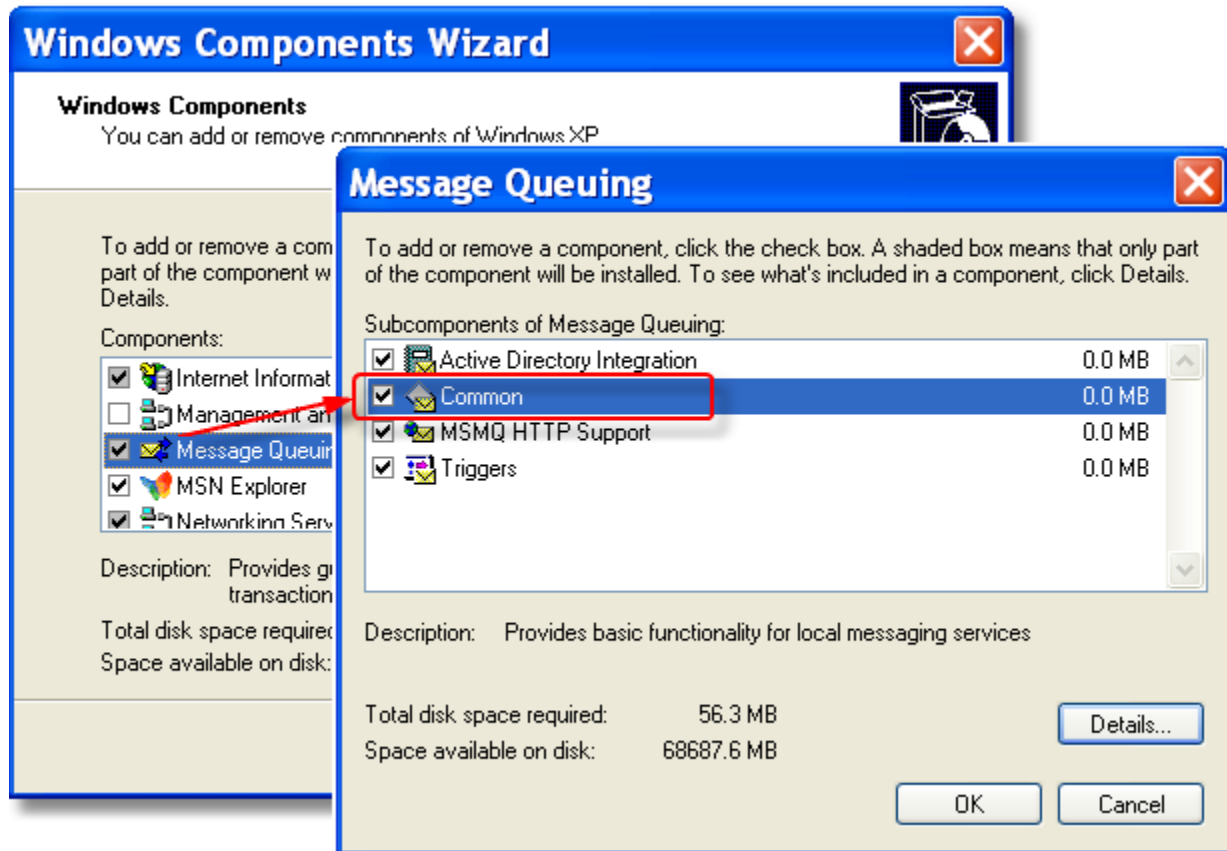
5. Now, for any one message, you can use Properties from the right-click menu to view the internals of the message. You can see the message body, as shown here, and also other properties of the message.

**Note:** You may have to select **Action->Refresh** from the overhead menu to get new messages to display while the Computer Management window is open.

## How do I Set Up My Computer for MSMQ?

MSMQ or Microsoft Messaging Queues is Microsoft's standard for sending messages to queues managed by Microsoft. MSMQ is a part of the Windows 2000 and Windows XP setup but is not part of the "Typical" setup configuration. Let's see how to install it.

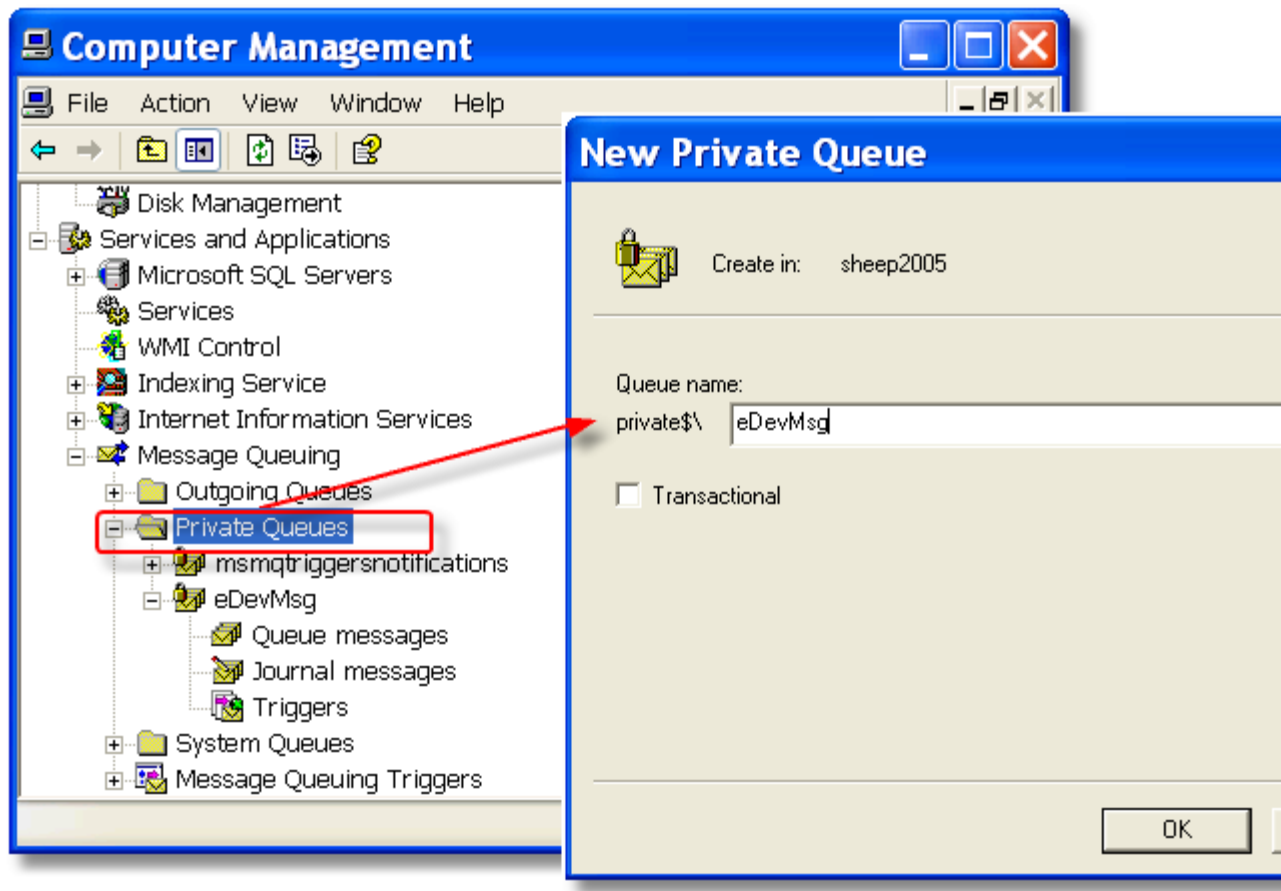
### Installing MSMQ



Before you can work with MSMQ, you need to have it installed on your computer. To do this:

1. Select Windows **Start->Control Panel**
2. Click on *Add/Remove Programs*
3. Click on the *Add/Remove Windows Components* icon on the left.
4. Select *Message Queuing*. Check the check box.
5. Click on the *Details* button.
6. Select *Common*.
7. Keep pressing *OK* to continue the install.

## Adding Queues



Messaging

1. Open **Control Panel->Administrative Tools->Computer Management**.
2. Open **Services and Applications->Message Queueing**. You can add queues to Private or Public queues, but you will only see Public queues if *Active Directory* is installed.
3. To add a queue, select **New->Private Queue** from the right-click menu. (or Public Queue, as needed).
4. A New Queue dialog will appear. Give the queue a name (in our example, it is **\$\eDevMsg**. This is the name you will use when opening up a queue in eDeveloper.
5. Check the *Transactional* box if needed. Some versions of MSMQ do not allow sending non-transactional messages to a transactional queue.
6. Then click OK. The new queue will be created.

## Installing the Messaging Component

This is installed when you installed eDeveloper, if you selected it as part of the installation. If you don't have it, you should rerun the Magic installation, using the Add/Remove option.

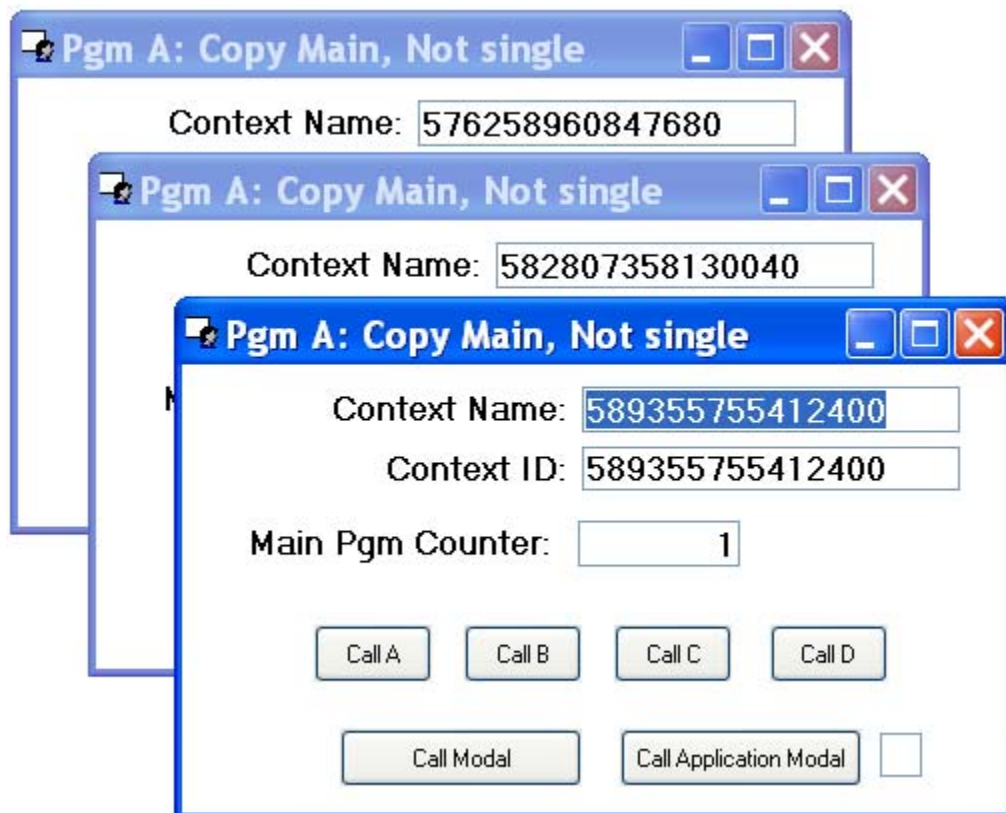
There are two logical names that are used in working with messaging:

- ***MessagingComponentDir:*** this points to the messaging component. The component is an *.ecf* file, and contains programs and handlers for you to use in working with messaging. An *.eci* file is also included, so you can add this component to your application.
- ***MessagingErrorLogFile:*** this points to the location of the error log.

For help in how to set up the component, see Chapter 16, “How do I Load a Component Into My Project?” on page 418.

## Chapter 25: Multi-Tasking

How do I Run More Than One Interactive Task Simultaneously?



By default, eDeveloper programs run one runtime tree at a time. That is, when you open a program from a menu, any other runtime tree that happens to be open will be automatically closed. The program call other tasks, but the tasks work in a hierarchical tree fashion. You can use global events to bring up tasks that are not in the hierarchy, and they will run simultaneously with the main task. However, you cannot keep the same program open with multiple windows.

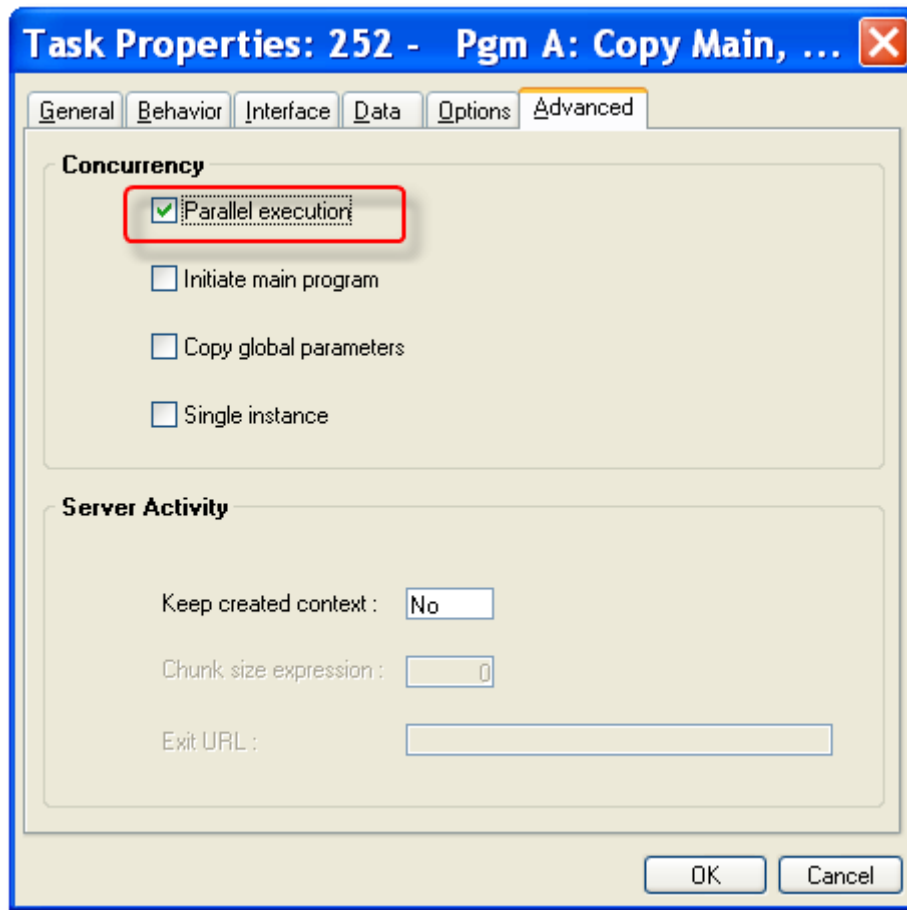
However, you *can* make your programs run in true parallel mode. When the programs are running in parallel, as shown above, each one can start its own hierarchical tree, or call more parallel programs.

Each parallel program has a unique Context ID, which is an automatically-assigned 15-digit number in an alpha field (you can rename it if you want: see Chapter 25, “How do I Set a Different Name for the Current Context?” on page 668).

In our example, we have Pgm A, which is called three times. Whether it is called from the menu or from another program, it has a unique ID and runs independently of the other windows.



## Making a task run in parallel



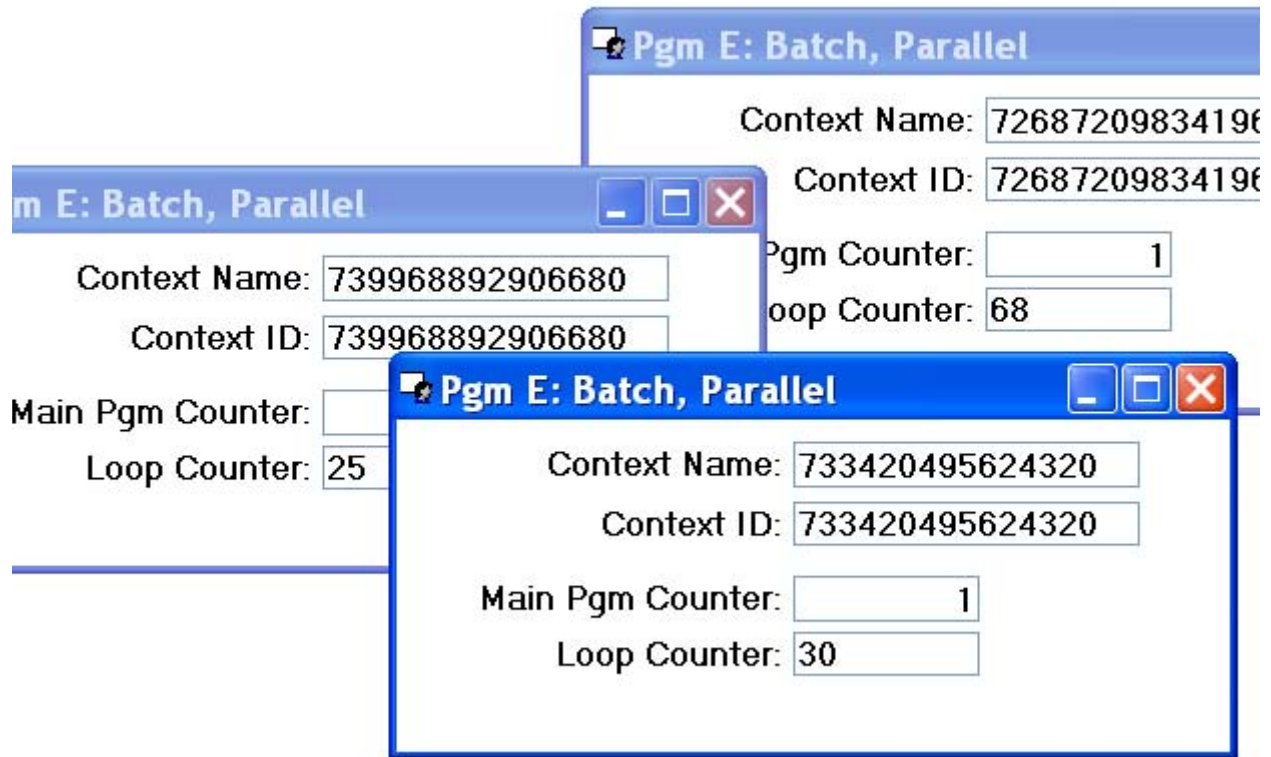
Here is how to make a task run in parallel:

1. Go to *Task Properties* (**Ctrl+P**).
2. Click on the *Advanced* tab.
3. Check the *Parallel execution* box.

Now the program will run in parallel.

When the program runs in parallel, you have some other options about how it runs. You can read more about these in Chapter 25, "How do I Control the Initialization of a Parallel Task?" on page 674.

## How do I run a Report or Batch Process in Parallel to Other Tasks?



Batch processes can run in eDeveloper in the same way that interactive tasks can. Each parallel batch task gets its own context, which closes when the batch task closes.

### Making a batch task run in parallel

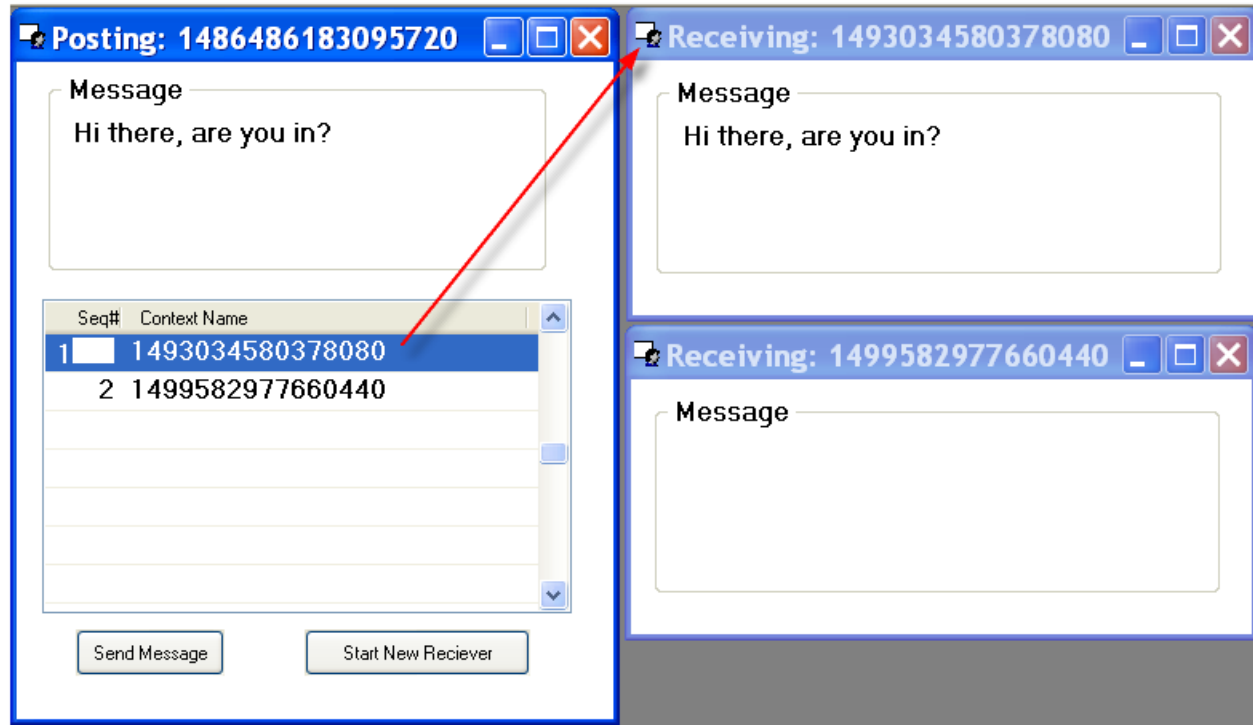
Here is how to make a task run in parallel:

1. Go to *Task Properties* (**Ctrl+P**).
2. Click on the *Advanced* tab.
3. Check the *Parallel execution* box.

Now the program will run in parallel.

## How do I Manage Communication Between Parallel Tasks?

Multi-Tasking



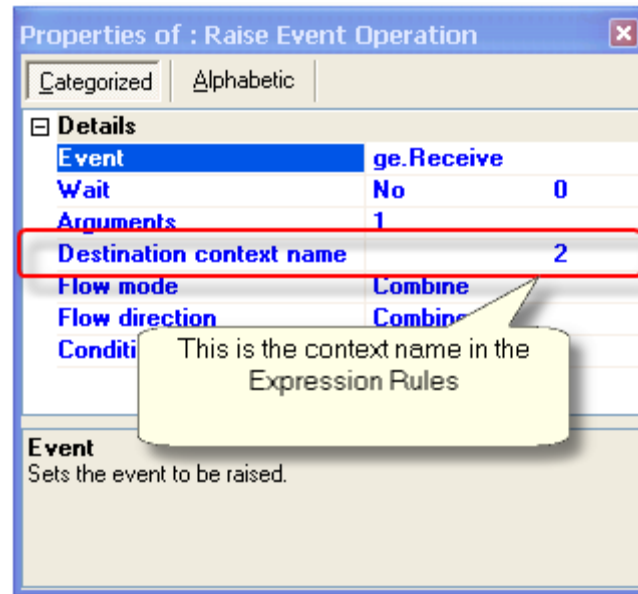
Because parallel tasks run in separate spaces, you cannot pass parameters directly between them, or store data in the Main program. So, in order to “call” a parallel task, you post an event to it. The event can have parameters to pass data into the parallel task. Here is how you do it.

**Prerequisite:** You have to know the context id of the program you are trying to communicate to. See Chapter 25, “How do I Retrieve the Context ID of a Called Parallel Program?” on page 670 for how to do that.

1. Set up a global event that has the parameters you need to pass between the programs. In our example, we set up an event called “ge.Receive”, that has one parameter, the message we want to send. Note that Wait must be set to No.

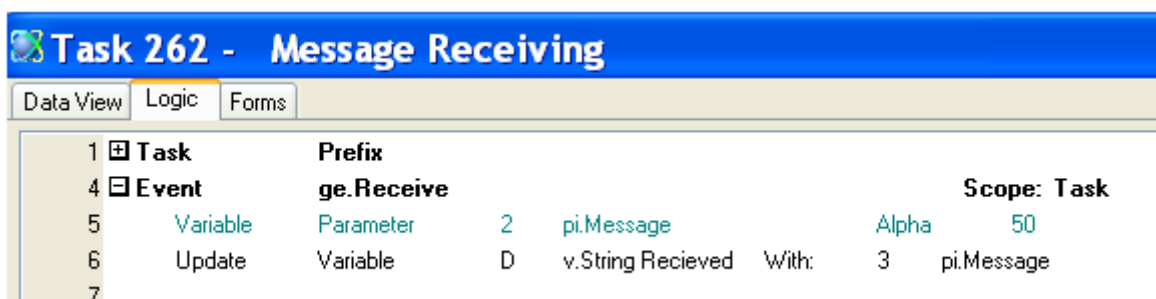


2. In the first program, raise the event, sending the parameter, as shown here. You must set Wait to No or you will get an error when you try to turn it.



3. In the *Event Properties*, set the *Destination context name* to the name of the context you are communicating to. Most of the time this will be some long alpha number string generated by eDeveloper, such as '1493034580378080' in our example. However, there is also the Main context (called 'Main') and contexts can be renamed to any string if you like.

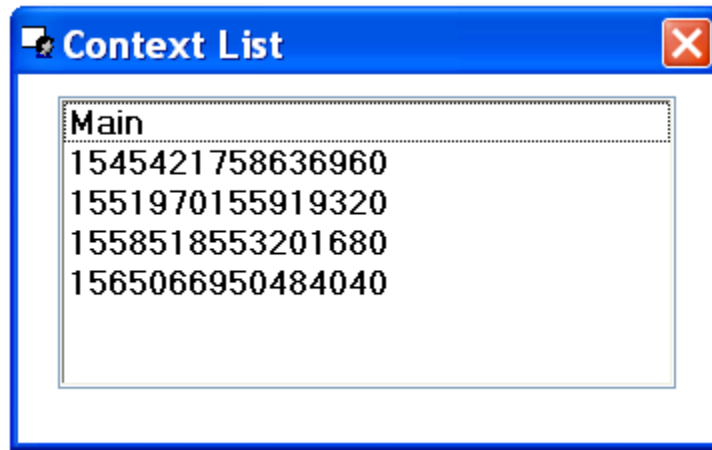
Now, when the event gets raised, it will not be raised in the current context, but in the other context. Note that this gets a little confusing. In this instance, we are raising a `ge.Receive` event while we are trying to *send* a message.



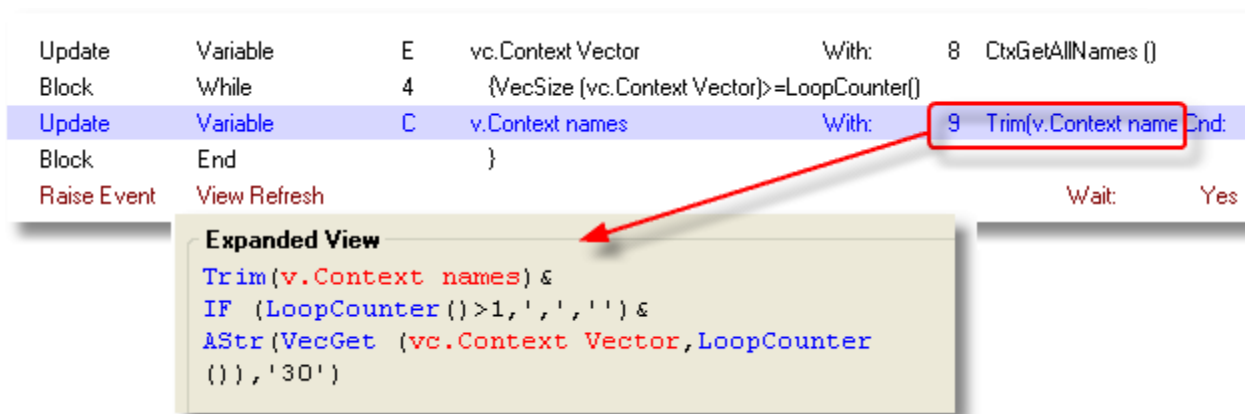
What happens in the other context is that the event gets raised, and it does the appropriate processing. The receiving task doesn't know the origin of the event that sent the data, nor does it need to. The message arrives and is displayed.

However, if this task also needed to send data *back* to the sending program, then it would have to know the context name of the sending program, to reverse the process. This could be done by simply using an additional parameter in the event, which would contain the context name of the sender.

## How do I Retrieve a List of All Running Contexts?



While you were working in the Debugger, you view all the existing contexts on the Context pane of the Debugger. However, if you would also like to retrieve a list of all the running contexts from within your program at runtime, you can use the `CtxGetAllNames()` function. This returns an array of context names, and you can manipulate the array with the usual array functions.



If you would like to display the contexts onscreen, you can concatenate the array into a comma-delimited list, as shown here.

## How do I Fetch the Name of the Current Context?

When communicating between contexts, a program will often need to know its own name, so that it can identify itself to other programs.

To fetch the context name, use the **CtxGetName()** function. There are no parameters. It returns an alpha string, which can be up to 128 characters long. (the context names assigned by eDeveloper are alpha strings of numbers, about 16 characters long, but the programmer can rename the context using up to 128 characters).

## How do I Set a Different Name for the Current Context?

Event	ge.RenameContext				Scope	Task
Variable	Parameter	3	pi.New Context Name	Alpha	15	
Evaluate	Expression	4	CtxSetName(Trim(pi.New Context Name))			
Update	Variable	C	v.My Context Name	With:	2	CtxGetName ()

When you need to rename a context, use the **CtxSetName()** function. Renaming a context is useful when dealing with a single instance context as the name is static and known.

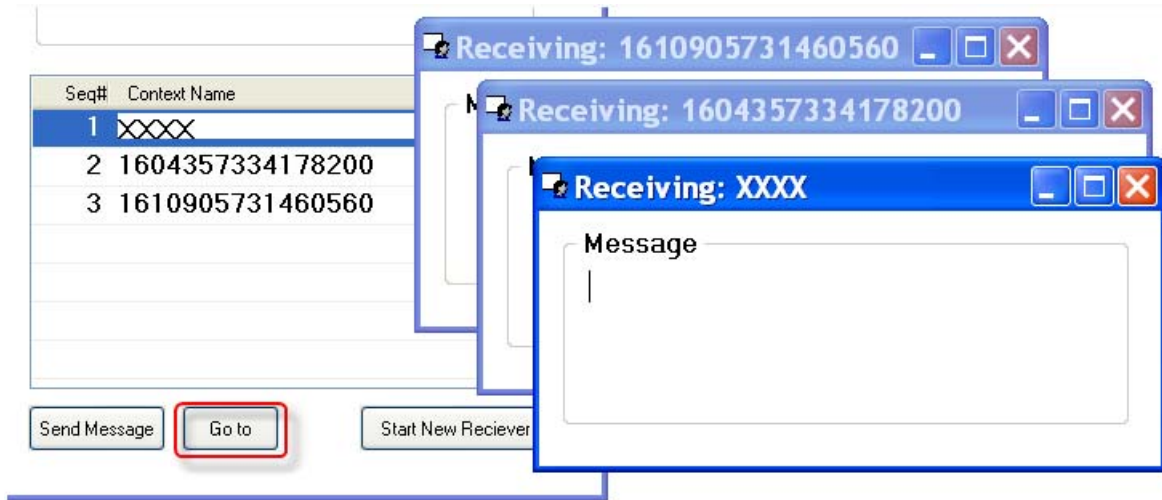
The syntax is:

**CtxSetName(<new name>)** where:

**<new name>** is a text string up to 128 characters long.



## How do I Switch Control to a Different Context?



When the user is working with interactive contexts, the user can just click on the desired context. However, you can also shift control programmatically, by using the **SetContextFocus()** function.

9	Event	e.Set Focus	Scope Task
10	Evaluate	Expression	1 SetContextFocus(Trim(Context Name))

The syntax is:

**SetContextFocus(<name>)** where:

**<name>** is the text string that represents the name of the context. You should use a Trim() function around the name to get rid of extra spaces.

After the function executes, the focus will move to the selected context, if it is found.

The function returns TRUE if the context exists and the focus is switched to that context; otherwise it returns FALSE.

## How do I Retrieve the Context ID of a Called Parallel Program?

The screenshot shows a software interface with a 'Logic' tab. A 'Call Operation' is highlighted in the Logic view. The 'Properties of : Call Operation' dialog is open, showing the 'Details' tab. The 'Returned Context Id' property is set to 'F'. The 'Variable List' dialog is also open, showing a list of variables. The variable 'F Context Name' is selected, and its 'Data Source' is 'Context IDs'.

#	Variable Name	Attribute	Data Source
---	Main Program		
---	Message Posting		
C	v.My Context Name	Alpha	Virtual
D	v.String to send	Alpha	Virtual
---	== Call another Pgr		
F	Context Name	Alpha	Context IDs
I	Original ID	Alpha	Context IDs

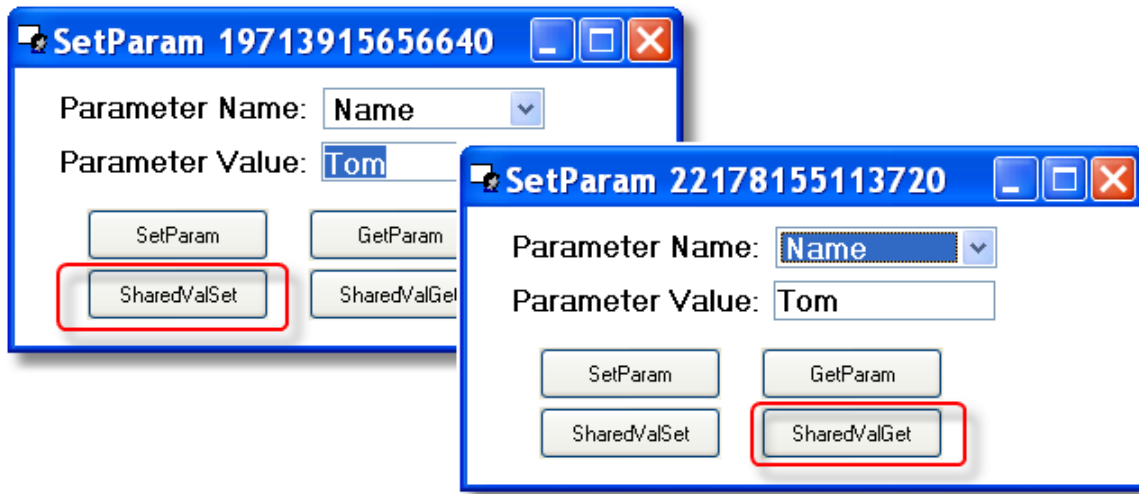
When you call a parallel program, you can easily save the Context ID of the called program by specifying a variable in the *Returned Context Id* property of the Call Operation. To do this:

1. Go to the *Call Program* operation that calls the parallel program.
2. Press **Alt+Enter** to bring up the Call properties.
3. Go to the *Returned Context Id* property.
4. Zoom to select an alpha variable that will hold the returned Context Id.

Now, after the program is called, the variable will contain the Context Id value.

**Note:** The called program does not have any mechanism for discerning the Context Id of the *calling* program. So, if it is needed by the called program, pass it in as a parameter.

## How do I Share Variables Amongst Contexts?



With a single-context environment, are commonly shared by putting them in the Main program, or by using the **SetParam()** function. However, neither of these options works between contexts. In the example above, each window can store values within its own context using **SetParam()** and **GetParam()**. But in order to get two or more contexts to look at the same set of values, the **SharedValSet()** and **SharedValGet()** functions need to be used.

### Using SharedValSet()

6	Event	SharedValSet	Scope Task
7	Evaluate	Expression	5 SharedValSet (Trim(Parameter Name),Value)
8			

The syntax is:

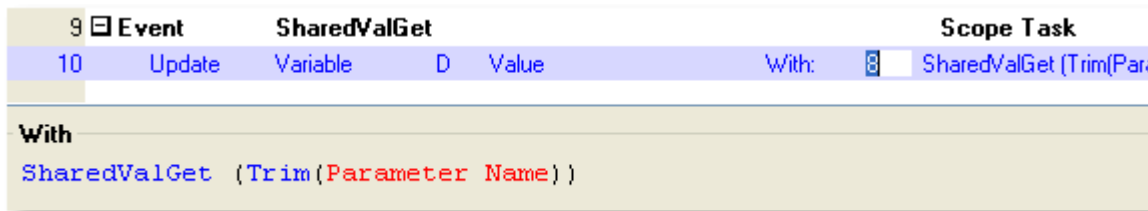
**SharedValSet(<name>, <value>)** where:

**<name>** is the name of the variable. This can be a hard-coded string in quotes, or a variable as in the example above.

**<value>** is the value of the variable, which will be stored. This can be a variety of data types, but it is up to you to make sure that when you fetch it, you fetch it into the correct data type. For instance, if you store a date, you should fetch it back into a date, although eDeveloper will attempt to do the conversion based on the stored data type.

The function always returns TRUE.

## Using SharedValGet()



The syntax is:

**SharedValGet(<name>)** where:

**<name>** is the name of the variable. This can be a hard-coded string in quotes, or a variable as in the example above.

The function returns the value of the variable as it exists in eDeveloper's memory.

**Hint:** *The shared values show up on the variable list of the debugger, at the top of the list.*

## How do I Share Memory Tables Amongst Contexts?

You cannot share memory tables directly between contexts. You can, however, store them in BLOBs and store the BLOBs using the SharedVal functions.

Event	e.Copy Context List				Scope: SubTree	
Update	Variable	F	b.TableBlob	With:	4	MTblGet ('4'DSOURCE,")
Evaluate	Expression	5	SharedValSet ('ContextTbl',b.TableBlob)			

For example, let's store a memory table in shared memory.

- First we use `MTblGet ('4'DSOURCE, '')` to pack Data source #4, a memory table, into a BLOB, `b.TableBlob`.
- Then we use `SharedValSet ('ContextTbl', b.TableBlob)` to store the BLOB in shared memory, under the name 'ContextTbl'.

Event	e.Fetch Context List				Scope: SubTree	
Update	Variable	F	b.TableBlob	With:	6	SharedValGet ('ContextTbl
Update	Variable	G	v.Table Return Code	With:	3	MTblSet (b.TableBlob,'4'D

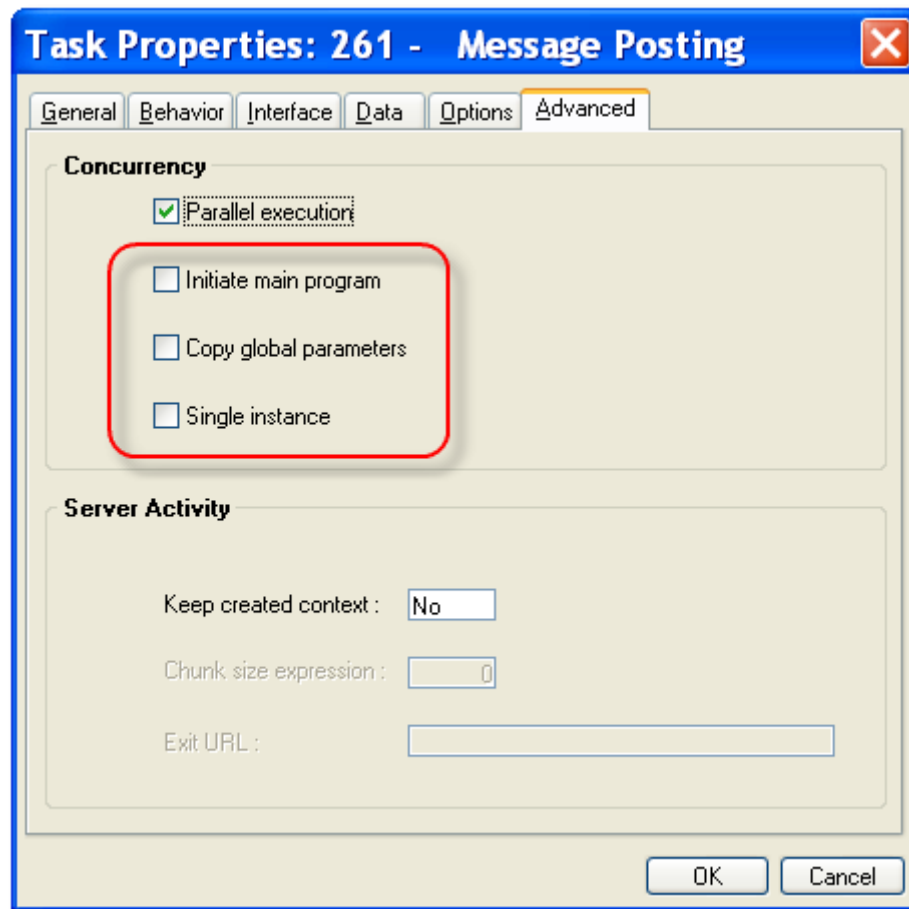
Now we reverse the process.

- First we use `SharedValGet ('ContextTbl')` to fetch the BLOB back from shared memory.
- Then we use `MTblSet (b.TableBlob, '4'DSOURCE, '', 1)` to unpack the BLOB into Data source #4, the memory table.

For more information on using the MTbl functions, check the eDeveloper Help files.

For more information about using SharedVal functions, see Chapter 25, "How do I Share Memory Tables Amongst Contexts?" on page 673.

## How do I Control the Initialization of a Parallel Task?



Once you check *Parallel execution* in the *Task Properties*, three other options become available for you to set. The first two of these you can use these to control the initialization of the parallel program.

### Initiate main program

If checked, the Main program will re-run before the parallel program runs. In other words, you get a fresh copy of the Main program, just as if you had re-started eDeveloper. Task Prefix runs, and any values that were stored in variables will not be there in the new context.

If not checked, the variables in the Main program are copied. It is as if a snapshot were taken of the Main program for the new context. From there on out, changes made to the new context are local to that context, and don't effect the original context's Main program at all.

### Copy global parameters

If checked, a copy of the existing global parameters will be made. The new context will get a snapshot of the global parameters. From then on out, changes made to the global parameters will be local to the new context.

If not checked, the new context gets a fresh copy of the global variables.

These would be the **SetParam** variables. The **SharedValSet** and **SharedValGet** global parameters are truly global between the contexts.

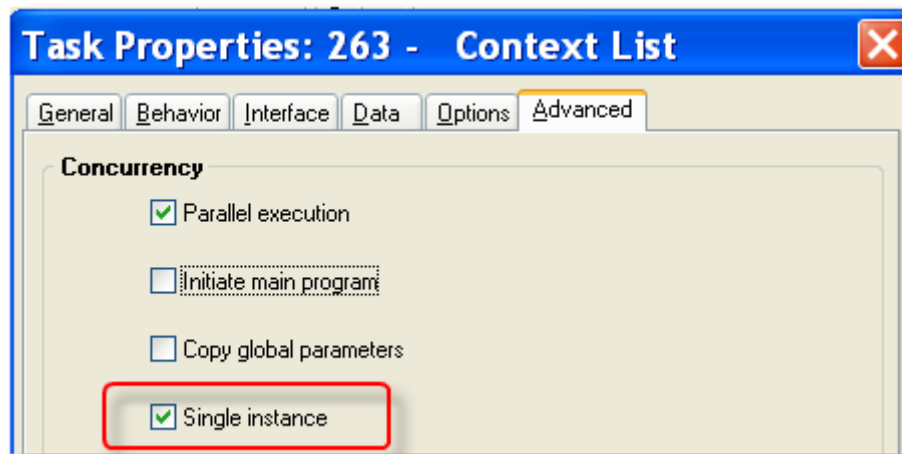
Otherwise you will get a fresh version of the global parameters.

## How do I Force a Single Instance of a Running Program?

There are two basic types of parallel tasks.

In the first type, you might open up one, two, or six different windows simultaneously. For instance, you might use these for different documents you are working on at the same time, or in our example, multiple chat windows. This is the default type of parallel program in eDeveloper.

In the second type, you want to access the same window every time you select it. This would be the case, for example, for a menu of open documents, or our list of open chat windows. This type of parallel program is called a *Single instance* program.



If you want the program to be a Single instance program, simply:

1. Go to *Task properties* (**Ctrl+P**).
2. Select the *Advanced* tab.
3. Check *Single instance*.

Now, when you first select this program from a menu or start it from another program, it will open in its own context. But if it is selected again, while it is already running, control will shift to the existing instance of the program rather than creating a new one.

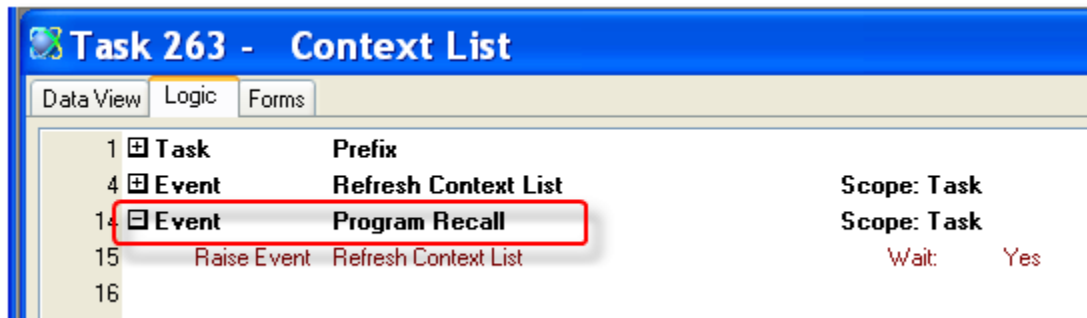


## How do I Identify a Repetitive Call to a Single Instance Program?

The first time a Single instance program is called, it creates a new context and starts. Task Prefix is executed.

The second time a Single instance program is called, the existing instance is used. Focus shifts, but Task Prefix is not executed.

Therefore, you need another way to trap the fact that this program was re-called.



Fortunately, there is an easy way to do this. There is an *Internal event* called Program Recall. This is only executed if the Single instance program is *re-entered* (the first time it is entered, you can use Task Prefix).



## Chapter 26: **Window Interface**

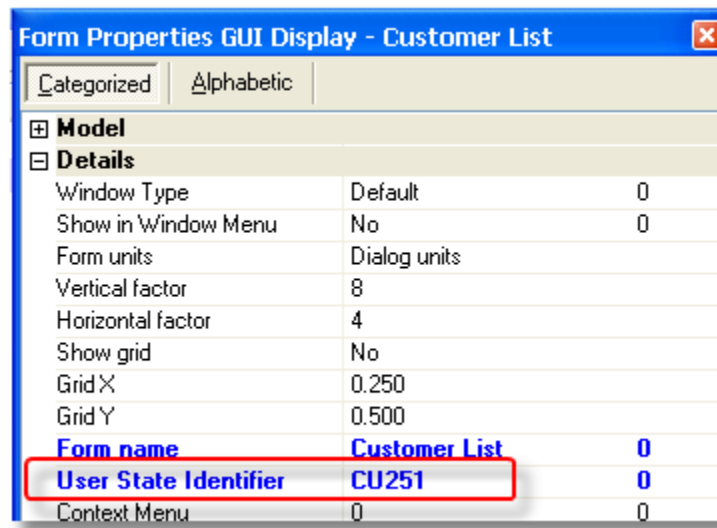
---

### How do I Keep the End User's Form Customization?

In eDeveloper working under Windows, there are a lot of things the user can do to a window to make it more usable. For instance, the user can resize the columns on a table, change the sort order, or even rearrange the columns. The user can also change the position and size of the window.

Very likely the user would like to have all these customizations saved between sessions. However, it would be a lot of programming work if you had to do this manually! Fortunately, eDeveloper will do it for you. Here's how.

## Using Form State Persistency

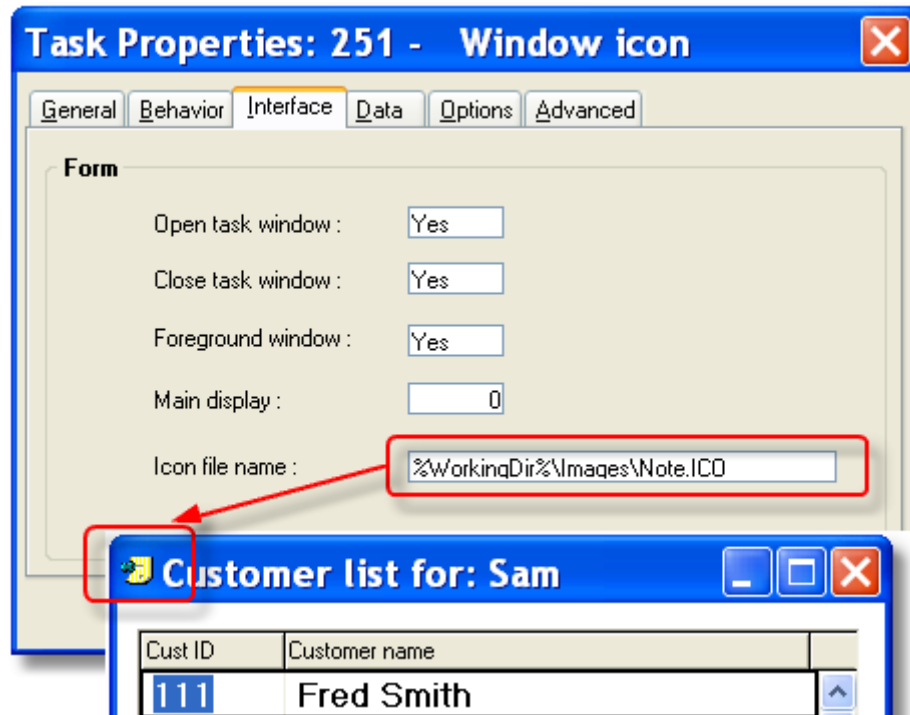


1. Go to the *Forms* tab of your task.
2. Type **Alt+Enter** to bring up the *Form Properties*.
3. Go to the *User State Identifier* property.
4. Type in some unique identifier. Here we use a program ID, which is just a number we assign to each program so we can refer to it easily, but any text string will work.

Now, customizations made by the user to this window will be remembered when the user re-enters the window.

**Note:** This feature isn't active when you are working in the Studio. That is, when you add a User State Identifier, then run your program with F7, the form customizations are not preserved.

## How do I Set the Icon for a Window?



There is a default icon file you can set for the entire application, in **Application Properties->Startup->Icon file** name. However, you can also set up individual icons for any window.

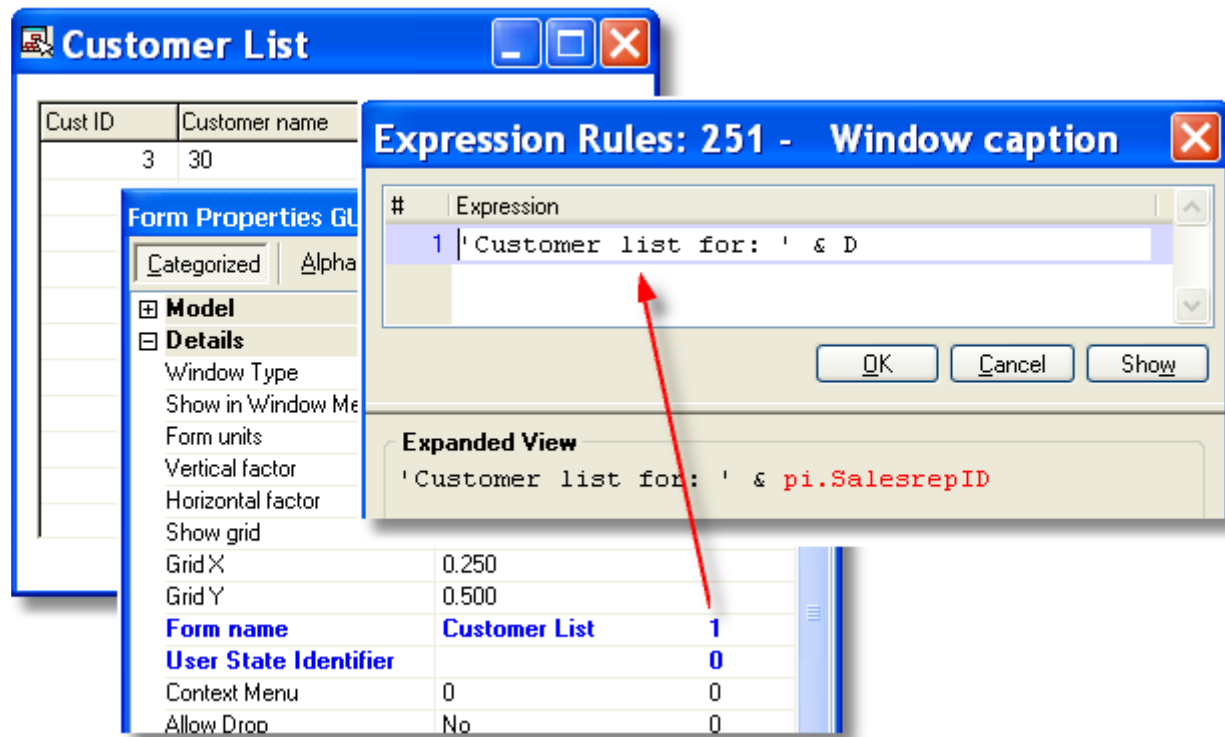
1. Go to the task for which you would like to set the icon.
2. Press **Ctrl+P**. This will bring up the Task Properties dialog.
3. Click on the *Interface* tab.
4. Zoom from the *Icon file name* field to select an icon file, or type in a relative path, or use a logical name to set the directory.

The icon will appear at runtime in the upper left hand part of the window.

## How do I Dynamically Set the Caption for a Window?

By default, the *Form name* property is inherited from the name of the Task. You can easily change it by typing over the text in **Form Properties->Form name**. However, you can also set the name dynamically, so it can change even while the task is running.

### Setting the Form name property to an expression



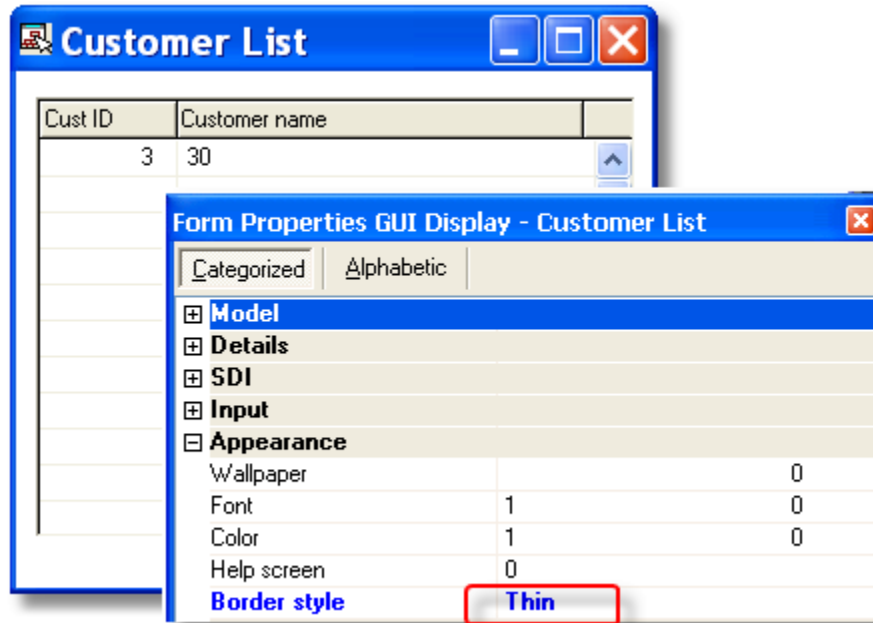
1. Go to the *Forms* tab of your task.
2. Type **Alt+Enter** to bring up the *Form Properties*.
3. Go to the *Form Name* property.
4. Zoom from the *Expression* column, to enter an expression that will evaluate to an alpha string.

In the Studio, the Form name will still appear as the text in the Form name property. However at runtime, the Expression will be evaluated and the results will be displayed in the title bar.

## How do I Prevent the User from Resizing a Window?

By default, the user can drag the edges of a window to resize the window. However, if you can, if you want, disallow this by setting the *Border style* to **Thin** or **None**.

### Preventing resizing



1. Go to the *Forms* tab of your task.
2. Type **Alt+Enter** to bring up the Form Properties.
3. Go to the Border style property.
4. Select Thin or None.

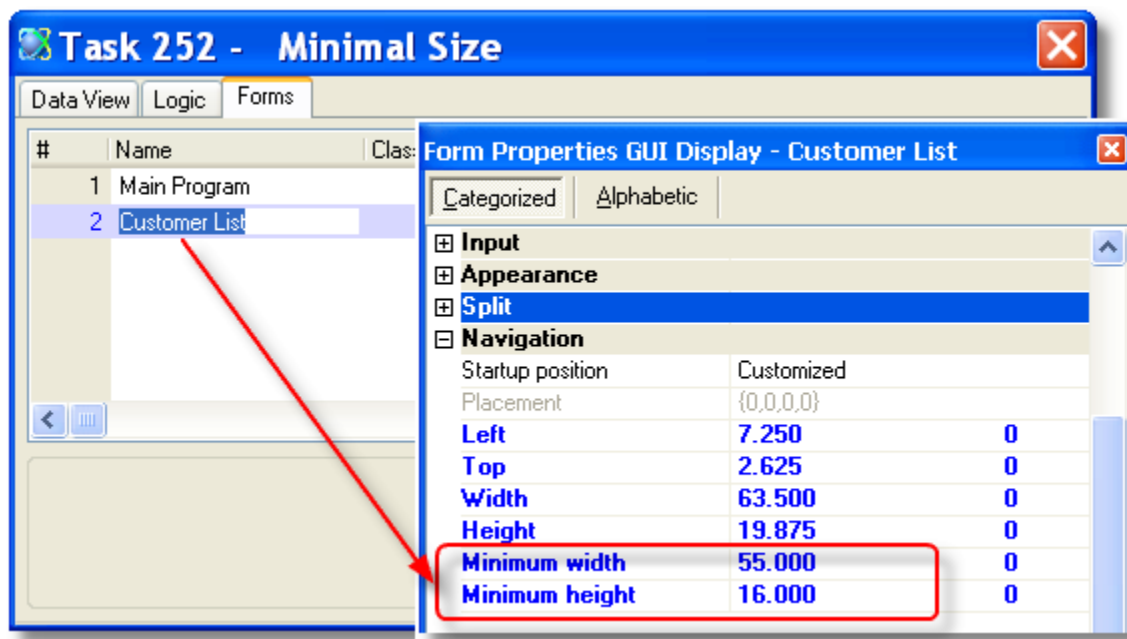
Now, the cursor will not change at the edge of the window, and the user will not be able to resize it.

## How do I Set a Minimal Size for a Window?

The Windows you create in eDeveloper will, by default, be resizable by the user. Depending on the placement settings, controls such as the tables will also resize as the screen resizes. However, it is then possible for the user to compact the form so much that it looks very odd.

To prevent this you can set a form Minimum size. The form will still expand to whatever size the user sets, but it will not become any smaller than the size you specify.

### Setting the minimal size



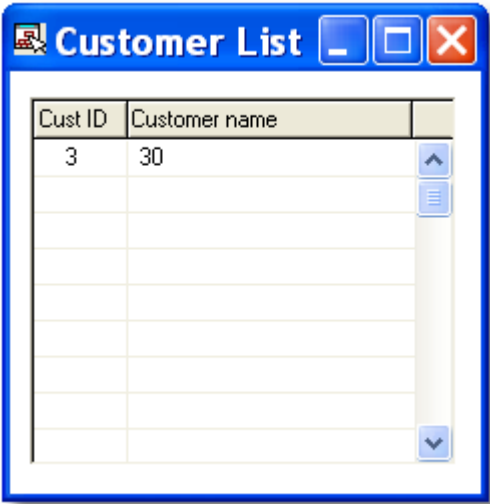
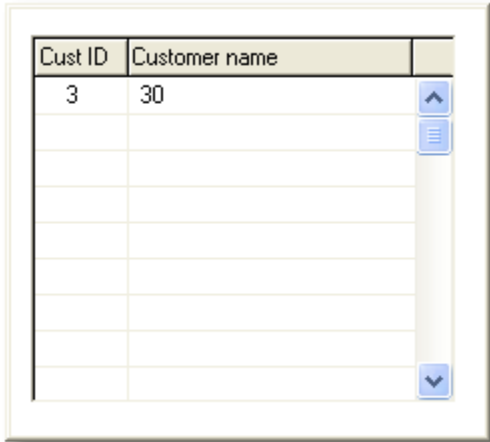
1. Go to the *Forms* tab of your task.
2. Type **Alt+Enter** to bring up the Form Properties.
3. Type in values for the *Minimum width* and *Minimum height* properties.
4. Alternatively, you can use Expressions to calculate these values at runtime, by zooming on the Expressions field to the right.

Now, in our example, the form will not contract below 55 dialog units wide and 16 tall.

**Hint:** An easy way to find the numbers for the minimal size settings is to manually size the window to the size you want, then copy the numbers from the *Width* and *Height* properties of the form.



How do I Remove the Window Title?

With title bar	With no title bar
	
Form Properties->Title Bar->Yes	Form Properties->Title Bar->No

By default, all windows have the standard Windows title bar at the top. You can turn this off by setting the *Title Bar* property of the Form to **No**.

If you want to just blank out the text but leave the title bar, set the *Form Name* property to blank.

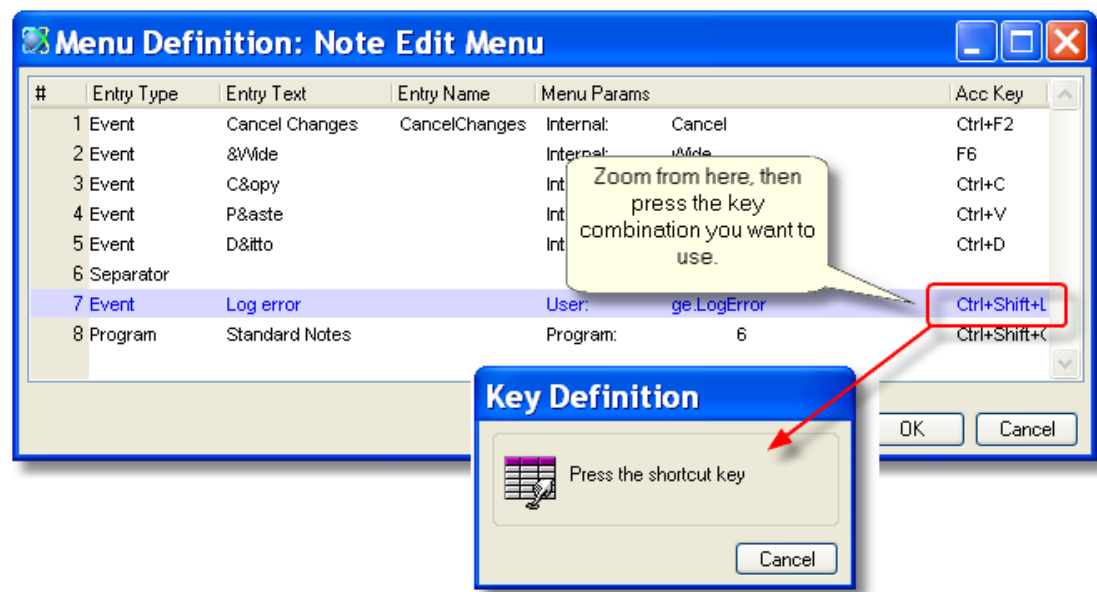


# Chapter 27: Menuing

## How do I Set an Accelerator for a Menu Entry?

Many users like to use Accelerator keys, otherwise known as “Hot keys”, to quickly access functions they use a lot. When you are creating an application, you can set your own Accelerator keys as needed, for any menu entry.

### Setting Accelerators for non-Internal events



1. Position the cursor on the *Acc Key* column.

2. **Zoom (F5 or double click).** The *Key Definition* window will appear.
3. Press the key combination you want. In our example, we pressed **Ctrl+Shift+L**. You can use any key combination you want, including **Alt** keys and **Enter**.
4. As soon as you press your key combination, the *Key Definition* window will close, and your key combination will appear in the *Acc Key* column.

Now, when you run your project, pressing the Accelerator key will execute the item on the menu.

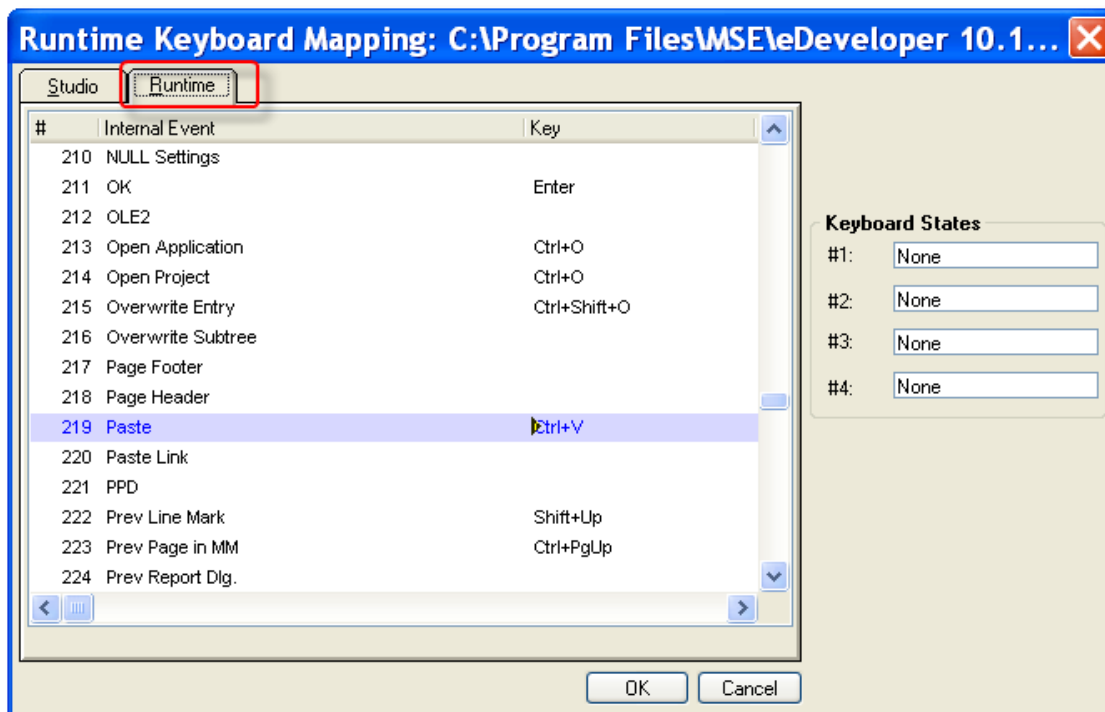
## Setting Accelerators for Internal Events

If you try to set an *Accelerator* for an *Internal event*, you will get a message:

**The shortcut of an internal event should be set in the keyboard mapping file.**

The Accelerator for an Internal event is set automatically, based on the keyboard mapping (**Options->Settings->Keyboard Mapping**).

However, when you change the keyboard mapping, you need to be sure to change the keyboard on the *Runtime* tab, as shown below. When you change the keyboard mapping, the changes will be automatically reflected in your menu entry.



## How do I Change the Menu of a Running program?

You can design your menu structure within eDeveloper, in the Menu Repository. However, you can also modify the menu entries at runtime, to make the menus change according to what program is running, or to change according to what is going on.

### Changing overhead menu entries

You can also choose which overhead menu entries are functional. These functions do not change the items on the menu, but they do cause the menu items to be disabled or to disappear.

- **MnuCheck()** to check and uncheck a menu item. This causes a checkmark to appear, or not.
- **MnuEnable()** to enable and disable a menu item. This causes the menu item to be greyed out.
- **MnuShow()** to make a menu item appear or disappear.

These affect items that are on the menus. They are explained more fully in Chapter 27, “How do I Hide/Reveal a Menu Entry?” on page 695.

### Adding entire overhead menu branches

The menu functions allow you to add and remove menu items. These functions don’t just change the menu entries; they can be used to build menus on the fly, adding complete menu branches within an existing menu structure. This is a very powerful feature, and gives you a lot of flexibility. There are three menu functions that are used:

- **MnuAdd()** to add a menu item.
- **MnuRemove()** to delete a menu item.
- **MnuReset()** to reset the menu to the default settings.

**Note:** If you are changing menus only for security reasons, it is easier to use the Authorize option for that menu item. If the user is not authorized to use the menu item, it will automatically disappear without additional programming.

#### MnuAdd()

**MnuAdd()** allows you to add menu entries on the fly, anywhere within the menu system. The syntax is:

**MnuAdd(MenuNumber, MenuPath)** where

**MenuNumber** is the menu you want to add. This is the position of the menu in the menu repository. You should use the MENU literal. In our example we will use ' 2 ' MENU.

**MenuPath** is a path of *Entry Names* of the menu item you want to insert this one after.

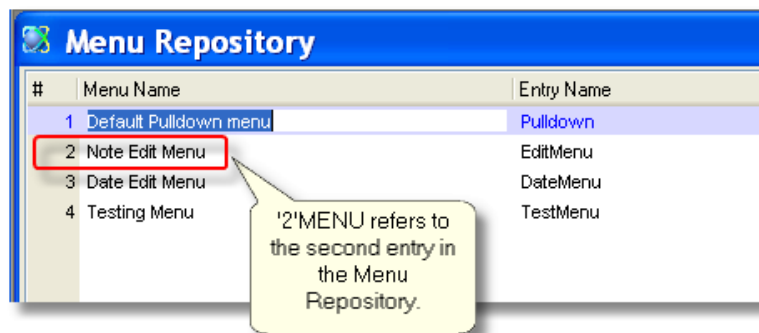
Let's take a look at an example. Our overhead menu looks like this:

#	Entry Type	Entry Text	Entry Name	Menu Params
1	Menu	&File		SubMenu: 6
2	Menu	&Edit		SubMenu: 18
3	Menu	&Options		SubMenu: 21
4	Menu	&Utilities	UtilityMenu	SubMenu: 2
5	Menu	&Window		
6	Menu	&Help		

#	Entry Type	Entry Text	Entry Name	Menu Params
1	Menu	Setup	SetupMenu	SubMenu:
2	Menu	Browser	BrowserMenu	SubMenu:

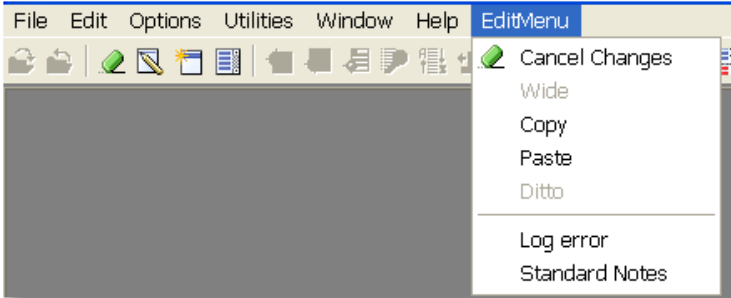
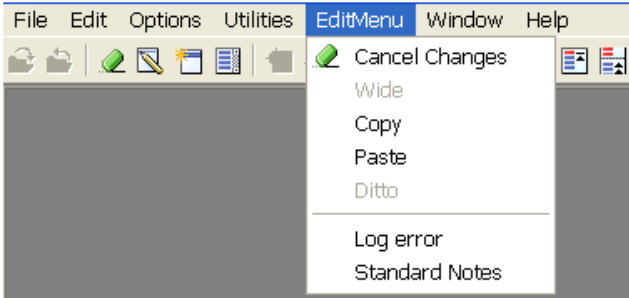
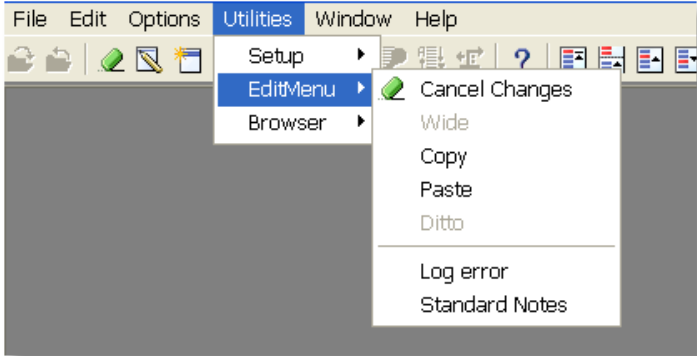
Our Menu repository looks like this. Menu 2 is the one we are going to add to the pulldown menu:



#	Menu Name	Entry Name
1	Default Pulldown menu	Pulldown
2	Note Edit Menu	EditMenu
3	Date Edit Menu	DateMenu
4	Testing Menu	TestMenu

'2'MENU refers to the second entry in the Menu Repository.

Now let's see what three different versions of `MnuAdd()` will give us:

<div><div><code>MnuAdd('2'MENU, '')</code></div><div>This adds the #2 menu, the EditMenu, in the last position of the default pulldown menu.</div></div>	
<div><div><code>MnuAdd('2'MENU, 'UtilityMenu')</code></div><div>This locates the menu after the Utility menu at the default pulldown level.</div></div>	
<div><div><code>MnuAdd('2'MENU, 'UtilityMenu\SetupMenu')</code></div><div>This locates the menu on the Utility menu, after the Setup menu.</div></div>	

MnuRemove()

`MnuRemove(MenuNumber, MenuPath)` where

`MenuNumber` is the menu you want to add. This is the position of the menu in the menu repository. You should use the MENU literal. In our example we used '2'MENU.

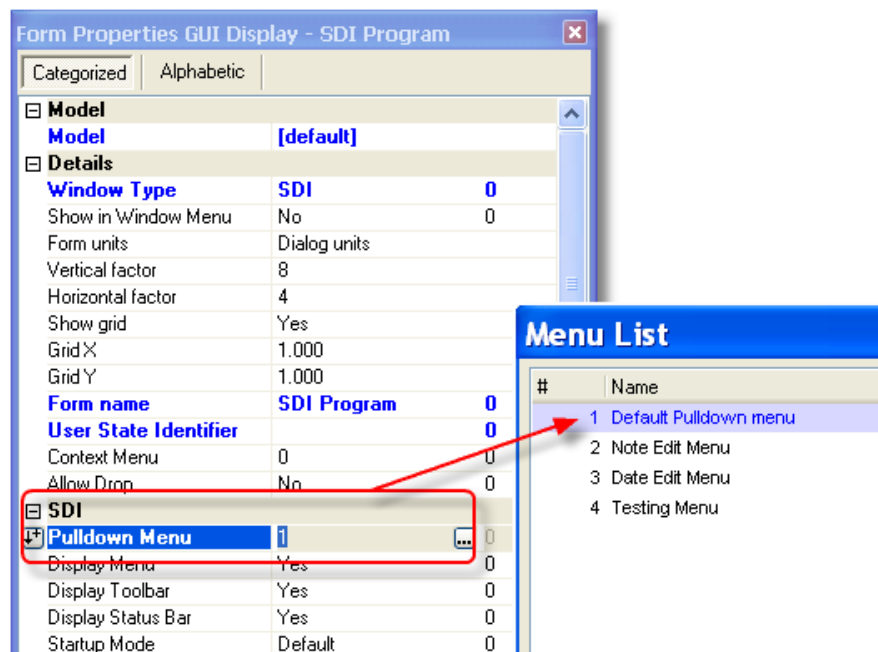
**MenuPath** is a path of *Entry Names* of the menu item you want to insert this one after.

**MnuRemove()** works like **MnuAdd()**, but in reverse. It removes the menu that was added. **MnuAdd('2'MENU, 'UtilityMenu\SetupMenu')** or **MnuAdd('2'MENU)** would both remove the menu we added in our example.

### MnuReset()

**MnuReset()** resets the menus back to where they were before the **MnuAdd()** functions. For our example, we could use **MnuReset()** instead of **MnuRemove()**. But, **MnuAdd()** doesn't remove just one specific menu that was added; it removes all of them.

## Changing the menu of an SDI program

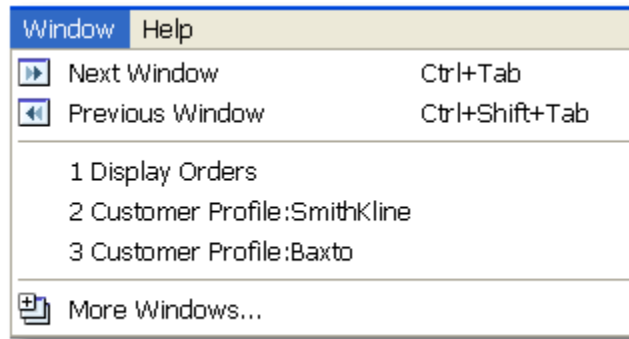


An SDI program has its own pulldown menu. You specify the Pulldown menu property on the SDI form. You can use an expression, but the expression is only evaluated once, when the program starts, so changing the expression won't have any effect on the menu. Instead, use the Mnu functions on the SDI menus if you need them to change after the program starts.



## How do I Enable Keyboard Window Switching?

You have the option in eDeveloper of allowing your user to keep several windows open at one time. To make it easy for the user to jump between these open windows, you can use the Windows menu facility.



You can allow a user to jump between several open windows by using the *Window menu* facility. In this example we have three windows open at the same time. One displays orders. The other two windows are the same Customer Profile program, called for two different customers.

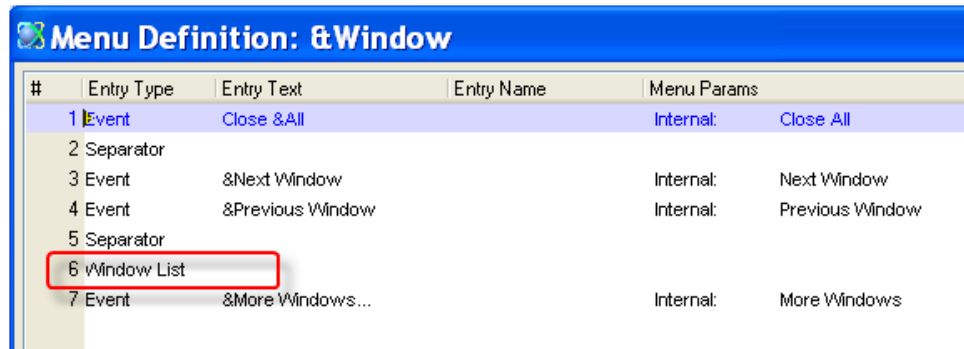
*Navigating the windows:* The user can use the *Next Window* (**Ctrl+Tab**) or *Previous Window* (**Ctrl+Shift+Tab**) to move between the open windows. Or the user can choose the open windows from the open window list that is on this menu. Typing the first letter(s) of the menu item will move the cursor to that item.

*More Windows:* If there were more than 9 open windows, the user could choose *More Windows* to show the entire list.

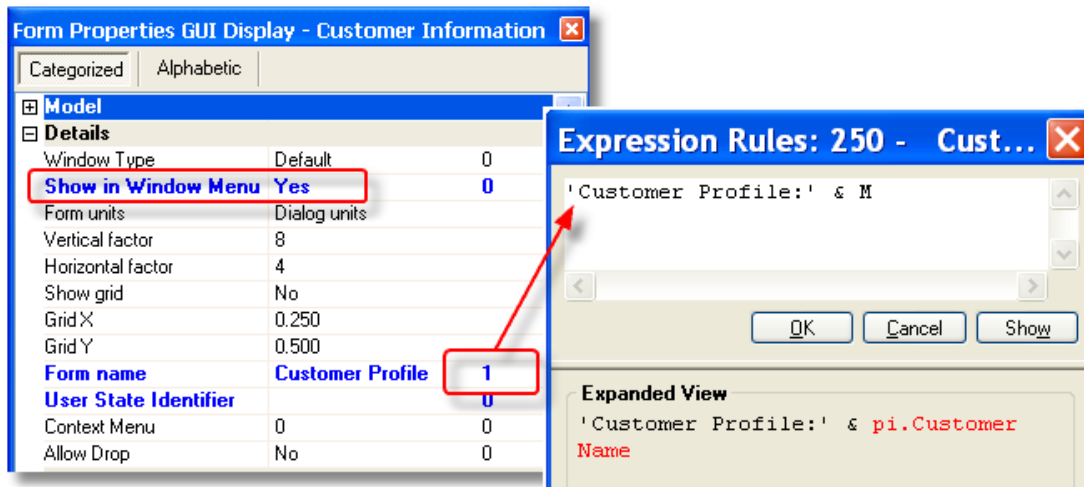
*Order of the list:* The list can either be ordered according to the most recently used window, or by the order in which they were created. You can control this for the application in *Application properties->Startup->Window Sort by*. This affects both the menu list, and using **Ctrl+Tab** to move between windows.

Now, let's see how to use this facility.

## Using the Windows Menu



1. First, you need to have the window menu options where you want them. By default, there is an overhead menu option called *Window*, as shown above. The list of open windows is displayed in an entry type called *Window List*. You can put the entry elsewhere, if you like.



2. Next, you need to enable the Window menu for the programs you want to show on the menu. You do this by setting **Form Properties->Show in Window Menu** to **Yes**. Note that you can only do this for *Window Types* of Default, SDI, Floating, Tool, and Fit to MDI.
3. It's a good idea to make sure each window has a unique form name, because the form name is what displays on the window list. In our example, we used an expression in the Form name, using a parameter, so that each window would have a unique name per customer.

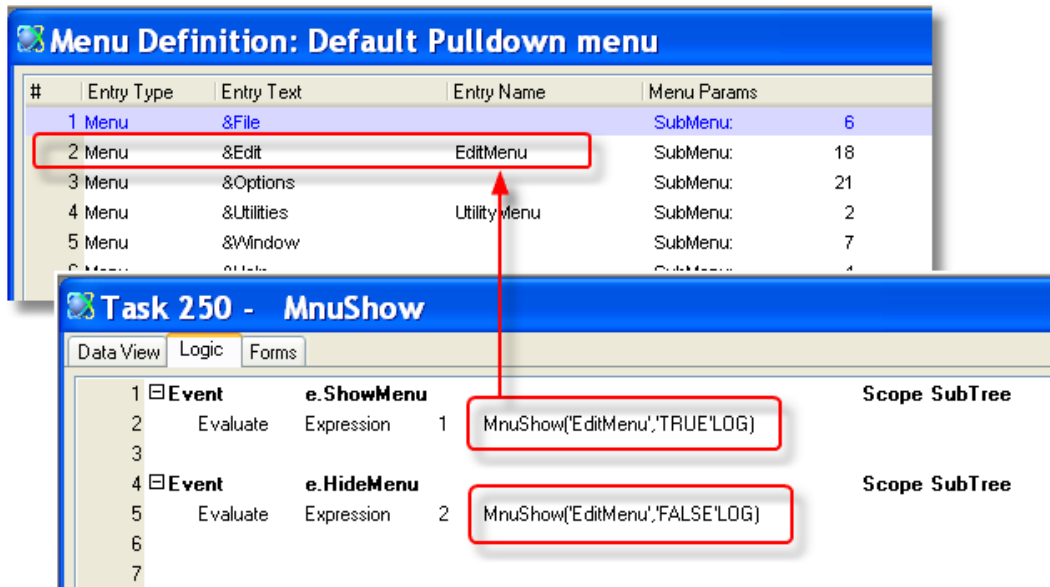
Now, when the programs run, they will appear on the *Window menu*.

## How do I Hide/Reveal a Menu Entry?

You can hide any entry that is on a menu at runtime by using the **MnuShow()** function. Alternatively, you can use **MnuEnable()** or **MnuCheck()** to keep a menu entry from being functional while still being visible.

**Note:** Menu entries also disappear if a Right is indicated in the *Menu properties*, and the user doesn't have the specified *Right*.

### Using MnuShow()



The syntax of **MnuShow()** is:


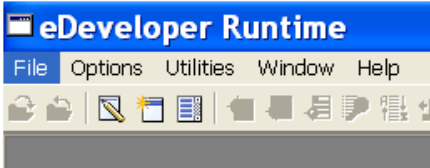
**MnuShow(MenuName, True/False)** where

**MenuName** is the *Entry Name* of the menu you want to add. In our example, we added an *Entry name* of "EditMenu" to the default edit menu.

**True/False** is a 'TRUE'LOG to show a menu item, 'FALSE'LOG to hide it. In our example, we have two events set to push buttons, one to show and one to hide the menu.

**Note:** The menu EntryName is evaluated at runtime, so it doesn't have to be hard-coded.

Now let's see what happens.

	
MnuShow('EditMenu','TRUE'LOG)	MnuShow('EditMenu','FALSE'LOG)

How do I Remove a Menu Bar?

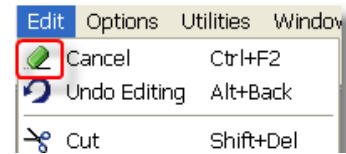
You can remove the various menu bars from either an MDI or SDI application. Here is how to do it.

Which Menu	MDI	SDI
<i>Pulldown Menu</i>	Set <b>Application Properties-&gt;StartUp-&gt;System Pulldown menu</b> to zero.	Set <b>Form properties-&gt;SDI -&gt;Pull-down Menu</b> to zero. Or, set <b>Form properties-&gt;SDI -&gt;Display Menu</b> to No.
<i>Toolbar</i>	Set <b>Options-&gt;Environment-&gt;Display Toolbar</b> to No.	Set <b>Form properties-&gt;SDI -&gt;Display Toolbar</b> to No.
<i>Status bar</i>	If you need to remove the status bar, make it an SDI program.	Set <b>Form properties-&gt;SDI -&gt;Display Status Bar</b> to No.
<i>Title bar</i>	If you need to remove the title bar, make it an SDI program.	Set <b>Form properties-&gt;Input-&gt;Title Bar</b> to a No.

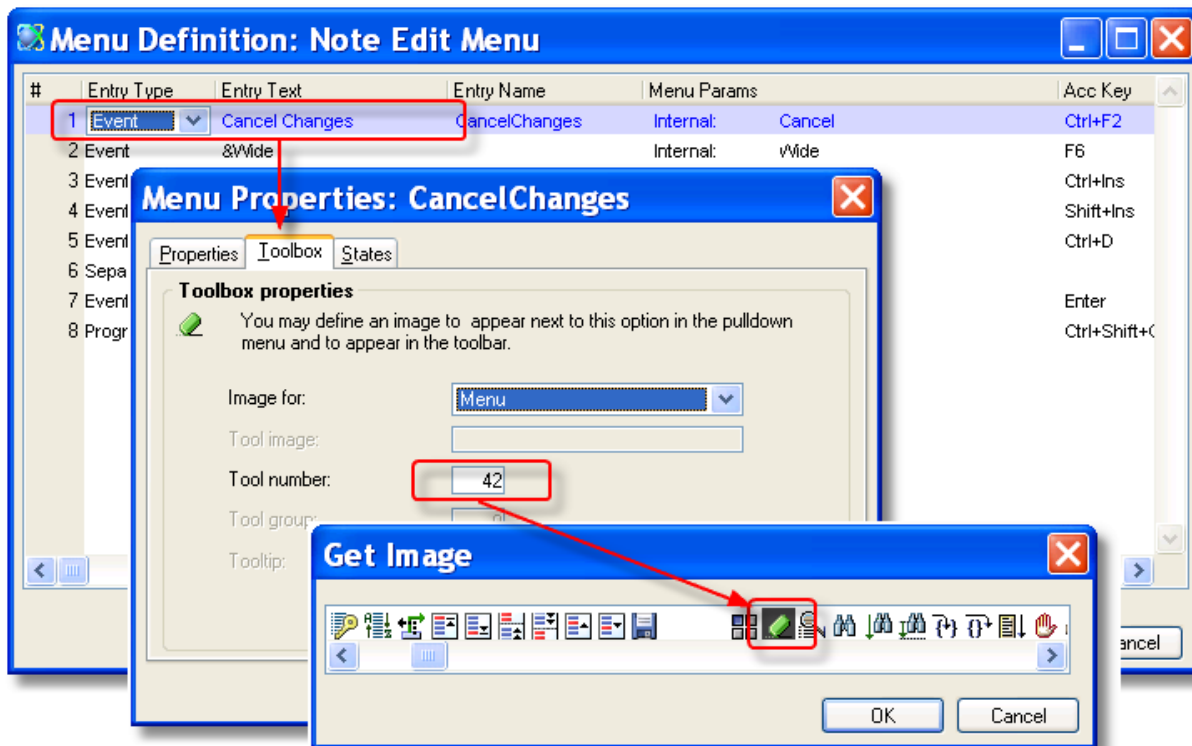
## How do I Add an Icon to a Menu Entry?

You can add an icon to any menu entry, whether the menu is being accessed from the pulldown or the context menu structure. The same icon can also be used on the toolbar (see Chapter 27, “How do I Add an Icon to the Toolbar?” on page 700).

You can choose one of the internal eDeveloper icons, or, you can specify a bitmap file to use a customized icon.



### Specifying a menu icon



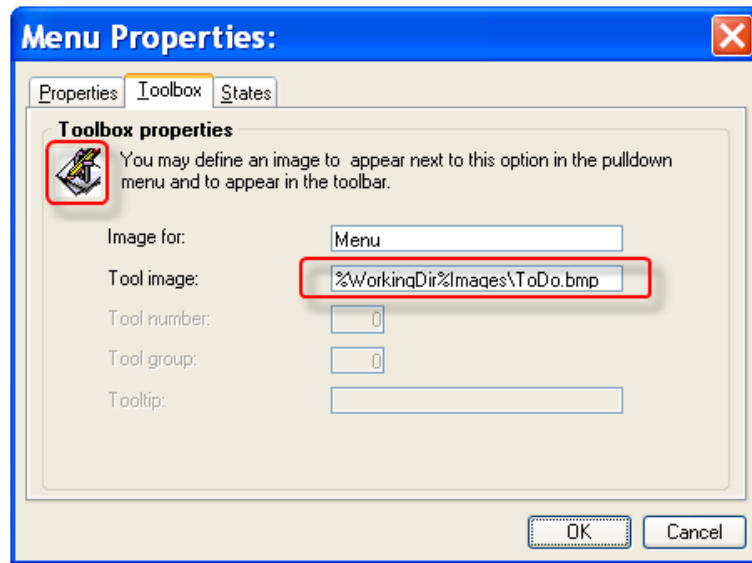
You can add an icon to a menu entry by specifying it in the *Menu Properties*.

1. Go to the *Menu Repository* (**Shift+F6**, or **Project->Menus**).
2. Go to the menu item that needs the icon. Press **Alt+Enter** to access the *Menu properties*.
3. Click on the *Toolbox* tab.
4. Set *Image for:* to *Menu* (if you want the icon only on the menu) or *Both* (if you also want it on the Toolbar).
5. To choose an icon from the internal eDeveloper icons, zoom from the Tool number field. Find the icon you want, and press Enter. The icon number will be brought back into the Tool number field, and a picture of the icon you chose will appear at the top of the Toolbox tab.

Now, the icon will appear on the menu entry at runtime.

Alternatively, you can choose a bitmap file for a customized icon, as shown below.

### Choosing your own icon



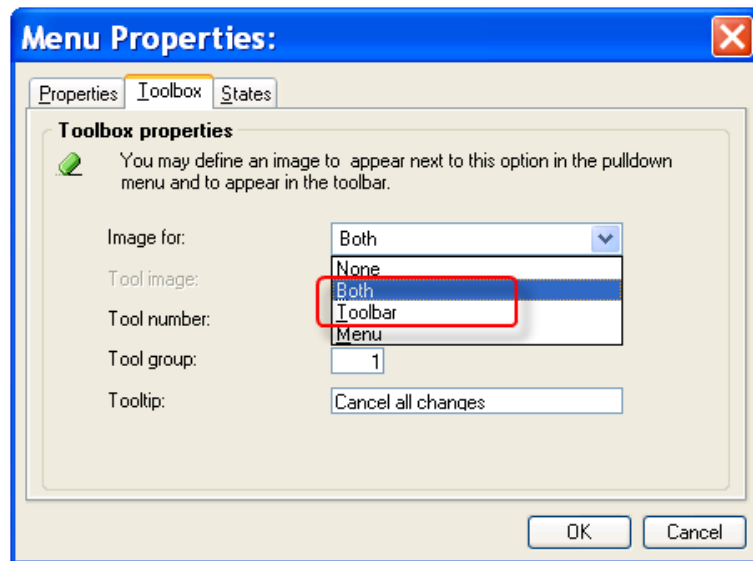
You can also specify your own icon. The icon should be 16x16 pixel bmp.

1. Proceed as above, but instead of specifying a *Tool number*, specify a *Tool image*. You can zoom from the Tool image field to select the file name, but it is best to use a logical name, as shown in our example.
2. The icon will appear at the top of the *Toolbox* tab.

## How do I Add an Icon to the Toolbar?



You can add a toolbar icon for any menu entry.



To do this, follow the instructions for setting an icon to a menu entry (Chapter 27, “How do I Add an Icon to a Menu Entry?” on page 698), but in the *Image for:* field, specify *Both* or *Toolbar*.



## Chapter 28: Unicode

---

### How do I Enable Support for Multi-Lingual Data?

eDeveloper has excellent support for multi-lingual data. Internally, eDeveloper uses Unicode format when possible, and it has tools to allow you to translate between Unicode and other representations.

However, in order to work with Unicode on your computer, you need to be sure your computer is set up to support Unicode correctly. In the United States, at least, current computers do not install the full Unicode font set by default, so you will not be able to view Unicode characters.

Also, Unicode data needs to be displayed in a field with a Unicode attribute type, that is using the Unicode font.

In this section, we will deal with these issues.

### The Unicode Font

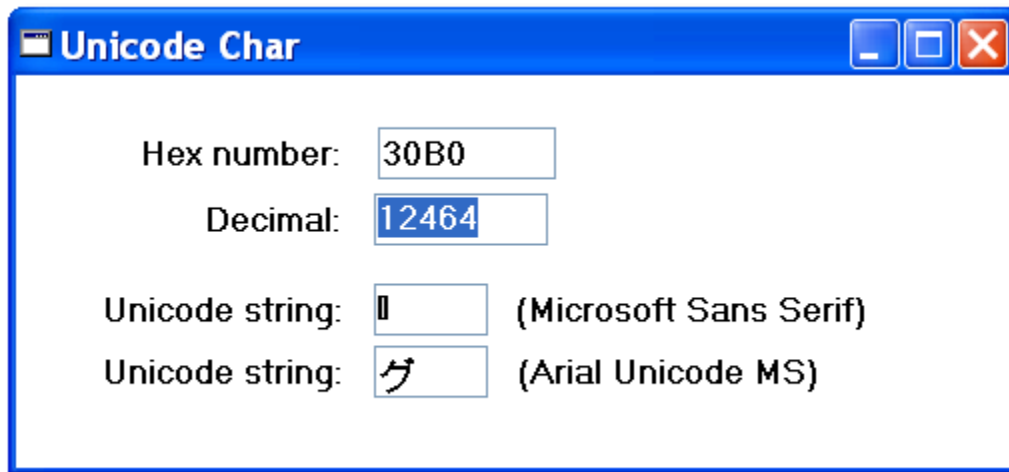
Back in the days of Ascii, all characters could fit within one byte. This was mainly because the first programmers mostly spoke English, which has a mere 26 characters in the alphabet, leaving plenty of room for special characters. However, the “special characters” were mapped out differently depending on the usage, so, Ascii code 210 could mean a funny looking E, funny looking O, or graphic line character, depending on the “code page” being used.

In order to get around this issue, Unicode was invented. Unicode essentially combines all the world’s alphabets into one big code page with many thousands of characters. You can find the mapping of these characters online, at [www.unicode.org](http://www.unicode.org). There you will find that the original Ascii fonts, 0 thru 128, are mapped as they always have been, but other more esoteric characters have been added. For instance, hex code 30B0 is the Japanese character



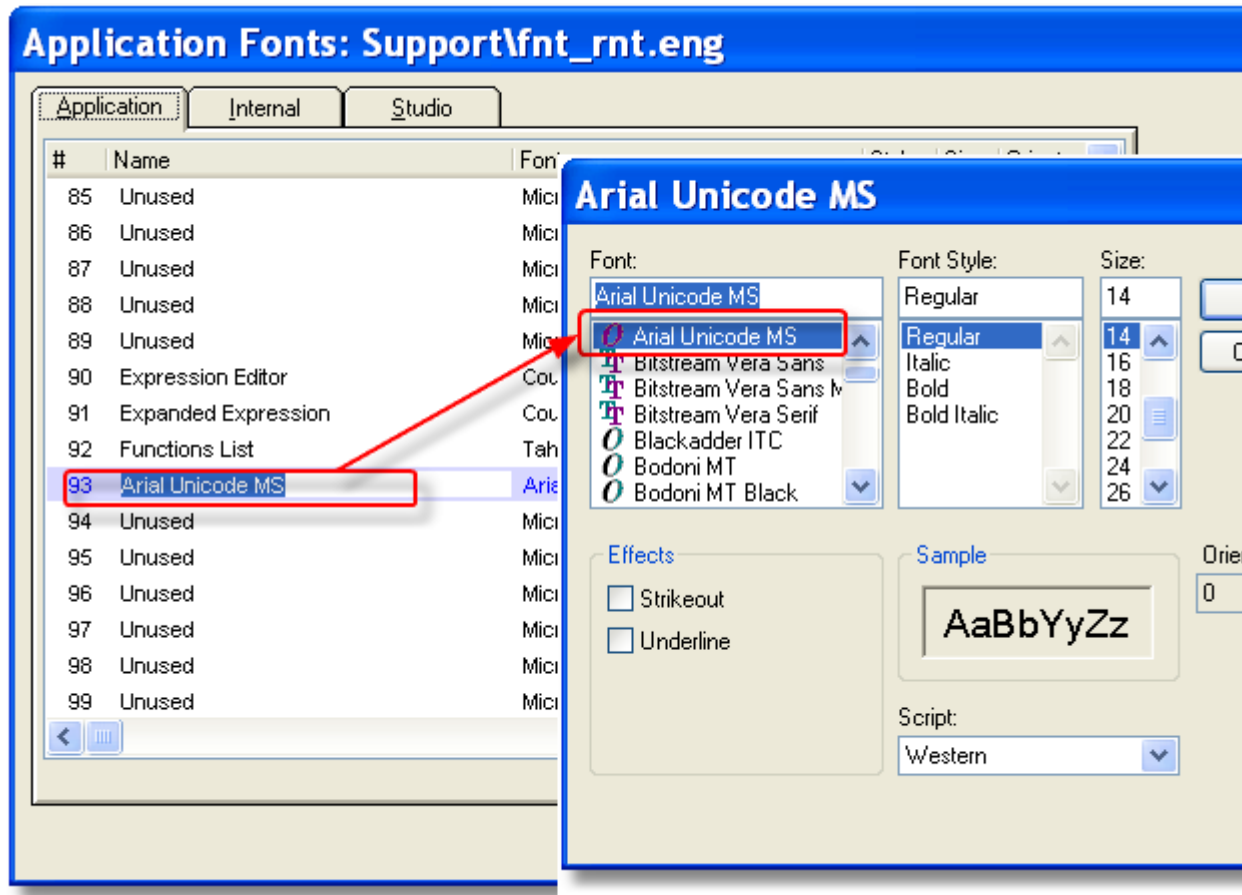
on the right. Other hex codes exist for other languages, both existing, historical, and fictional (Klingon characters are in there too).

This is very good in theory, but it makes for a large Windows font. As a result, many Windows systems do not have a Unicode font installed for you to view these Unicode characters. One good Unicode font comes with Microsoft Office, Arial Unicode MS, but it is not installed by default when you install Microsoft Office. Follow the directions on <http://office.microsoft.com/en-ca/assistance/HP052558401033.aspx> to install it.



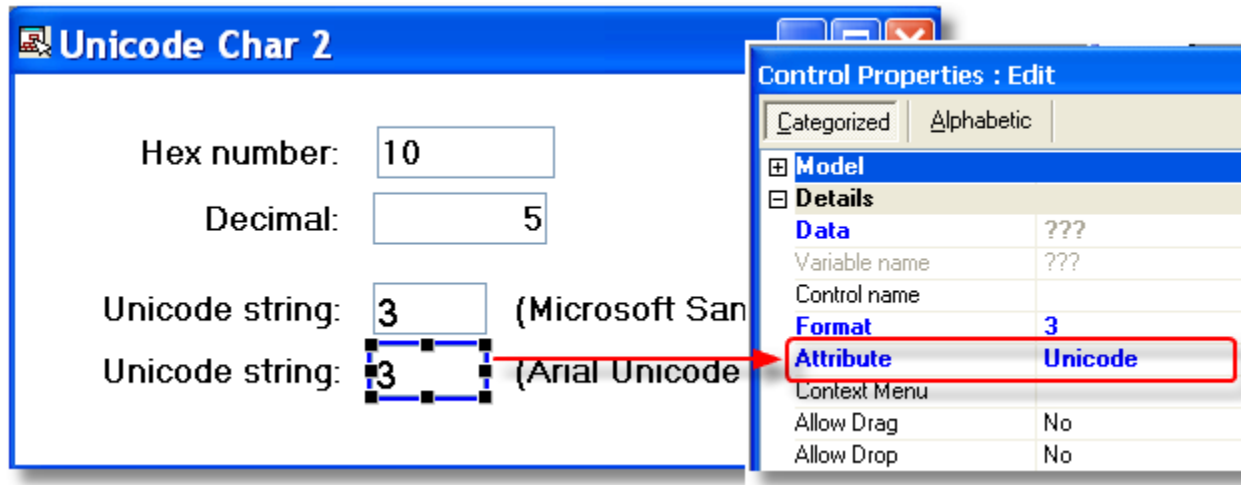
If you are displaying characters in one particular language, however, you may want a specialized Unicode font for that language. In our sample above, we display the Unicode for the Japanese character 30B0, but it

is not as pretty as the picture on the code page. Getting a specialized Japanese font would likely work better.



Once you install the desired Unicode font on your computer, you need to point to it in eDeveloper. Select that font in **Options->Settings->Fonts**, and use it whenever you need a Unicode font. In our example, font 93 is Unicode.

## The Unicode Attribute

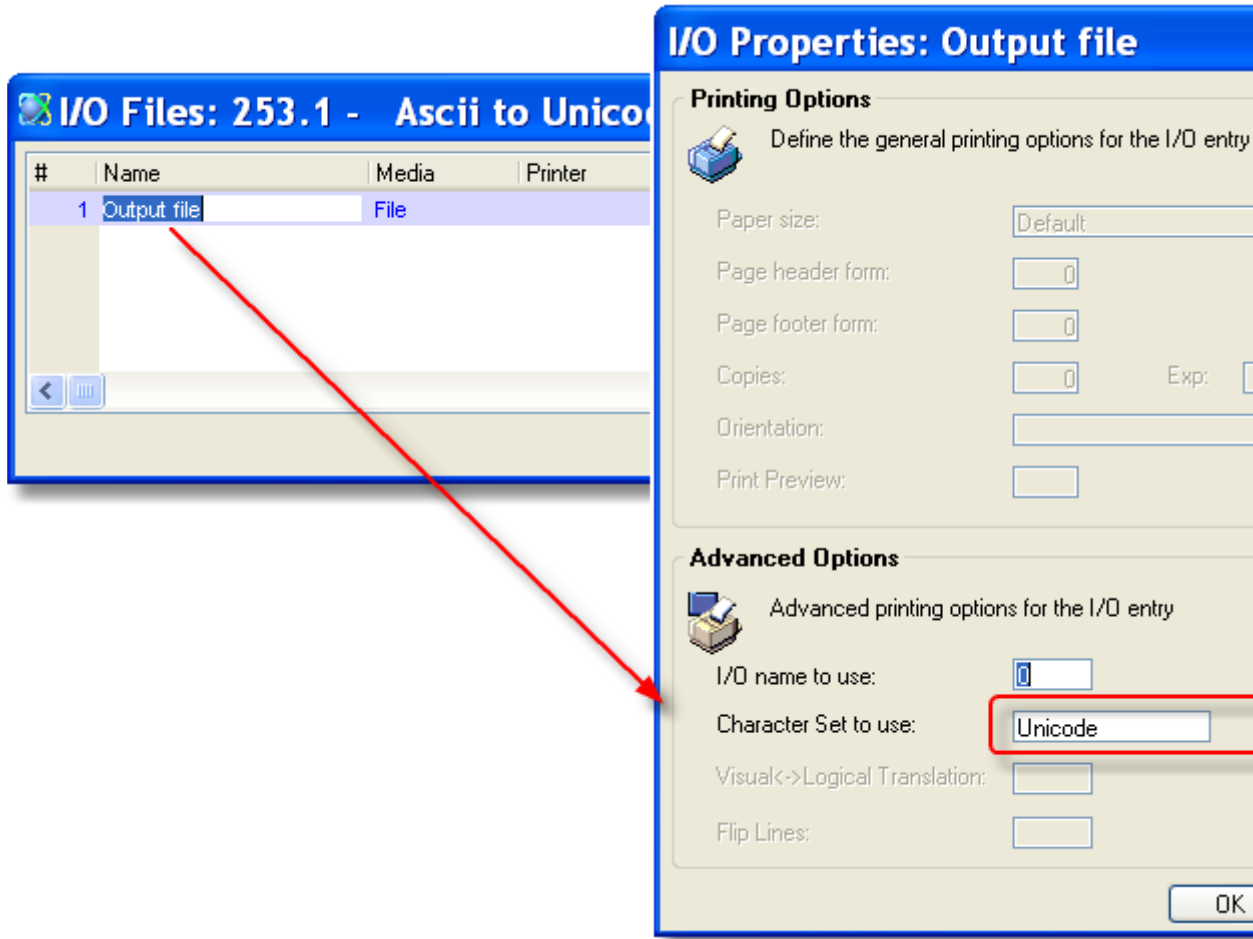


Also, you need to be sure the Unicode characters are in a field with a Unicode attribute. If you move the Unicode data to an Alpha, Memo, or RTF field, they will not display properly.

## How do I Create or Read a Unicode text file?

You can write and read Unicode text files as you would any other text file in eDeveloper. You need to do two things:

- Store the Unicode data in fields with the Unicode attribute (use **UnicodeFromANSI** () if you need to convert the code from ANSI, or **UnicodeFromANSI** () to convert it into ANSI).
- Use the Unicode attribute in the I/O File properties, as shown below.



The Unicode will be formatted automatically.



So, if M is our string with all the Ascii codes, and we use **UnicodeFromANSI** (M, 0), we get:

Unicode Default Codepage

```
#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[^_`abc
defghijklmnopqrstuvwxyz{|}~□€□„f„...†‡~%Š<œ□Ž□□‘’•—™š>œ□žŸıø£¤¥¦
§¨©ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãä
åæçèéêëìíîïðñòóôõö÷øùúûüýþÿ
```

But if we use **UnicodeFromANSI** (M, 437) we get:

Unicode Codepage 437

```
#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[^_`abc
defghijklmnopqrstuvwxyz{|}~□ÇüéâäåàçêëèìíîïÿÄÅÉæÆôöòûüÿÖÜø£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãä
åæçèéêëìíîïðñòóôõö÷øùúûüýþÿ
```

## Converting from Unicode back to ANSI

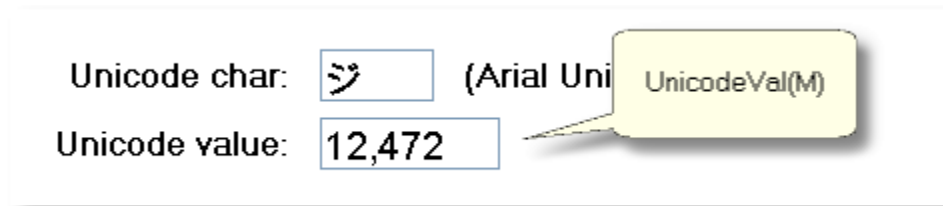
You can convert from Unicode back you ANSI by using the function:

**UnicodeToANSI** (String, CodePage) where:

- **String** is the Unicode alpha string to convert
- **CodePage** is the code page that will be used to interpret the ANSI.

## How do I Find the Unicode Value of a Character?

You can convert a Unicode character into a decimal value by using the **UnicodeVal()** function. In this example, we convert a Katakana character into its underlying decimal value, 12,472.



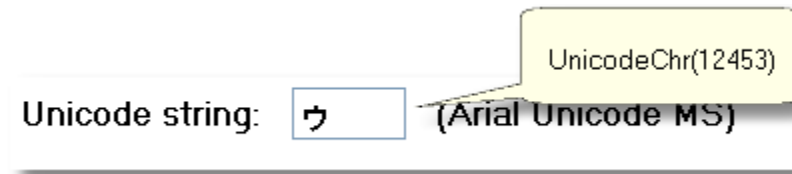
**UnicodeVal(UnicodeCharacter)**

Where **UnicodeCharacter** is character with a Unicode attribute. This returns a decimal number representing the Unicode value.



## How do I Convert the Unicode Value to a Unicode Character?

You can convert a number into a Unicode character by using the `UnicodeChr()` function.



### `UnicodeChr(Number)`

Where *Number* is a decimal number representing the Unicode value. This returns a Unicode character, which will display as the appropriate character in a Unicode field.

**Note:** The number here needs to be decimal, but the Unicode code pages are typically printed with hex codes. To translate these, you can use the `HVal()` function. For instance, `UnicodeChr(HVal(3085))` will give the same result as the example above: `UnicodeChr(12453)`.



# Chapter 29: Application Debugging

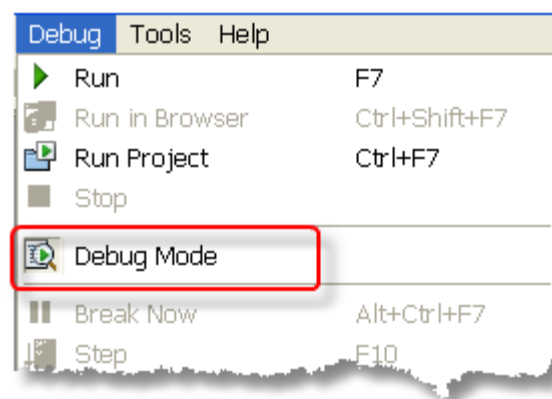
## How do I Debug my Application Using the Debugger?

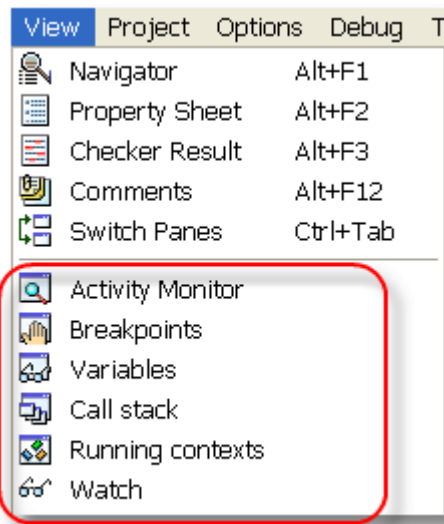
eDeveloper has a powerful debugging tool built in. You can use it to view exactly what the engine is doing as it executes, and to view the call stacks, database access, see what is going on as the program executes, view all the contexts, set watches on variables, view all the variables in memory, and even change the variables as the program is running.

Here we will review how to use the debugger.

### 1. Setting debug mode on

The first thing you need to do is to make sure you have *Debug Mode* switched on. Do this by selecting **Debug->Debug Mode**, or clicking on the icon on the Toolbar. It will show as depressed when it is selected.





## 2. Opening up your monitor windows

When the debugger runs, it will show data in a series of windows. These windows can be merged together or viewed separately, just like the other panes in the Studio. You can choose to have all of them open, or none of them.

Open up the monitors you want by clicking on the desired item in the View pulldown menu. You can open up the monitors after you have started the debugging process also.

## 3. Setting your breakpoints and watches

Now, you need to tell the debugger where to pause. There are two ways to do this:

- Setting a breakpoint, which can be on an operation or a change of a variable
- Using *Break Now*

Each of these are useful in different ways. For a full discussion of how to use these, see Chapter 29, “How do I Set Breakpoints in the Application?” on page 715.

It is important to set the breakpoint or use *Break Now*, because the variables, call stack, and contexts become visible only during a breakpoint. However, you can also set them as the program is running.

## 4. Start running your program or project


Now, you need to start running your program or project. If you are just debugging one program, position the cursor on that program and press **F7** (**Debug->Run**). You don't need to exit the task you are working on to run it; eDeveloper will save your changes before running the program.

If you want to test the entire project, press **Ctrl+F7** (**Debug->Run Project**).

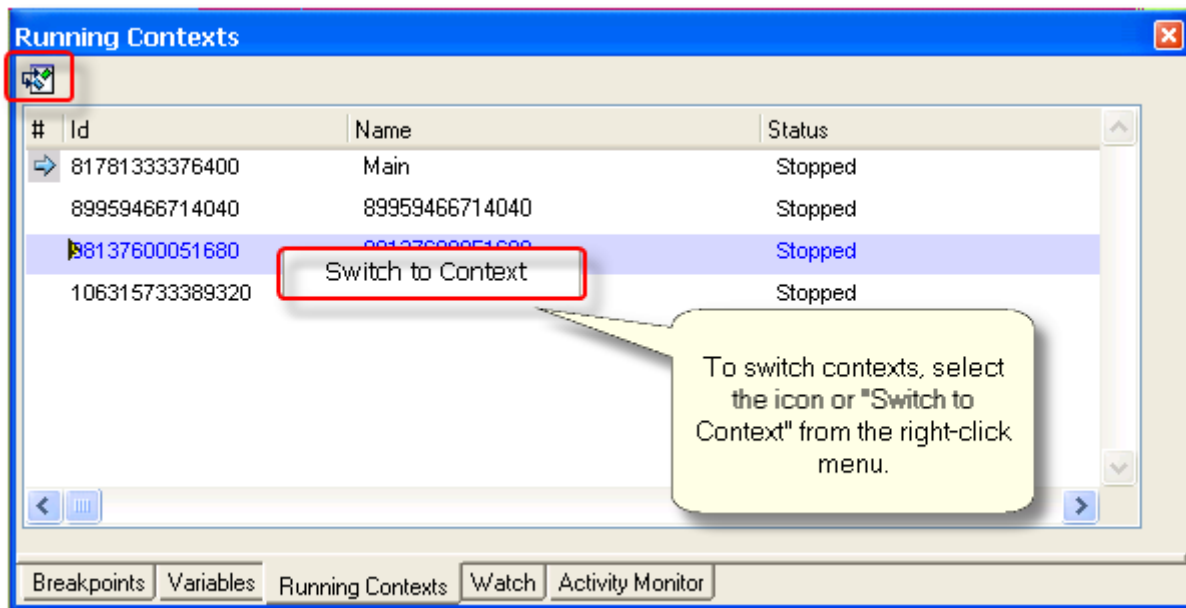
## 5. Stepping through your project

Once you have your breakpoints established, you can step through your program, operation by operation, while watching the execution in the runtime window. You have several step options, as shown in the table below.


Command	Accelerator Key	What it does
Continue	F7	Once the program has started running, Continue will cause the program to execute until it hits the next breakpoint.
Step	F10	Proceeds to the next operation in the current program, the pauses again. However, if the operation is a call to another program, it doesn't pause within the called program.
Step Into	F11	Works like Step, except that if the operation is a call to an eDeveloper task or program, or a Raise Event, you will remain in Step mode inside the called task or program.
Step Out	Shift+F11	Stops step-by-step operations within the current handler, but resumes them again in the calling handler.
Toggle Breakpoint	F9	If, while you are paused on a breakpoint, you decide you don't need to use this breakpoint any more, you can turn off that breakpoint by pressing F9.  Or, if you are stepping through a program with F10, you can set new breakpoints with F9.

When you are finished debugging, you can stop the program quickly by pressing **Debug->Stop**, or selecting the  on the toolbar. If there is a major problem with your program, you can restart the eDeveloper engine by pressing **Ctrl+Shift+F9 (Debug->Reset Runtime Engine)**.

## How do I Use the Debugger When Running Parallel Programs?



The debugger can be very useful when you are running multiple simultaneous contexts. The Context view will display a list of all the existing contexts.

From the Running Contexts list, you can switch between contexts by choosing **right-click->Switch to Context**, or using the  icon. This is only enabled for contexts where the status is “Stopped” so you may have to add breakpoints to the contexts you want to switch to.

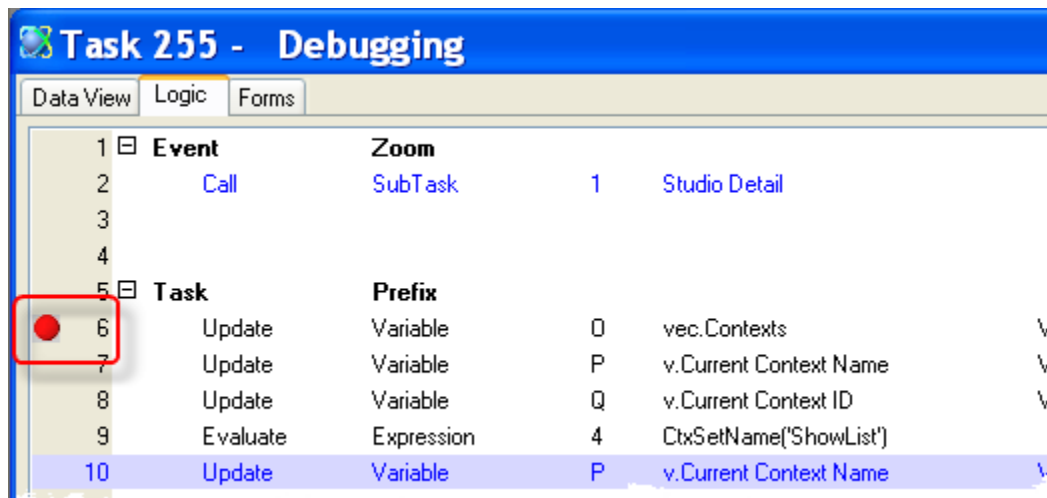
## How do I Set Breakpoints in the Application?

In order to view what is going on during a program while it is running, you will set breakpoints. There are three different kinds of breakpoints available:

- Setting a breakpoint on an operation
- Setting a breakpoint on a variable
- Using *Break Now*

Each of these are useful in different ways, and are described below.

### Setting a breakpoint on an operation



You set breakpoints within the eDeveloper Studio. Whenever the debugger encounters a breakpoint, it will pause. Once the debugger is paused, you can view the variables, call stack, and contexts. Then, you can choose to continue, or step through the code.

To set a breakpoint:

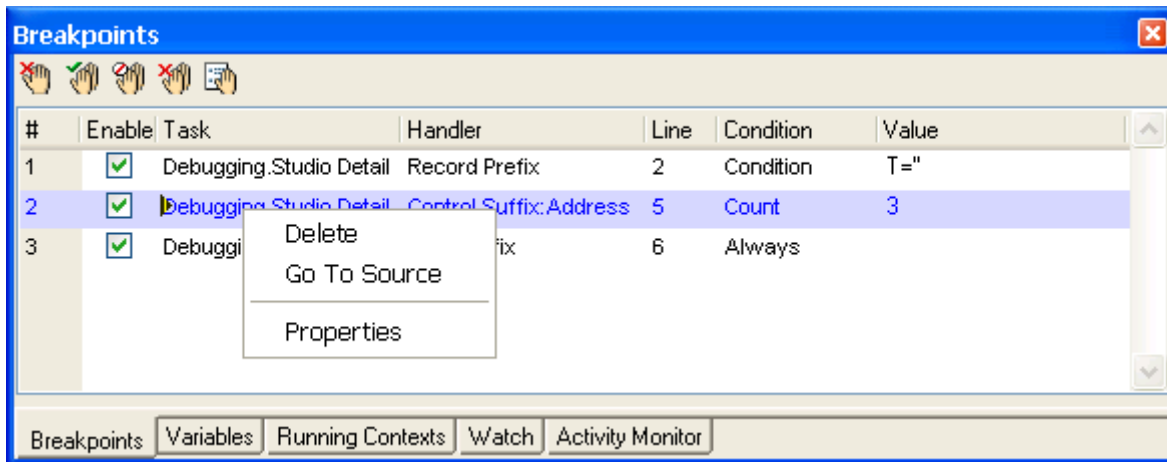
1. Go to the operation you want to break on.
2. Press **F9** (**Debug->Toggle breakpoint**).

You can turn off the breakpoint by pressing F9 again while positioned on that operation. However, you can also turn off a breakpoint by deleting it from the breakpoint pane. This is a convenient way to turn off all the breakpoints when you are finished debugging.

You can set breakpoints while your program is running. The program will open in read-only mode, so you can't change the actual program, but you can add and delete breakpoints.

In addition, you can modify the operation of the breakpoint in the breakpoint pane, as shown below

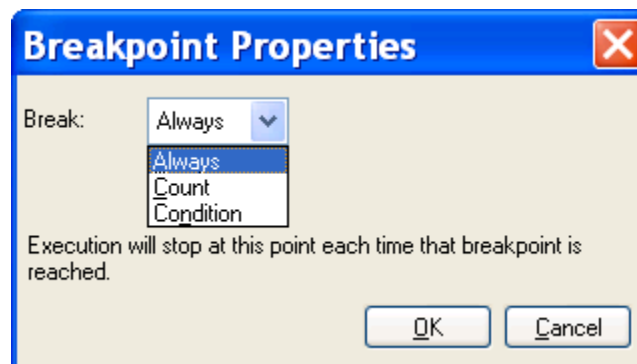
## Turning breakpoints on and off



Whatever breakpoints you set will show up in the Breakpoint pane. Here you can modify them in several ways. You can:

- Turn the breakpoints on and off by checking and unchecking the **Enable** box. Or turn all of them on with , or all of them off with .
- Delete an entry by selecting **Delete** from the right-click menu (or pressing F3, or the icon). Or delete all of them with .

## Finer control of breakpoints (breakpoint properties)



In addition, you can fine-tune when breakpoints happen by using the breakpoint Properties.

1. Position the cursor on the breakpoint you want to modify.
2. Select **right-click->Properties** (or , or **Alt+Enter**). The **Breakpoint Properties** will appear.
3. For the Break field, select *Always*, *Count*, or *Condition*.
  - If you select *Always*, there is nothing else to do.

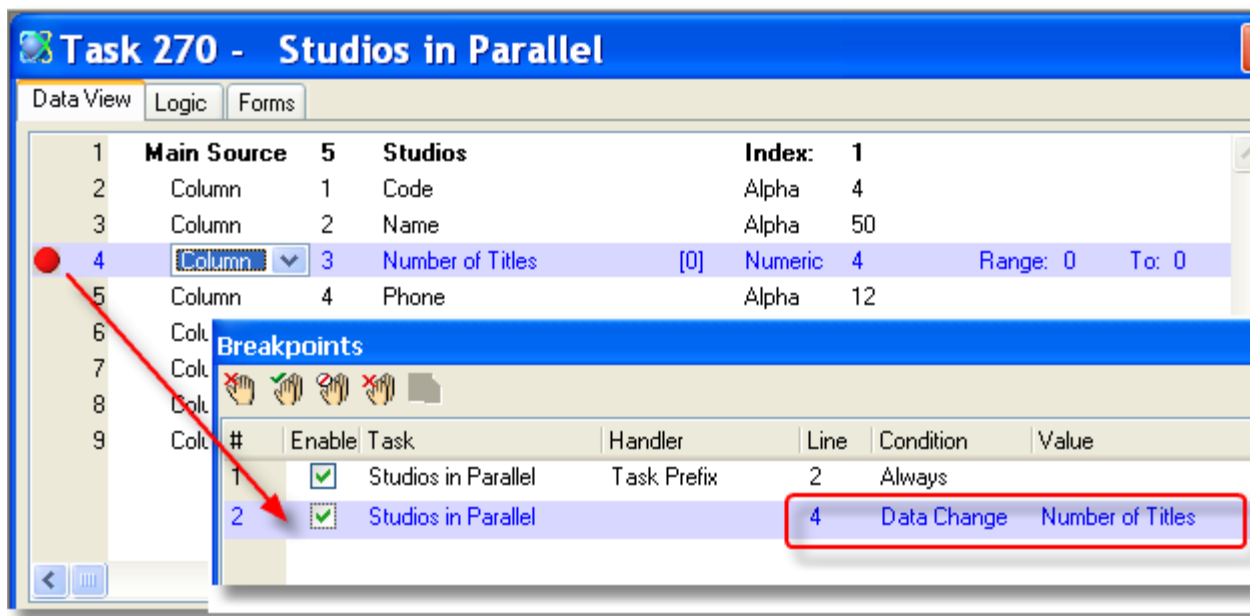


- If you select *Count*, enter the number of iterations you want to pause on. For instance, if you have a process that will loop 1,000 times, you can set the count to 300, and it will only start pausing after the first 300 iterations.
- If you select *Condition*, you can zoom to the Expression editor to enter a condition that will evaluate to True or False. The breakpoint will break if the condition evaluates to True.

### Go to source

Another useful thing you can do from the Breakpoints pane is to select **right-click->Go to source**. This will position you directly on the breakpoint in the Studio.

### Setting a breakpoint on an operation



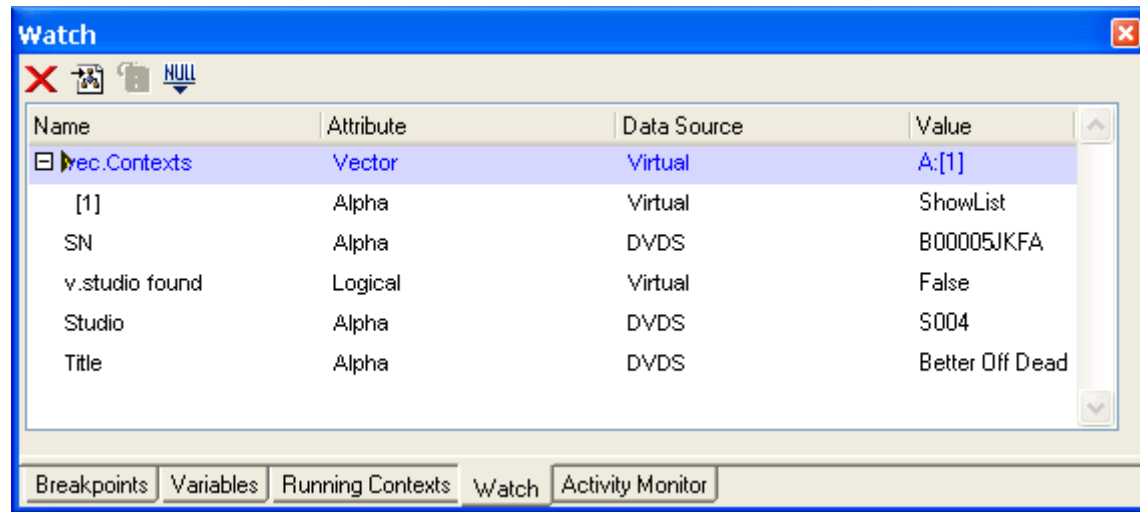
If you set a breakpoint on any variable in the Data View, then the Debugger will pause whenever the value of that variable changes. To set a Data Change breakpoint:

To set a breakpoint:

1. Go to the operation you want to break on.
2. Press **F9** (**Debug->Toggle breakpoint**).

The Data Change breakpoints are handled on the Breakpoint list as described for breakpoints on operations.

## Setting a watch on variables







It is also useful to have a shorter list of variables you are keeping an eye on. You can add variables to the Watch List. Then when you pause on your Data Change break, you can see how they changed. To add a watch on a variable:

1. In the Studio, position the cursor on the variable (Column or Virtual)
2. Select **Debug->Add to watch** or **right-click->Add to watch**

You won't see a red dot next to the variable as you do with a breakpoint, but you will see the variable listed in the *Watch* pane. The variables in the *Watch* pane are also visible on the *Variables* pane, but you can see them more conveniently.


From the *Watch* pane, you can:

- Turn off the watch by deleting the *Watch* entry (click on the , or press **F3**, or select **right-click->Delete**).
- Set the value of the variable to some other value (**right-click->Set data**, or select ).
- Set the value of the variable to Null, if allowed (**right-click->Set null to data**, or select ).
- Jump to the variable in the source code (**right-click->Go to source**, or select ).

## Using Break Now

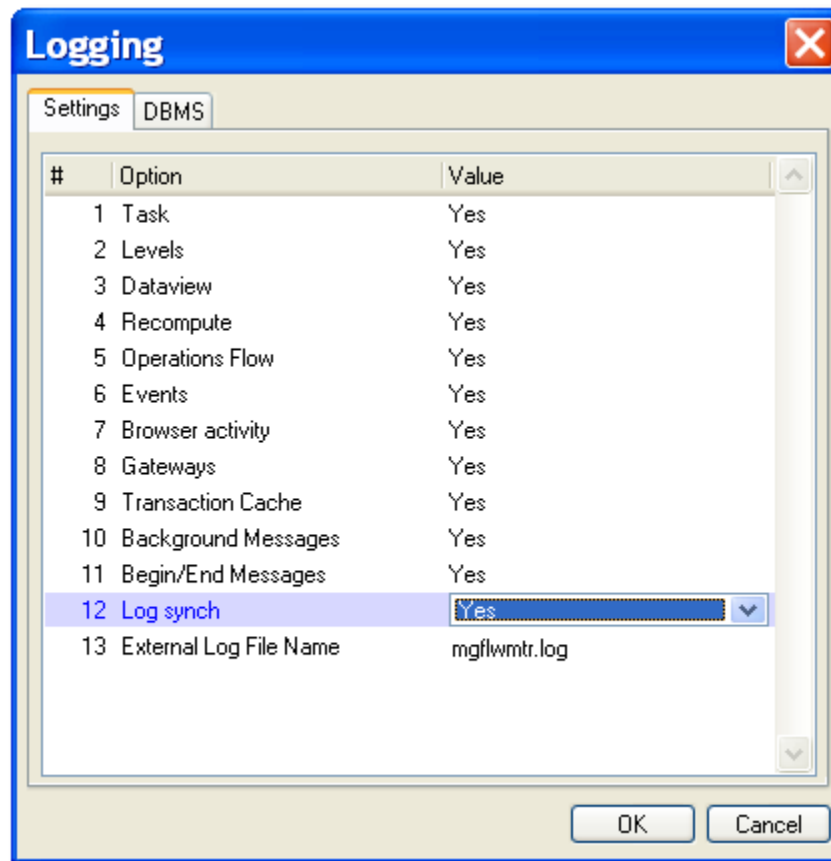
Even if you don't have any breakpoints or watches set, you can use *Break Now* to view the variables and call stacks whenever you like. This is particularly useful in online programs, where you might be working with the program and testing it, and then suddenly get a result you were not expecting. Since the variables, contexts, and call stack are only visible during a breakpoint, this using *Break Now* allows you to view those items.

To use *Break Now*:

1. Switch to the Studio window.
2. Press **Alt+Ctrl+F7** (**Debug->Break Now**, or press the  on the toolbar).

The debugger will pause in the same way it does during a breakpoint or watch.

## How do I Control the Information Logged by the Debugger?



You can control how much information gets logged by the Debugger by changing the settings on **Options->Settings->Logging**.

These settings affect not only the debugger, but also the external log file, if one is being used.

### The Logging() function

You can also turn logging on and off by using the **Logging()** function. This doesn't change the logging settings in the Magic.ini. The syntax is:

**Logging(start/stop, 'Filter')** where:

- **start/stop** is 'True' Log to start logging, or 'False' Log to stop it.
- **'Filter'** is a keyword that determines what logging entry will be started or stopped.

Here are some examples:

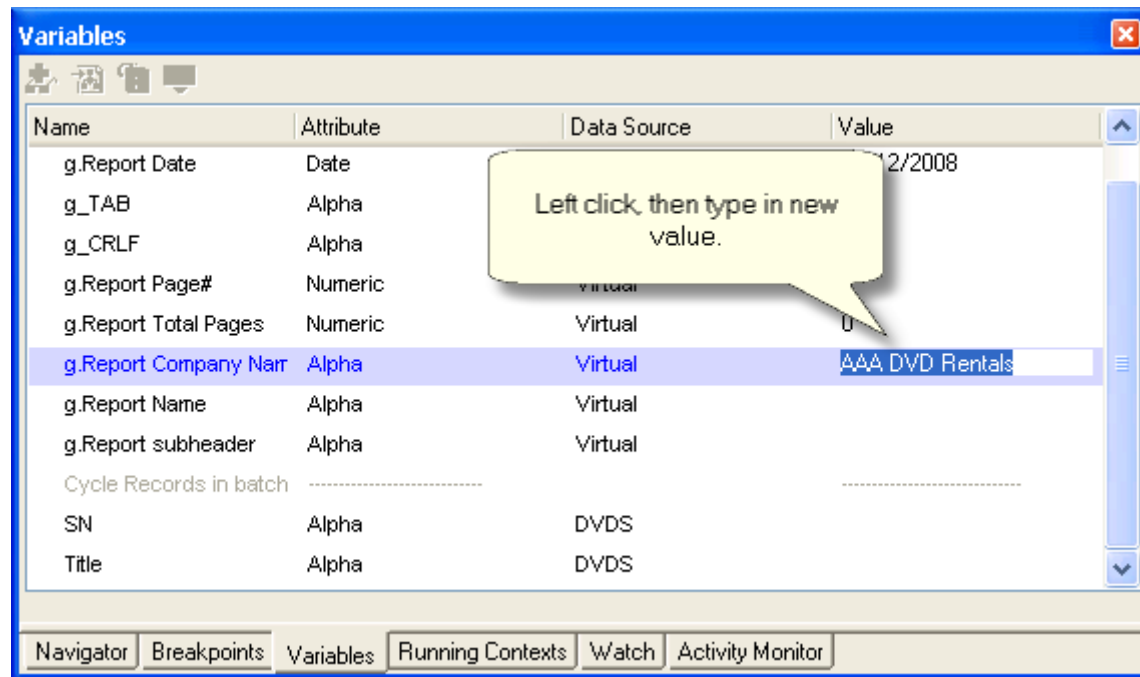
The Logging Expressions	Effect
Logging('True'Log, 'Levels')	Turns on logging for Levels
Logging('False'Log, 'Recompute')	Turns off logging for recompute activity.
Logging('False'Log, 'ALL')	Turns off all logging
Logging('True'Log, 'RESET')	Sets logging back to the value stored in the Magic.ini
Logging('True'Log, 'Oracle=D')	Sets Oracle logging to the Developer level.

See the eDeveloper Help for detailed information about the syntax.


### Performance issues

When using logging, do be sure it is turned off before putting the code into production. Logging significantly slows down the execution of a program.

## How do I Manipulate Data While Debugging?



You can change the data while the debugger is running. You can do this on either the *Variables* or the *Watch* pane.

1. Set your breakpoint so you will pause in a spot within the scope of the variable.
2. Go to either the *Variables* pane or the *Watch* pane, to find the variable you want to change.
3. Left click on the *Value* column, or select **right-click->Set Data**, or click on .
4. If the variable participates in a recompute, you will be prompted “Do you want all expressions that are affected by the new value to be recomputed?”. Click on the Yes button if you want the expressions to be recomputed.

## How do I Debug a Component?

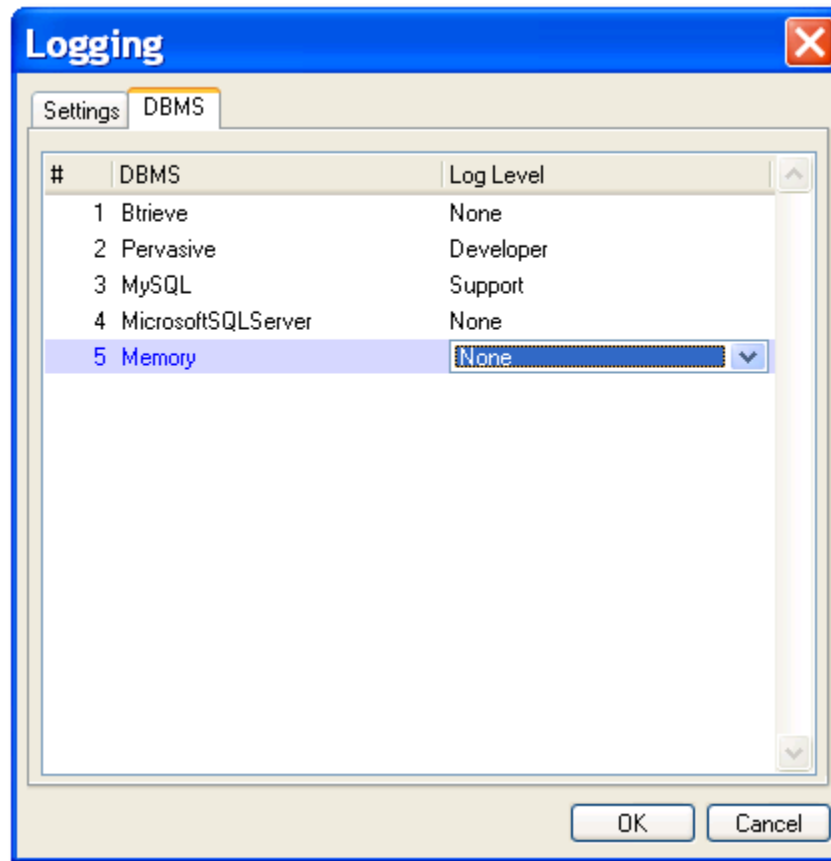
You can debug component programs as you would any program in the project, but only if the component is defined as a module of the current project.

To add a module to the current project:

1. Go to **Project->Add Module**. A file selector window will appear.
2. Select the project (**.edp** file) you want to use.

Once you have the component on the Navigator module list, you can click on the module to open it and add breakpoints. The breakpoints will show up on the same breakpoint list with the breakpoints from the calling task. When you reach the breakpoint in the component, the current task will close, the component will open, and you will see the execution of the code within the component.

## How do I Log the Database Activity?



You have the option of logging eDeveloper's interactions with whatever DBMS you are using. To turn on logging:

1. Go to **Options->Settings->Logging**.
2. Click on the *DBMS* tab.
3. Go to the *DBMS* you want to log.
4. Select a *Log Level* that isn't **None**. There are three logging levels, Customer, Support, and Developer. Customer is the lowest log level, which is the least verbose, and Developer is the highest log level.



## Log Level

Log Level	What it does
None	No logging is done
Customer	Lowest log level; only SQL commands are generated
Support	Medium log level
Developer	Full log

## DBMS Logging

In addition to eDeveloper's logging, you also might find it useful to use whatever logging facility is built into the DBMS you are using. These vary depending on the DBMS, but most tools have logging built in.

## Performance issues

When using logging, do be sure it is turned off before putting the code into production. Logging significantly slows down the execution of a program.



## Chapter 30: Events and Handlers

### How do I Refresh the Variable's Value After Returning From a Selection Program Opened by an Event?

When you are working in a handler, the values of the variables are not actually updated until you leave the handler. That means, if virtuals are updated within a handler that is tied to a field, the values do not show up onscreen to the user until the user leaves the field.

This is particularly an issue when you are using selection lists. Typically the selection list is accessed by the user zooming from a field. Here we will show you the various options you have for implementing these zoom fields in eDeveloper 10.

#### The problem: field doesn't refresh



First, here is the kind of zoom that will not work properly. When the user is parked on the StudioCode control, it brings up a selection list, and the user can select a studio. But the field will still show blank, until the user moves to the next field.

## Solution1: Using the Force Exit Event property

The screenshot shows two windows from the eDeveloper IDE. The top window, titled "Events: 263 - Solution1: Using force exit", displays a table of events. The bottom window, titled "Task 263 Solution1: Using force exit", shows the configuration for a task, with a red arrow pointing from the "e.Zoom with Force Exit" event in the top window to the "e.Zoom with Force Exit" entry in the task configuration.

#	Description	Trigger type	Trigger	Parameters	Force Exit
1	ge.Void	None		0	None
2	ge.Start	None		0	Editing
3	ge.Print	None		0	Editing
4	ge.COM Error	None		0	None
5	ge.LogError	None		0	Editing
6	e.Zoom with Force Exit	Internal	Zoom	0	Editing

Task 263 Solution1: Using force exit		on: StudioCode	Scope Task
1	Event		
2	Call	Program 261 Select Studio	[1 Arguments]
3			

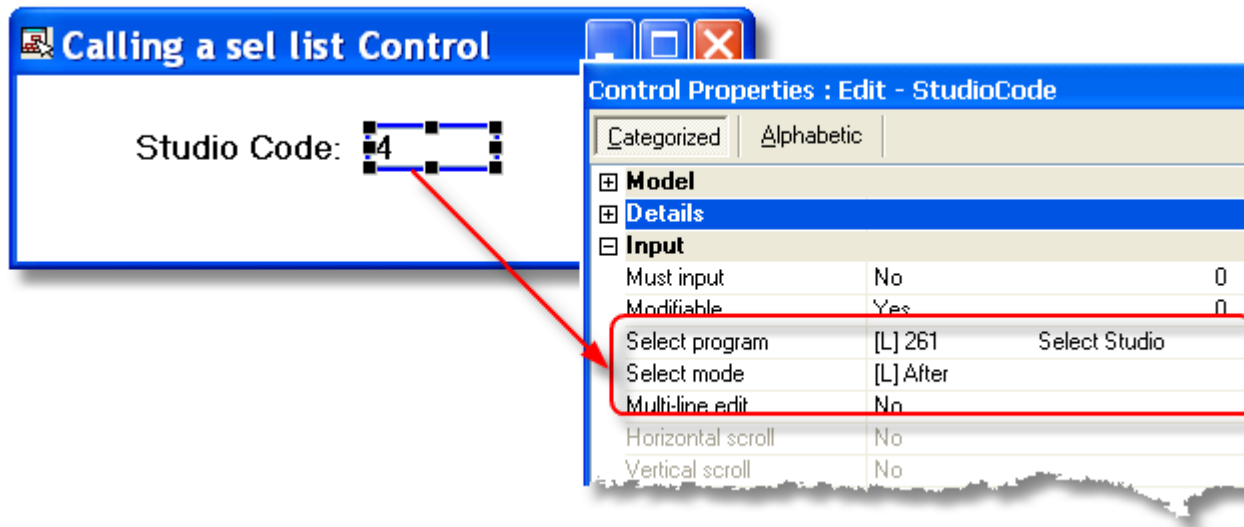
Here is the solution to making the field update.

1. A new event is created, which we named `e.Zoom with Force Exit`. This event is triggered by the zoom event, so it works like a zoom. However, the Force Exit property, on the right, is set to *Editing*. This forces the field to refresh itself.
2. The new event is used in place of the internal zoom event.

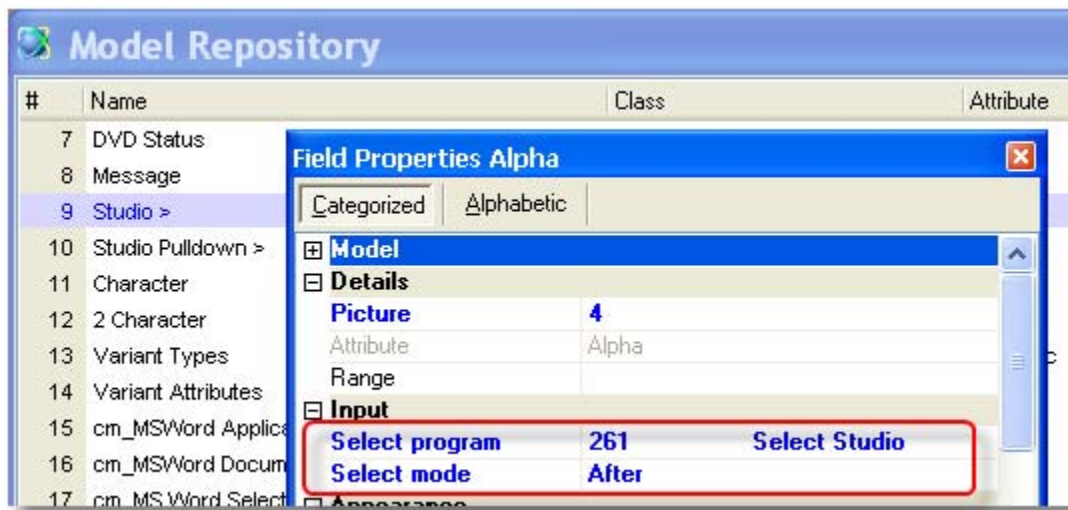
You can create this new event in the *Main Program* so it is global to the application.

**Note:** Normally when you are parked on a zoom field, the word “zoom” appears on the eDeveloper status bar at runtime, as a prompt to the user. When you use this method, the “zoom” prompt will still appear, because a zoom trigger is being used on the event.

**Hint:** If you want to replicate the action of a zoom after, then add a *Raise Event* for the Internal Event “Next Field”.

**Solution 2: Attach the selection list to the control**

Alternatively, you don't need to use an event at all for a selection list. You can simply use the Select Program property for the field.



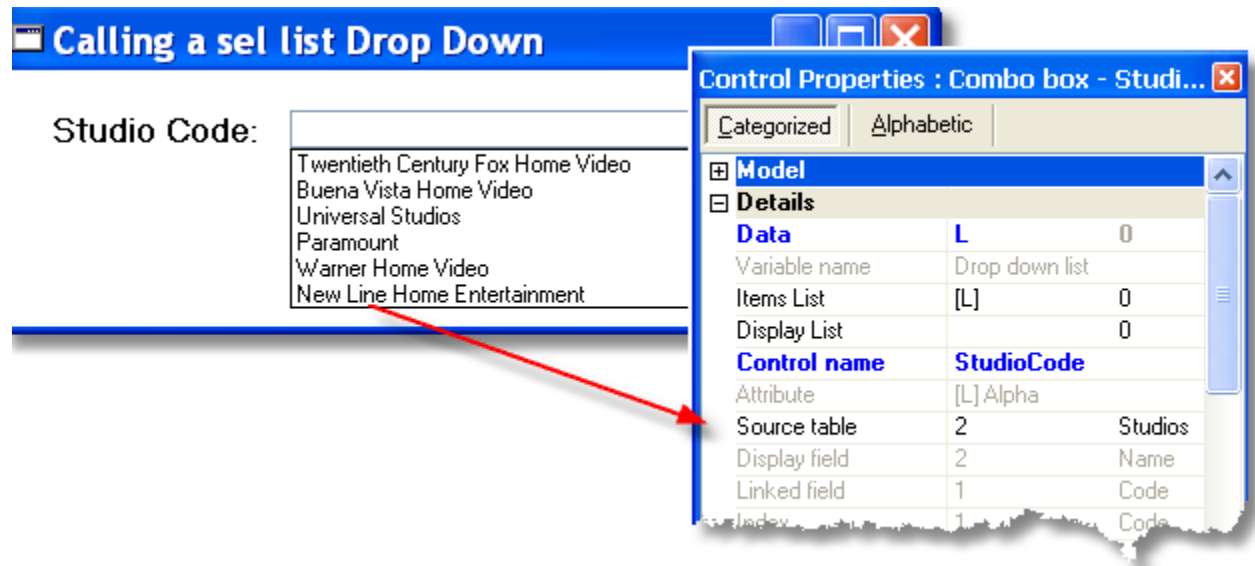
You can attach the program at the Model level, or at the Control level on the form. Either way, a zoom will call the selection program.

This has the advantage that you don't need to specify this in every task. In our example, the Model for "Studio code" will always call the "Select Studio" program unless we break the link.

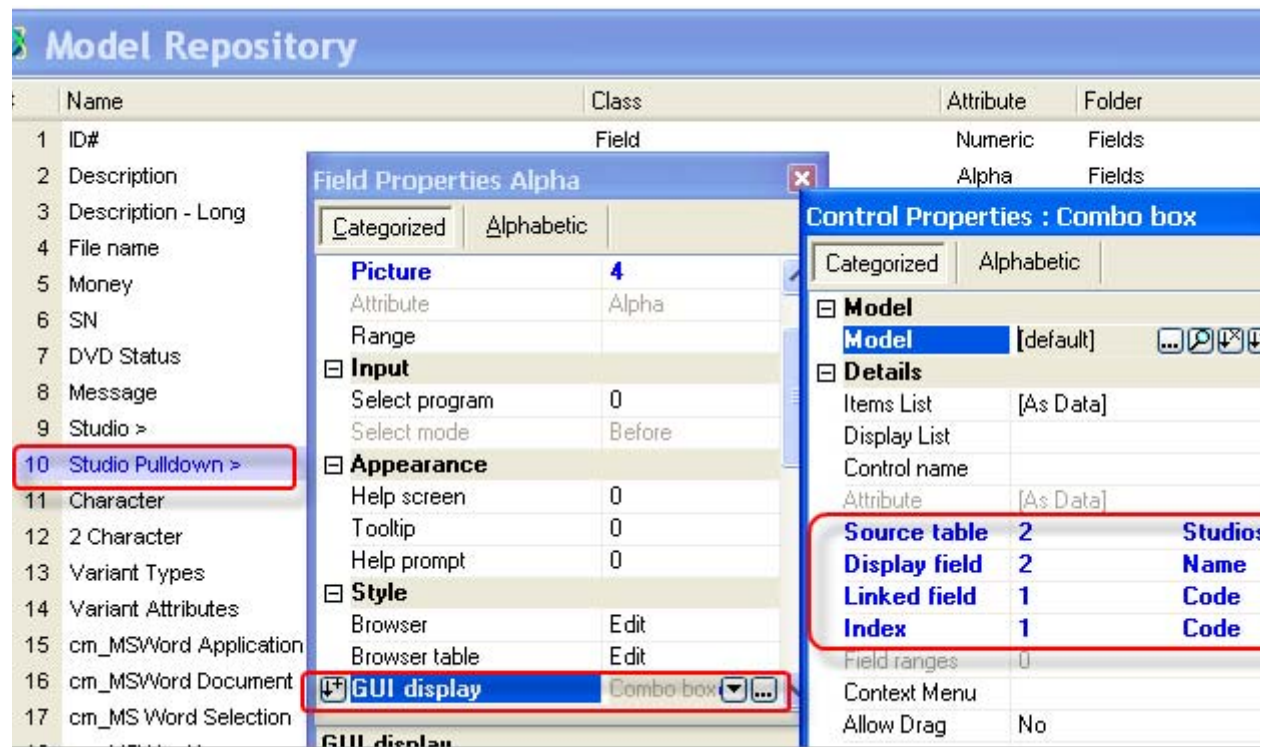
The disadvantage of this method is that it only works for selection lists that send back only one variable. In this example, we are only selecting the "Studio code". But if we needed to select a "Studio code" and a "Studio type", we would need to use an event.

## Solution 3: Use a Combo box

## Calling a sel list Drop Down

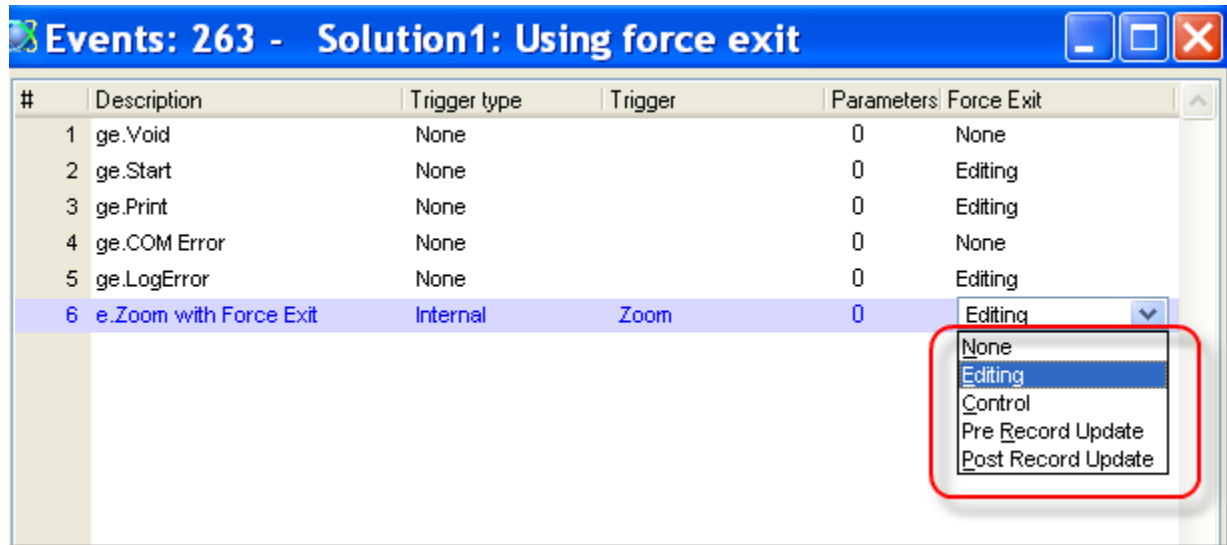


With eDeveloper 10, you can also just use a drop-down list, for many selection lists. Again, you can define this at a Model level, so you don't need to do it in every program. In this example, we created a Model



called Studio Pulldown, which shows as a *Combo box* and is automatically linked to the Studio table. All we need to do now is to use this model for the Studio code.

## How do I Force a Handler to Use the Newly Updated Values That Have Not Yet Been Committed?

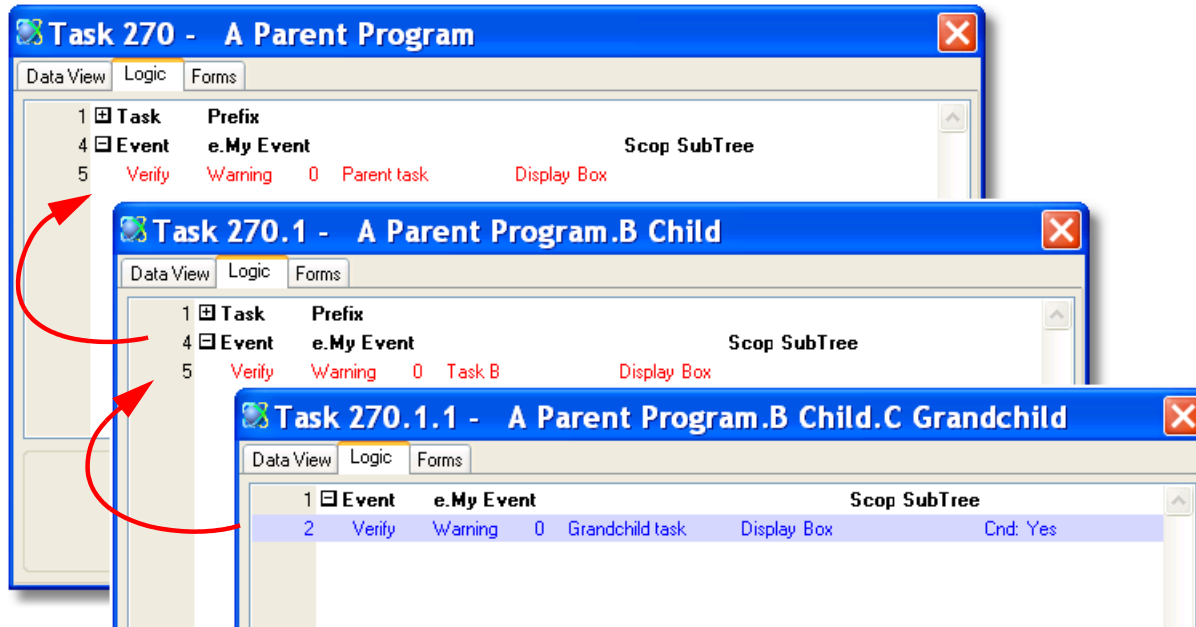


When you are working in a handler, the values of the variables are not actually updated until you leave the handler. That means, if virtuals are updated within a handler that is tied to a field, the values do not show up onscreen to the user until the user leaves the field.

In other cases, you might want the record to be updated before actions in the handler are completed. This might be the case, for example, if you are calling another program to print an order you are still editing. You want the printing program to print the most committed record.

You can handle these sorts of situations by using the “Force Exit” column of the Events repository. The exact way each of these work is spelled out in the eDeveloper Help system, but in practice, you can experiment and use the Debugger for your particular program. **Force Exit = Editing** works for most “field update” issues.

## How do I Handle the Same Event Using Several Handlers?

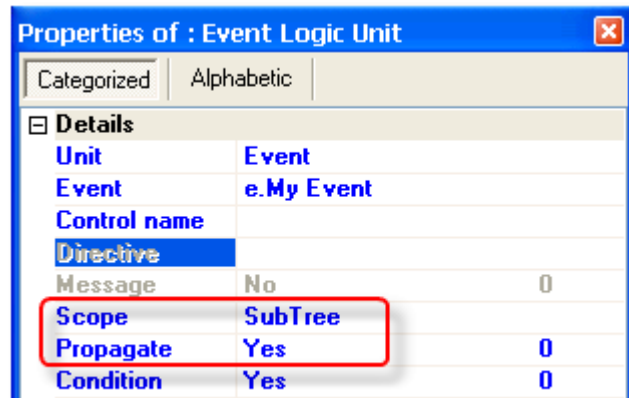


An event that is raised in eDeveloper can propagate “up.” If an event is raised in a child task, it can be handled in that task, and all the ancestor tasks, including the Main program.

### Allowing an event to propagate

You can control whether or not the event stops at a given handler or not, by setting the properties of all the logic units in the event tree.

1. Set the Propagate property to Yes (or write an expression that evaluates to ‘TRUE’ LOG when you want it to propagate).
2. For any task involved that is not the lowest-level one, Set the Scope property to Subtree.

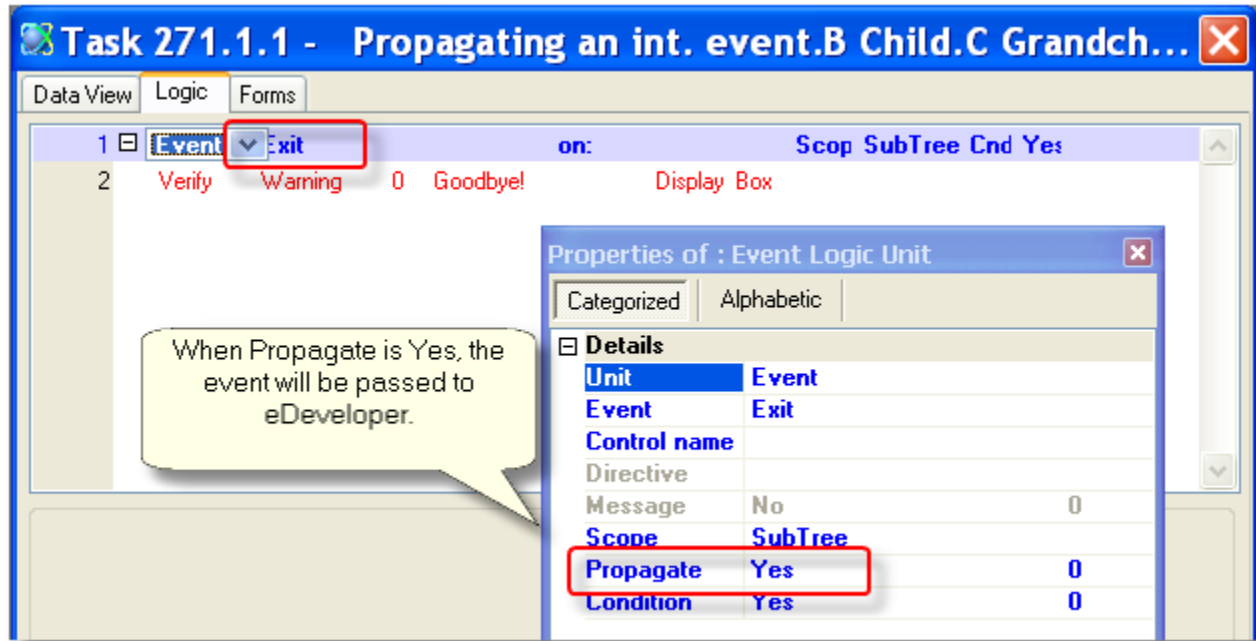


Now the event will be handled at one level and passed up to the next.


This works for all user events. For internal events, you may have to re-raise the event. For instance, if an Exit event is raised in the grandchild, it will be “used up” when the grandchild task exits. So if you want the next task up to also handle the *Exit* event, you would have to raise another Exit event in the grandchild task. (Subforms, however, would be a better choice in that case).



## How do I preserve eDev Functionality for Internal Events Handled by User Handlers?

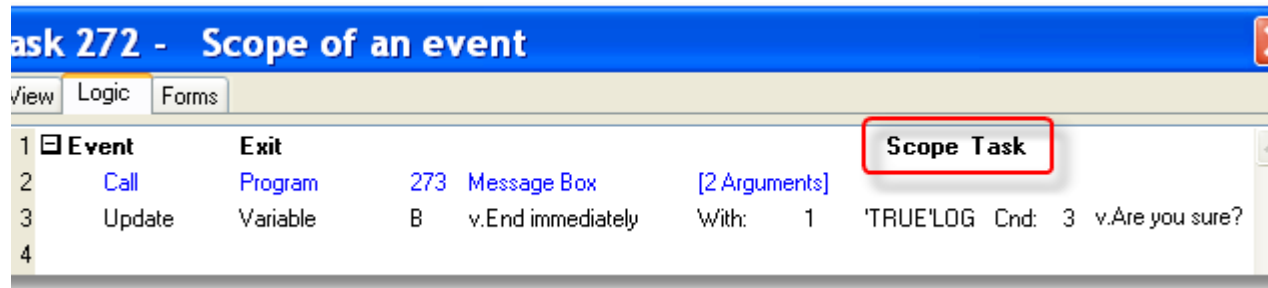


When you create a handler that is triggered by an Internal event, you have the choice of blocking that event, or propagating it.

In this example, when the user presses **Escape** or clicks on the  to exit the task, it raises the internal *Exit* event. With *Propagate* set to **Yes**, the warning message will display, then the *Exit* event will be passed to eDeveloper, which will close the task.

If *Propagate* was set to **No**, then the message would display, but the *Exit* would not be passed to eDeveloper, and the task would stay open.

## How do I Ensure That the Handling of an Event Will Only be Active in the Task Where It Was Declared?

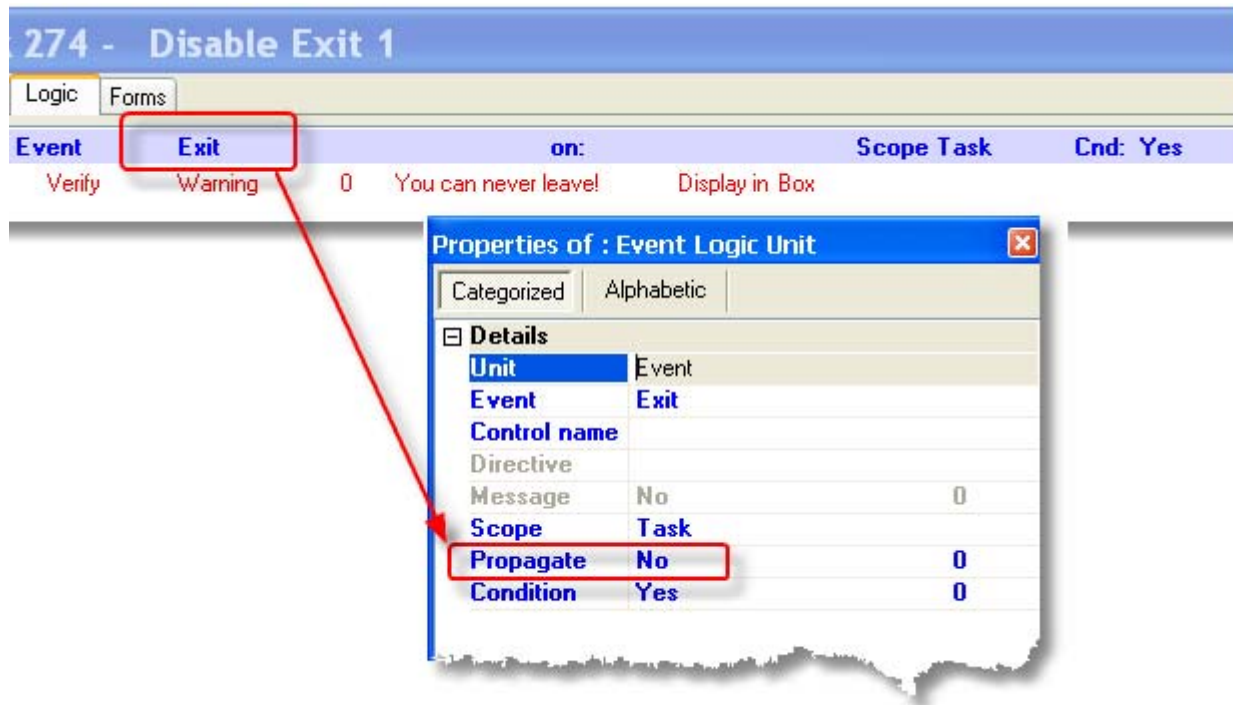




Sometimes you will want an event to only be active within a certain task. For instance, you might have a warning message that comes up when a user exits a certain critical task. However, that task might have descendent tasks that aren't so critical, and you don't want the warning message to pop up for those tasks.

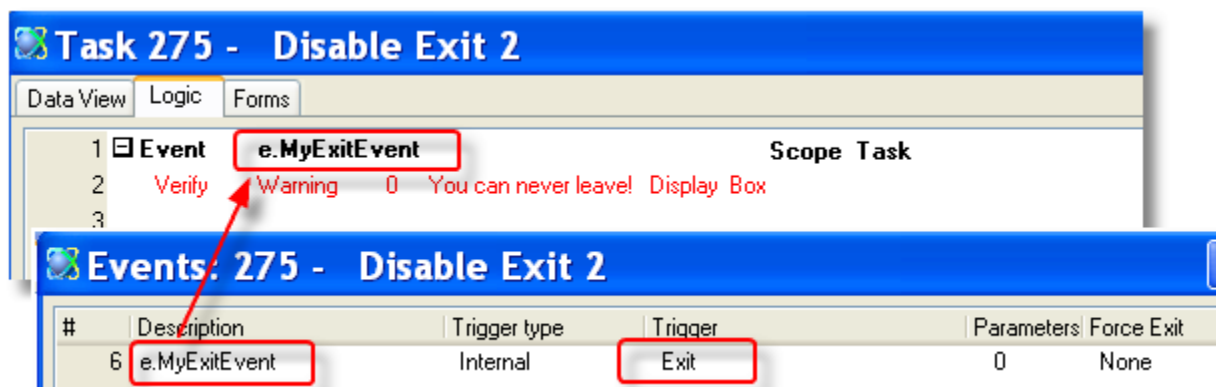
In this example, when the user presses Escape, an "Are you sure" message appears. If the user answers "Yes", then a variable is updated to force the task to exit. The original Exit action isn't propagated, so the task won't exit otherwise.

To set the handler so the event will only be handled within this task, set the *Scope* property to **Task**.

## How do I Disable eDeveloper Handling for Internal Events?



You can disable eDeveloper handling of Internal events by setting the Propagate property to No. In this example, the user can press Escape or click on the  and the task will never close, because the Exit event is effectively blocked. (Fortunately, when testing in the Studio, you can always use the  **Debug->Stop** option to get out of situations like this!).



What isn't so obvious is that the event will also be blocked if it is used as a Trigger to a User event. This task works just as the other one does: it totally blocks the user from exiting.

If you want to block an event, but also allow it to function under certain conditions, then you can use an Expression for the Propagate property. This is explained in the next example.

### Using Conditional Propagate

**Task 272 - Disable Exit Message**

**Logic View**

Line	Event	Program	Message Box	Arguments
1	Event	Exit	273 Message Box	[2 Arguments]
2	Call	Program	273 Message Box	[2 Arguments]
3	Update	Variable	B v.End immediately	With: 1 'TRUE'LOG Cnd: 3 v.Are you sur
4				

**Properties of : Event Logic Unit**

**Details**

Property	Value
Unit	Event
Event	Exit
Control name	
Directive	
Message	No 0
Scope	Task
Propagate	Yes 4
Condition	Yes 0

**Expanded View**

v.End immediately

You can also use an Expression with the Propagate property. Here, we trap the Exit event and give the user an “Are you sure?” message. If the user answers Yes, we update the variable “v.End immediately” to ‘TRUE’LOG. “v.End immediately” is used in the Propagate property, so if the user answers Yes then the Exit event is passed on to eDeveloper, and the task ends.

## How do I Restrict the Handling of an Event to a Specific Control?



Sometimes you will want to handle an event only if it happens while the user is working with a specific control.

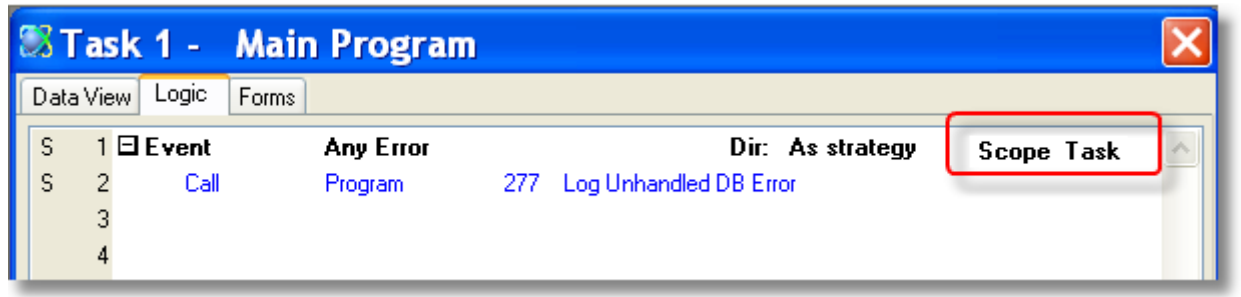
To select the control:

1. Go to the on: field of the event.
2. Zoom to the list of available controls.
3. Select the control you want to use.

Alternatively, you can just type in the Control name. The Control does not have to be in the scope of the task, since the Event handler may in fact be in a parent task or the Main program. In fact, the Control doesn't even have to exist yet.

**Hint:** You use the Control name to create generic events that work on many controls. For example, you might have several controls named "Order", which always contains an Order#. Pressing F1 on an Order control brings up a help screen for entering orders. Pressing F2 might show the current status of that order. Pressing Ctrl+P would print the order. But those same shortcut keys would call different programs if the control was named, say, "Customer".

## How do I Provide a Default Handler for my Entire Application?



If you want to create a Handler that will work for your entire application:

1. Create the handler in the Main program.
2. Set the Scope to Global

In this example we trap the *Any Error* event, and log the message using the database “err” functions. If the DB error was handled by a lower-level task, this handler will never be executed.

## How do I Handle Events Raised in a Hosting Application Using a Handler Defined in a Component?

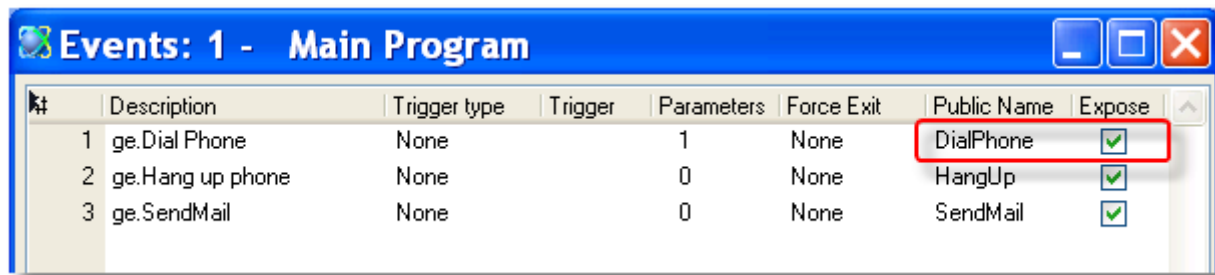
You can raise events in the host application, and have those events handled in the component. For instance, we could define an event to dial a phone number, and raise it in the host application when the user zooms on a phone number field.

There are three parts to this process. You need to:

- Define the event in the component
- Handle the event in the component
- Raise the event in the host

These three steps are defined below.

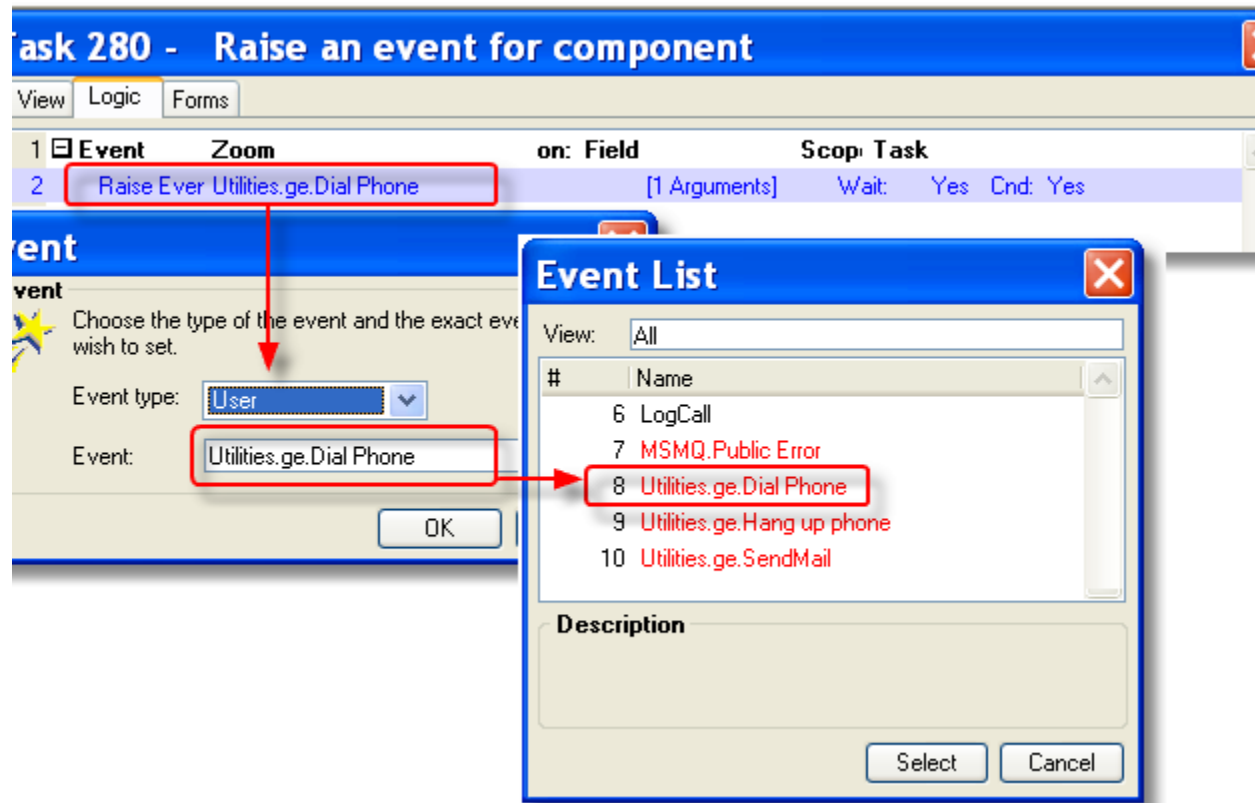
### Define the event in the Component



1. Enter the event in the Main program of the component. Make sure that Expose is checked, and that it has a public name.
2. Choose the Event when you are creating the *.eci*.

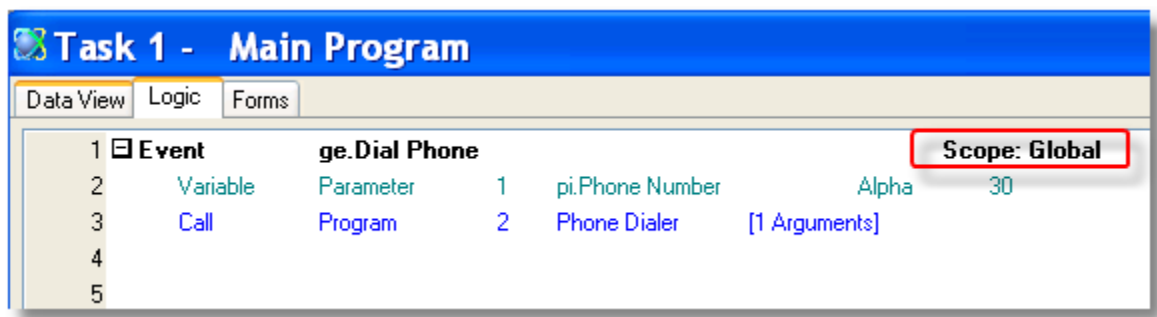
Now, when you work with the component, you will be able to choose this event.

## Raising the event in the Host



1. Go to the host program, to where you want to raise the event.
2. Raise the event as you would any user event. The component event will show up in red, underneath the host's own user events.

## Handling the event in the Component



1. Go to the component.
2. Go to the Main Program.



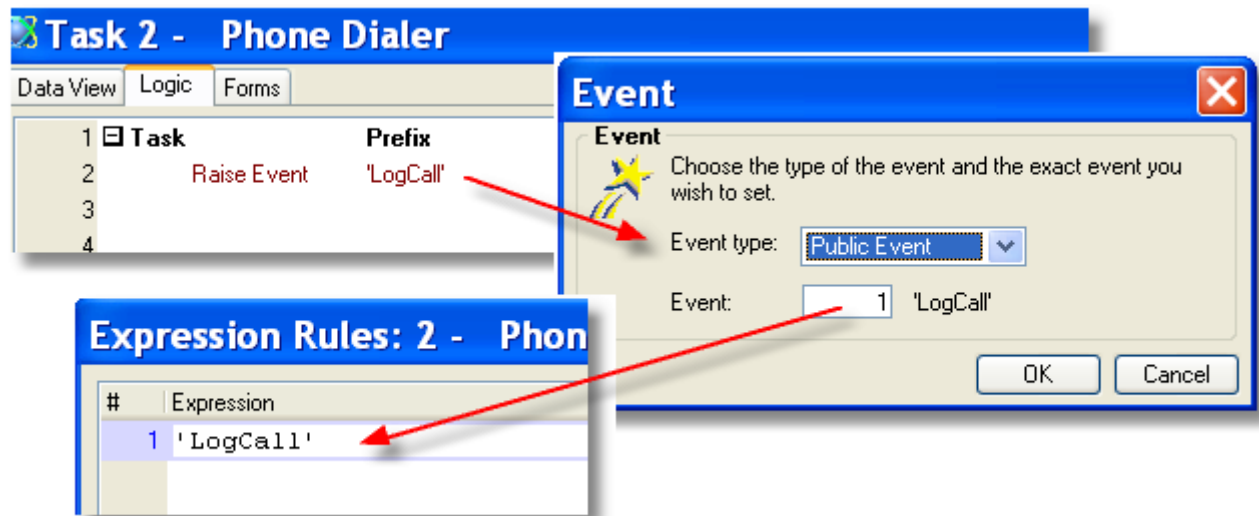
3. Create the event handler for your event. Set the *Scope* to **Global**.

Now this handler will execute when the host program raises the event.

## How do I Raise Events Defined in the Host Application From a Component?

Sometimes it is useful to be able to raise an event in a component that can be handled in the host component. For instance, it is common for packaged routines to raise an event if an error happened. In our example here, we raise an event in the component called “LogCall”, which passes parameters about the call, so the host application can store the information if needed.

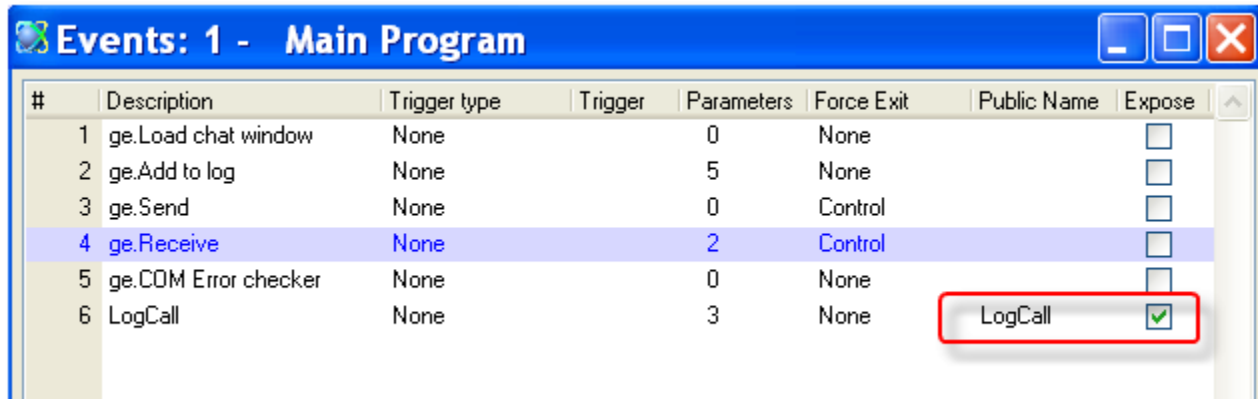
### Raising the host event



1. Create your *Raise Event* operation in the location you want it. The Event dialog will appear.
2. Go to the *Event type* field. Select **Public Event**.
3. Go to the *Event* field. Zoom, which will bring you to the Expression rules. Type in the name of the event, spelled exactly as it will be in the host event, in single quotes.

This will cause the component to raise an event that can be handled by the host component. Now you need to set up the host component to capture the event.

## Handling the event in the host



1. Create an event in the Main program, with the exactly the same *Public Name* as the event you raised in the component.
2. Check the *Expose* box.

Now, you can create handlers to handle the event as needed.

## How do I Force Immediate Handling for Events?

If you want an event to be handled immediately, set the *Wait* property to **Yes**. *Wait=Yes* causes the event to be handled immediately, without processing other commands. This is called *synchronous* processing, because the commands are handled in order.

**Task 281 - The effect of Wait**

Data View Logic Forms

Line	Event	Command	Wait	Scope	Task
1	Event	e.Start			
2	Raise Event	e.A	Wait: No		
3	Raise Event	e.B	Wait: No		
4	Raise Event	e.C	Wait: Yes		
5	Raise Event	e.D	Wait: No		
6	Verify	Warning	0	E	Display in Box
7					
8	Event	e.A			
9	Verify	Warning	0	A	Display in Box
10	Event	e.B			
11	Verify	Warning	0	B	Display in Box

For example, here we raise four events:

- Event A is raised, with *Wait=No*. Event A is put in the queue, but not executed. The next command is processed.
- Event B is raised, with *Wait=No*. Event B is put in the queue, but not executed. The next command is processed.
- Event C is raised, with *Wait=Yes*. Event C is immediately executed, without reading the next command.
- Event D is raised, with *Wait=No*. Event B is put in the queue, but not executed. The next command is processed.
- The next command is a Verify box that displays E. It is immediately executed.
- Then the queued events, A, B, and D, are executed.

The user will see the verify boxes in the following order: C, E, A, B, D.

---

## How do I Postpone the Handling of an Event?

If you want an event to be handled later, set the *Wait* property to **No**. *Wait=No* causes the event to be handled after other commands are read. This is called *asynchronous* processing, because the events can be interrupted and might not be handled in the order they are read.

For more information about how the *Wait* property works, see Chapter 30, “How do I Force Immediate Handling for Events?” on page 744.

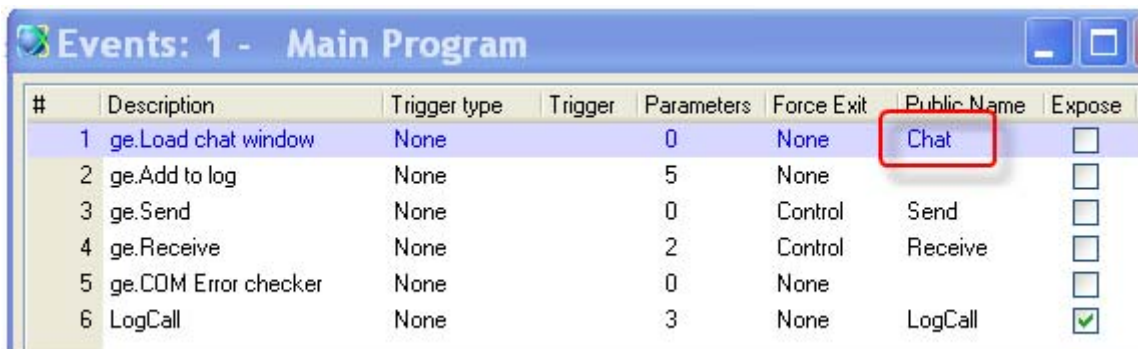
## How do I Raise A Dynamically Named Event?



There might be occasions where you want to raise an event using its text name. This might be the case, for instance, if you are raising an event in the host program from a component: you cannot just select the event because it is not in scope. Or, you might want to select between various events based on user input, or store the event name in a table.

In our example, we allow the user to choose the event from a list of events, then execute the chosen event.

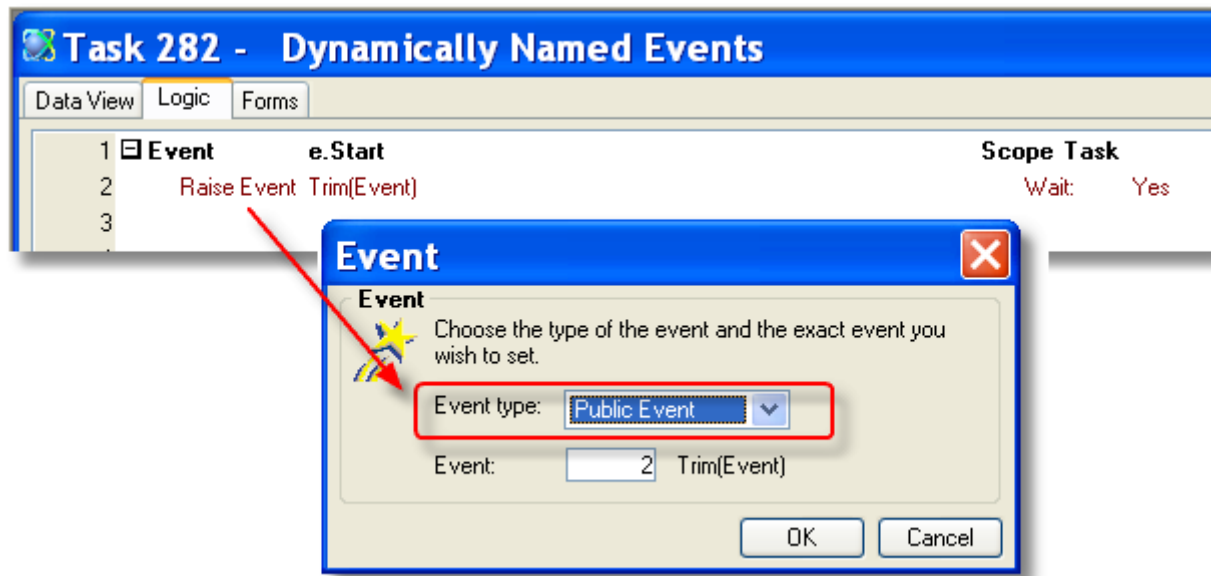
### Set up the event in the Main Program



#	Description	Trigger type	Trigger	Parameters	Force Exit	Public Name	Expose
1	ge.Load chat window	None		0	None	Chat	<input type="checkbox"/>
2	ge.Add to log	None		5	None		<input type="checkbox"/>
3	ge.Send	None		0	Control	Send	<input type="checkbox"/>
4	ge.Receive	None		2	Control	Receive	<input type="checkbox"/>
5	ge.COM Error checker	None		0	None		<input type="checkbox"/>
6	LogCall	None		3	None	LogCall	<input checked="" type="checkbox"/>

Before you can call a dynamically-named event, you must set up the event in the Main Program, and give it a Public Name. Here we have our events: *Chat*, *Send*, *Receive*, and *LogCall*.

Call the event with a Call public



Next, you need to call the event.

1. Go to the location where you want to raise the event.
2. Type 'R' to create a *Raise Event* operation. The *Event* box will appear.
3. Go to the *Event Type* field. Select **Public Event**.
4. Go to the *Event field*. Zoom to enter an expression. Make that expression evaluate to the exact name of the event (case matters, as do trailing spaces).

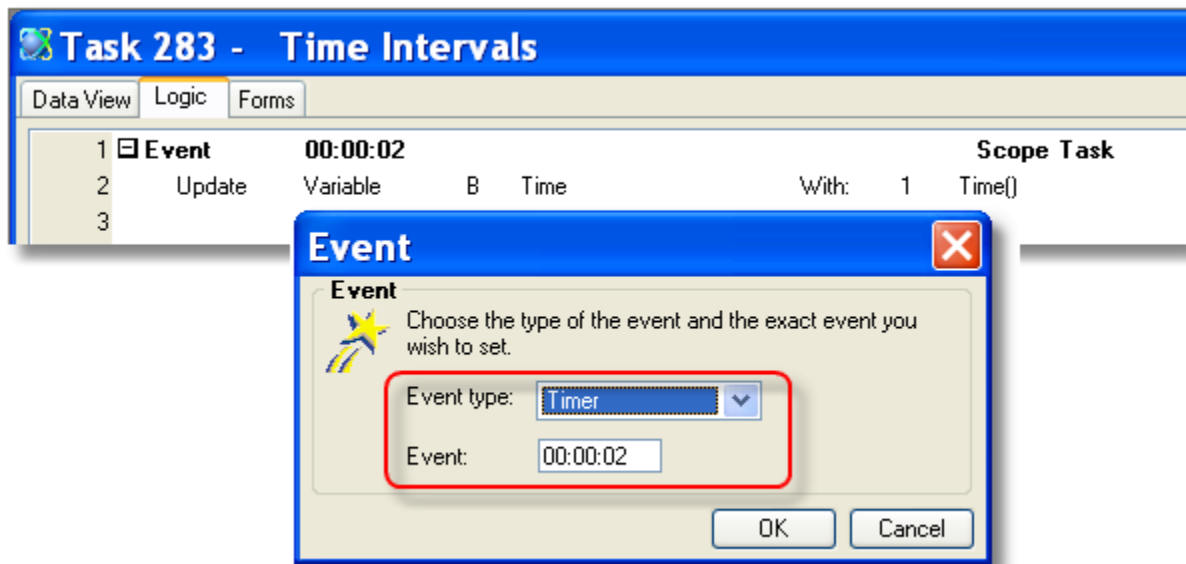
In our example, the user chooses an item from a list, which happens to be the exact name of the event. We just trim the spaces off it and raise it as a *Public Event*.

## How do I Execute a Set of Operations After a Time Interval?

It is often the case that you will want to have a process that operates based on some amount of elapsed time. For instance, you might want to check a mail queue every few minutes.

In eDeveloper this would be done using a *Timer* event. In our example, we are updating an onscreen clock every 2 seconds.

### Using a Timer event



1. Go to the location where you want to raise the event.
2. Type 'R' to create a *Raise Event* operation. The *Event* box will appear.
3. Go to the *Event Type* field. Select **Timer**.
4. Go to the *Event field*. Type in the amount of time you want to elapse between iterations of the event. The format is HH:MM:SS. In our example, we entered 00:00:02, so the event will be triggered every 2 seconds.

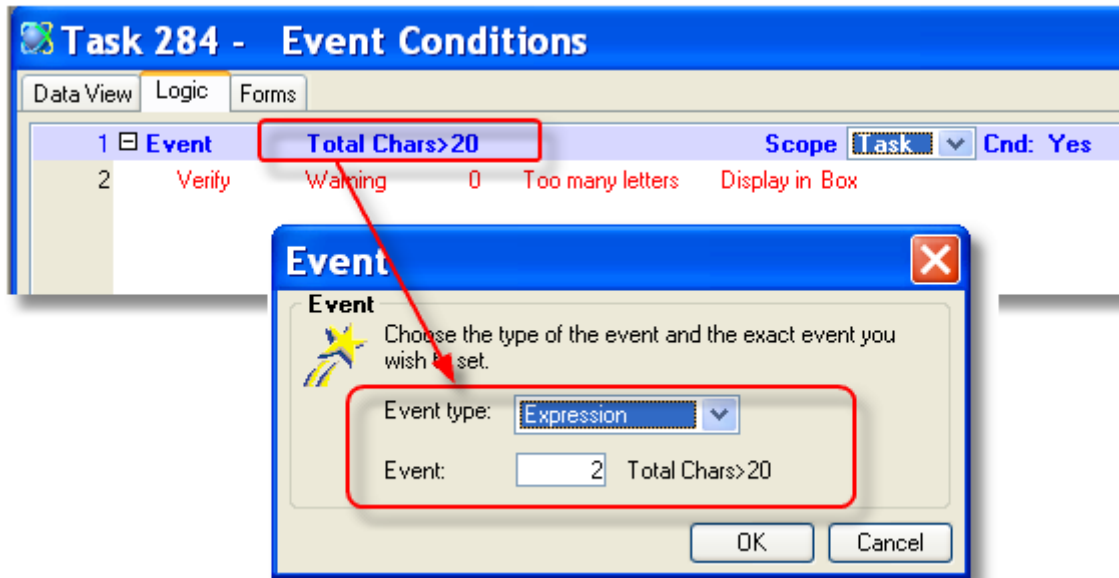
Now, the event will be triggered according to the time interval you entered.



## How do I Execute a Set of Operations When a Condition is Met?

It is often the case that you will want to have a process that happens based on some particular condition. For instance, you might want to give a message when an order total gets too large, or when email arrives. These sorts of events are entered using the an *Expression* event.

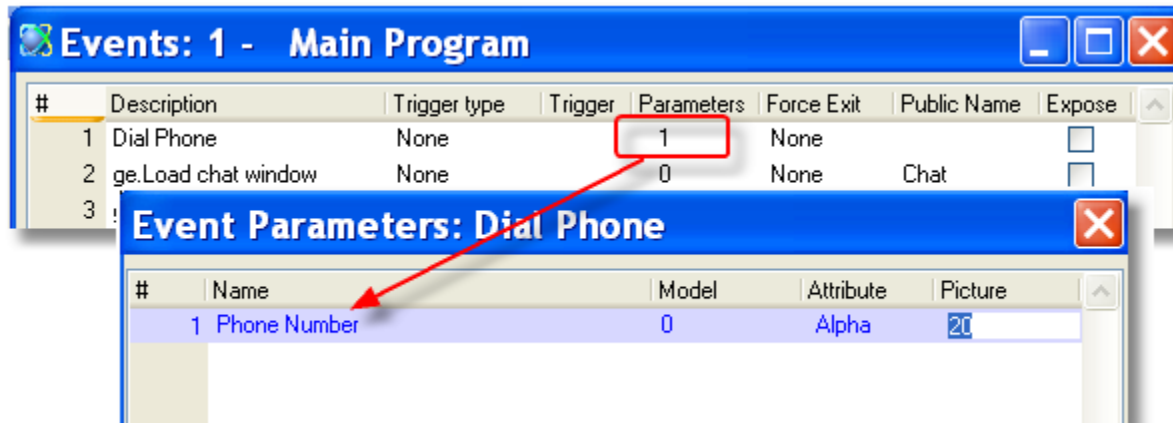
### Creating an Expression event



1. Go to the location where you want to raise the event.
2. Type 'R' to create a *Raise Event* operation. The *Event* box will appear.
3. Go to the *Event Type* field. Select **Expression**.
4. Go to the *Event field*. Zoom to the Expressions list. Enter an Expression that will evaluate to 'TRUE' LOG when you want the event to be triggered. In our example, the event will be triggered when the user types in more the 20 characters.

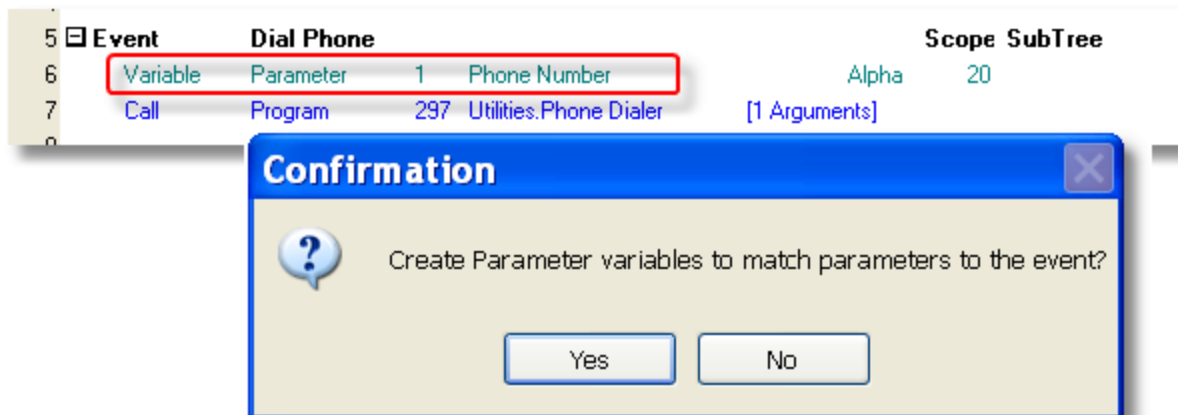
## How do I Send Information With a Raised Event?

Create the event with parameters



1. Create your event as usual.
2. Go to the Parameters column. Zoom. the Event Parameters list will appear.
3. Enter the parameters you would like the Event to accept.

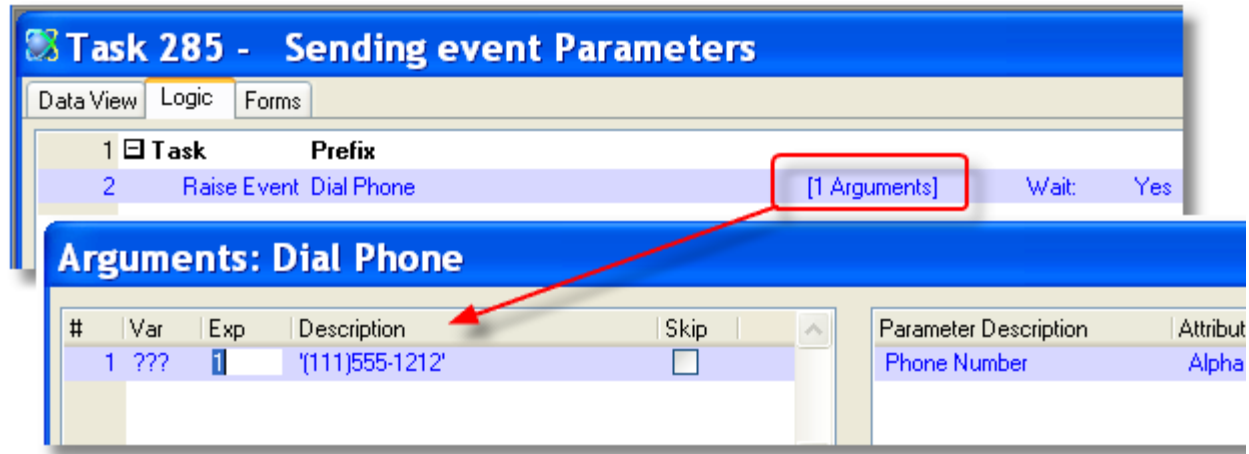
Accept the parameters in your Event handler



1. Go to the location where you want to raise the event.
2. Type 'R' to create a *Raise Event* operation. The *Event* box will appear.
3. Go to the *Event Type* field. Select **User**.

4. Go to the *Event field*. Zoom to select the event. Since the event we just created has parameters, you will get a Confirmation box: “Create Parameter variables to match parameters to the event?”. If you click Yes, then the parameters will be automatically created for you.

### Call the event



Now, when you raise the event, you can pass a parameter list, just as you would if you called a program.



## Chapter 31: Security

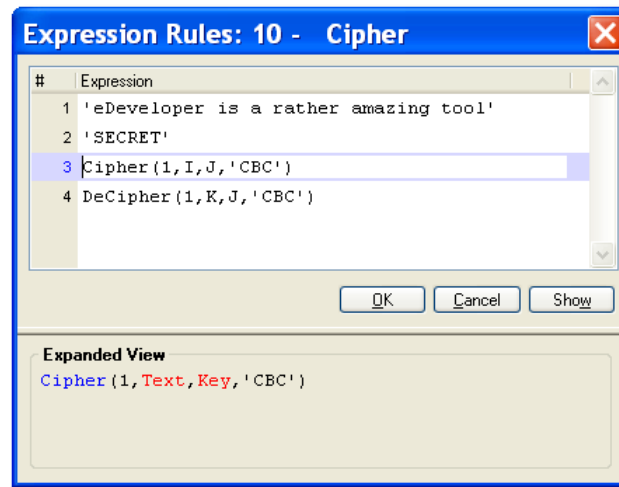
### How do I Encrypt And Decrypt Data?



You can use the eDeveloper **Cipher()** and **DeCipher()** functions to specifically encrypt and decrypt a BLOB of data. These support specific encryption algorithms, so you can decipher data coming from other applications. The supported algorithms include both symmetric algorithms like Blowfish and asymmetric algorithms such as RSA.

Symmetric algorithms make use of the same key for encrypting and decrypting data. Other algorithms are asymmetric, and you need a key pair, one to encrypt and one to decrypt.

## Using Cipher()



The syntax for **Cipher()** is:

```
Cipher(Cipher ID, Buffer, Key [, Mode, IV])
```

Where:

- **Cipher ID** is a number representing which encrypting algorithm to use. In our example we used *I*, which is Blowfish (See Chapter 31, “Supported encryption methods” on page 756).
- **Buffer** is the BLOB that contains the data to encrypt.
- **Key** is a BLOB containing the key. The required key length depends on which algorithm you are using. In our example our key is the word ‘SECRET’.
- **Mode** is an optional parameter specifying which mode to use. The allowable modes depend on the encrypting algorithm.
- **IV** is a BLOB containing an initialization vector. This parameter is also optional.

**Cipher()** returns a BLOB, which contains the encrypted text.

## Using Decipher()

The syntax for **Decipher()** is identical to that of **Cipher()**:

```
Decipher(Cipher ID, Buffer, Key [, Mode, IV])
```

Where:

- **Cipher ID** is a number representing which encrypting algorithm to use. In our example we used *I*, which is Blowfish (See Chapter 31, “Supported encryption methods” on page 756).

- **Buffer** is the BLOB that contains the data to encrypt.
- **Key** is a BLOB containing the key. The required key length depends on which algorithm you are using. In our example our key is the word 'SECRET'.
- **Mode** is an optional parameter specifying which mode to use. The allowable modes depend on the encrypting algorithm.
- **IV** is a BLOB containing an initialization vector. This parameter is also optional.

**Decipher()** returns a BLOB, which contains the decrypted text.

### Supported encryption methods

Algorithm Name	Cipher Code	Supported Modes and (IV Length)	Number of Keys Key Length	Symmetry
BLOWFISH	1	ECB - NA CBC - 8 CFB - 8 OFB - 8	Minimum: 1 Maximum: 56 Recommended: 16	Symmetric
CAST	2	ECB - NA CBC - 8 CFB - 8 OFB - 8	Minimum: 5 Maximum: 16 Recommended: 8	Symmetric
DES	3	ECB - NA CBC - 8 CFB - 8 OFB - 8	Number of Keys: 1 Supported: 8 Recommended: 8	Symmetric
IDEA	4	ECB - NA CFB - 8 OFB - 8	Minimum: 1 Maximum: 16 Recommended: 16	Symmetric
RC2	5	ECB - NA CBC - 8 CFB - 8 OFB - 8	Minimum: 5 Maximum: 16 Recommended: 8	Symmetric
RC4	6	Not Applicable	Minimum: 1 Maximum: NR Recommended: 16	Symmetric
RC5	7	ECB - NA CBC - 8 CFB - 8 OFB - 8	Minimum: 1 Supported: 255 Recommended: 16	Symmetric
DES3	8	ECB3 - NA CBC3 - 8	Number of Keys: 2 Maximum: 16 or 24 Recommended: 24	Asymmetric
RSA	9	Not Applicable	Minimum: 48 Maximum: 2048 Recommended: 128	Asymmetric



## How do I Encrypt Data Inside a Table?

Even when you use passwords and restrict access, it is possible, using low-level tools, to view data directly from a disk. For this reason it is necessary to encrypt tables that contain very sensitive information.

eDeveloper makes this very easy. Just follow the steps below.

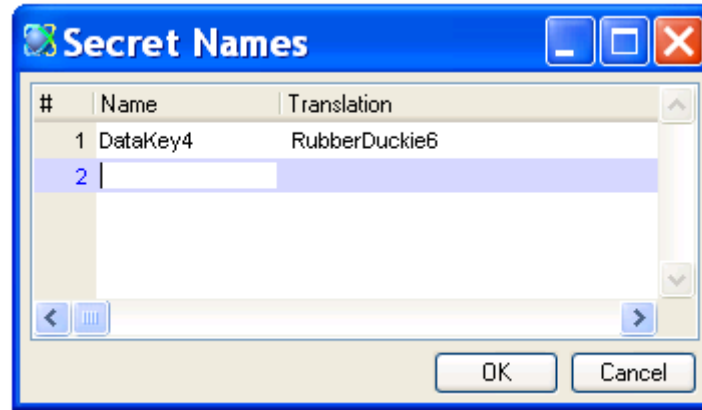
### Encrypting a database table

1. In the Data Repository, go to the table you want to encrypt.
2. Go to Data Source Properties (**Alt+Enter**, or **Edit->Properties**).
3. For the *Access key*, enter any string of characters. It is recommended though, that you use a Secret Name (Chapter 31, “How do I Hide Database Login Information?” on page 758).
4. Set *Encrypt table* to Yes.

Now, the data in the table will be encrypted. Only someone who has the same key can view the data.

**Note:** Not all DBMS's support encrypted data. If underlying the DBMS does not support this form of encryption, then the *Access Key* and *Encrypt table* fields will be greyed out.

## How do I Hide Database Login Information?

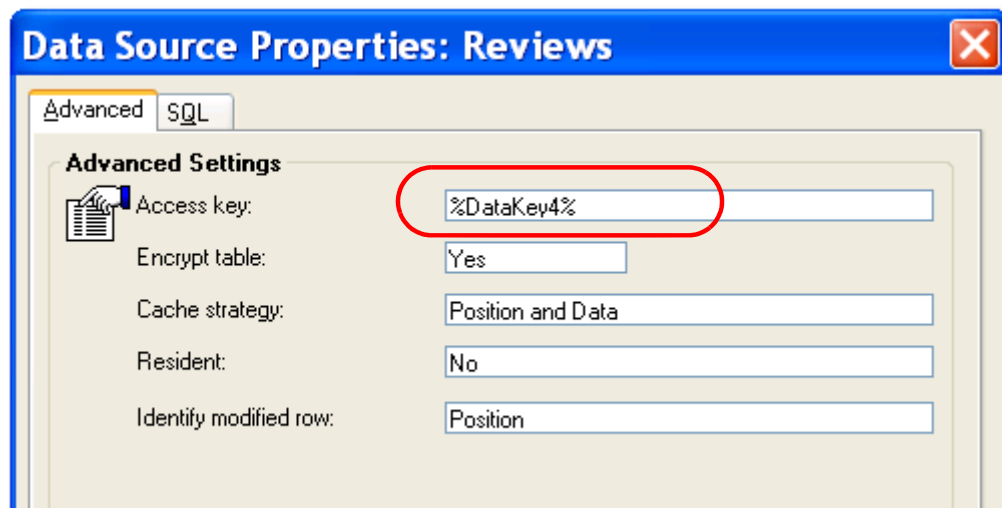


### Setting a Secret Name

1. Log on as Supervisor.
2. Go to **Options->Settings->Secret name**.
3. Enter a secret name. The “Name” column works like a Logical Name, with the exception that it cannot be viewed except by someone with Supervisor access.

The secret name is stored with the userid’s and passwords, in an encrypted format in the Security file.

### Using a Secret Name



Wherever you want to use the *Secret name*, enter the key as you would a *Logical Name*. It will be translated at runtime. Note that secret names can only be used in specific fields in eDeveloper, such as project access keys, user password fields, Server/DB properties, and data source access keys.

This means that every installation of your application can have their own set of secret names. The developer can code a value for the logical name, but will not know what the actual value is at runtime; the developer does not need to know the password for a database, for instance.

## How do I Declare Administrator Rights in an Application?

When you are working with an eDeveloper application, the user “Supervisor” has special privileges. When you log in as Supervisor, you have authority to create and modify other user accounts.

When you first create an eDeveloper application, the Supervisor already exists in the list of Users. In order to log in as Supervisor, you need to:



1. Select **File->Close application**, if an application is open.
2. Select **Options->Logon**. A login dialog will appear.
3. For *User ID*, type supervisor
4. Leave the *Password* blank
5. Press **OK**.

Now, logged in as Supervisor, you can set up the rest of the User IDs. It is recommended that you set a password for the “supervisor” user.

### Deleting the Security file

Whenever the security file gets deleted, or can't be found for some other reason, you can always log in as supervisor by using a blank password. This gives you access to all the Groups, Rights, etc. in the application. If you do not want this option to be available, then you need to use other levels of security to secure the application itself, such as:

- Non-public Rights
- The Super Right key in the Application Properties
- Secret names



## How do I Limit Execution of Specific Programs?



You can add a Right to each program to prevent it from being executed except by people who have a particular Right.

For instance, suppose we have a program to print paychecks. Very few people get to access this program. We can put a Right on the menu, so this option won't even show up, but we want to make sure that even most programmers can't get to it. Here is how we would assign the Right to this program:

1. Go to the program you want to protect.
2. Select **Options->Authorize**. The *Rights Assignment* box will appear.
3. Zoom from the *Execute/APG* field to select the right you want to use.

Now, anyone who does not have this Right cannot execute this program.

While this is a good backup form of protection, it's best to make it so sensitive programs don't even show up on the menu unless the person is authorized to run them. You can find out how to do that in Chapter 31, "How do I Customize the Menu According to the User Logged On?" on page 763.

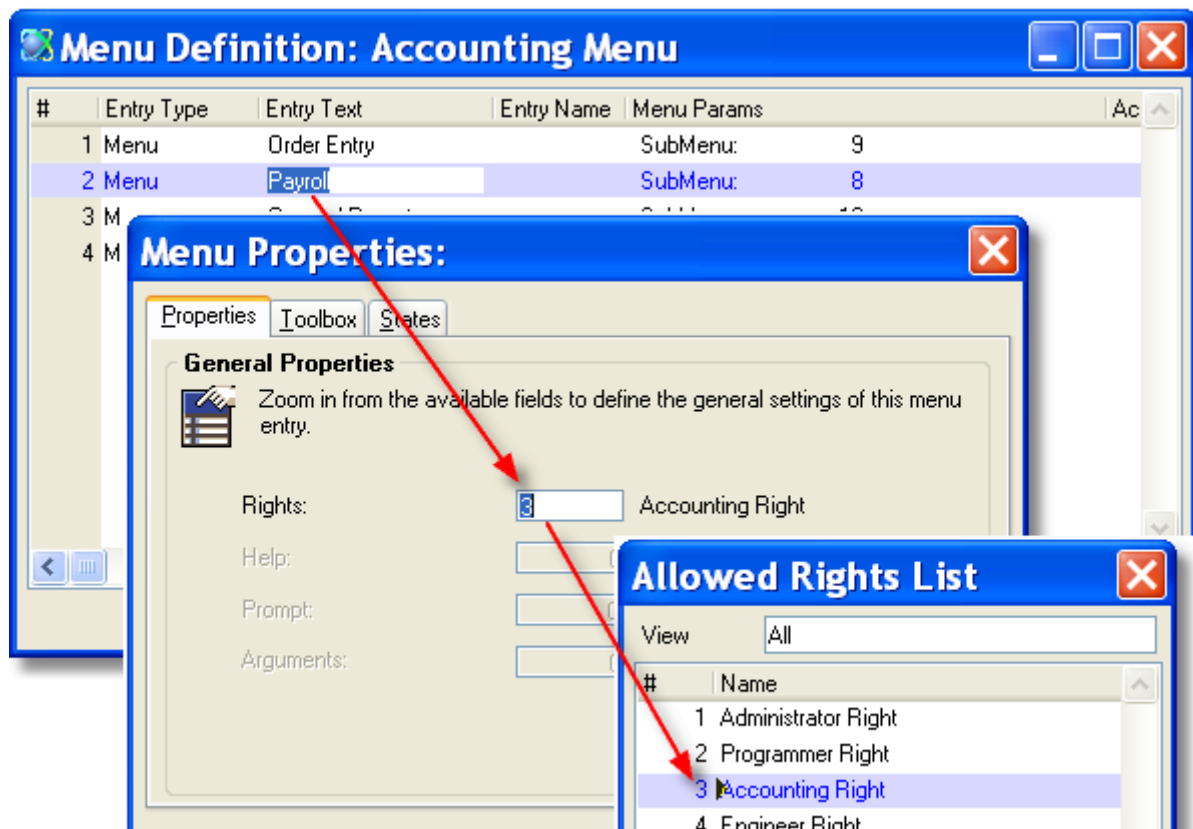
**Hint:** Since obviously the programmers will have to run this program to test it, it is a good idea to have two different Security files, one for the test environment and one for production. The programmers can then run sensitive programs, but only with test data.

## How do I Customize the Menu According to the User Logged On?

If the users only see the menu items that they can actually use to do their jobs, the menu system will be cleaner-looking and easier to use. Also, from a security standpoint, it's probably better that some features of the system aren't visible to everyone.

There are very robust features for customizing the menu in any way you wish, which are covered in Chapter 27, "How do I Hide/Reveal a Menu Entry?" on page 695. However, eDeveloper has a special security feature that is easier to use than hiding menu entries programmatically.

### Securing menu items



1. Go to the *Menu* repository (**Shift+F6**).
2. Go to the menu you want to secure. It's generally best to secure the menu at the highest level possible, by job function, but you can do this with any menu entry.
3. Press **Alt+Enter** to access the *Menu Properties*.
4. The cursor will be located on the *Properties* tab, on the *Rights* field. **Zoom** to select the Right you want to use.

Now, the menu entry will not appear unless the user has the assigned Right. In our example, the Payroll menu item will only appear for the users who have the Accounting Right.

**Note:** Your eDeveloper User ID must have access to a Right before you can use it in a program.



## How do I Limit Functionality According to the User Logged On?

You can use the eDeveloper security system to change functionality at any level within a program, by using the **Rights()** function in an expression. Here we will go through how to enter this function, and show a couple of useful examples.

### Creating a Rights() Expression

The syntax of the **Rights()** function is:

**Rights('a right')**, which returns 'TRUE'LOG if the person has that right. So, for example, you could just type in:

```
Rights( 'Accounting Right 'RIGHT)
```

and it would return 'TRUE'LOG for an authorized person.

However, that's a lot of typing, so here is the easy way to enter it:

1. Go to the Expressions repository (**Ctrl+E**).
2. Press **F4** to open up a line.
3. Type ri, then press **Ctrl+Space**. This will bring up a list of auto-complete choices: choose *Rights*.
4. Now your Expression will be:  

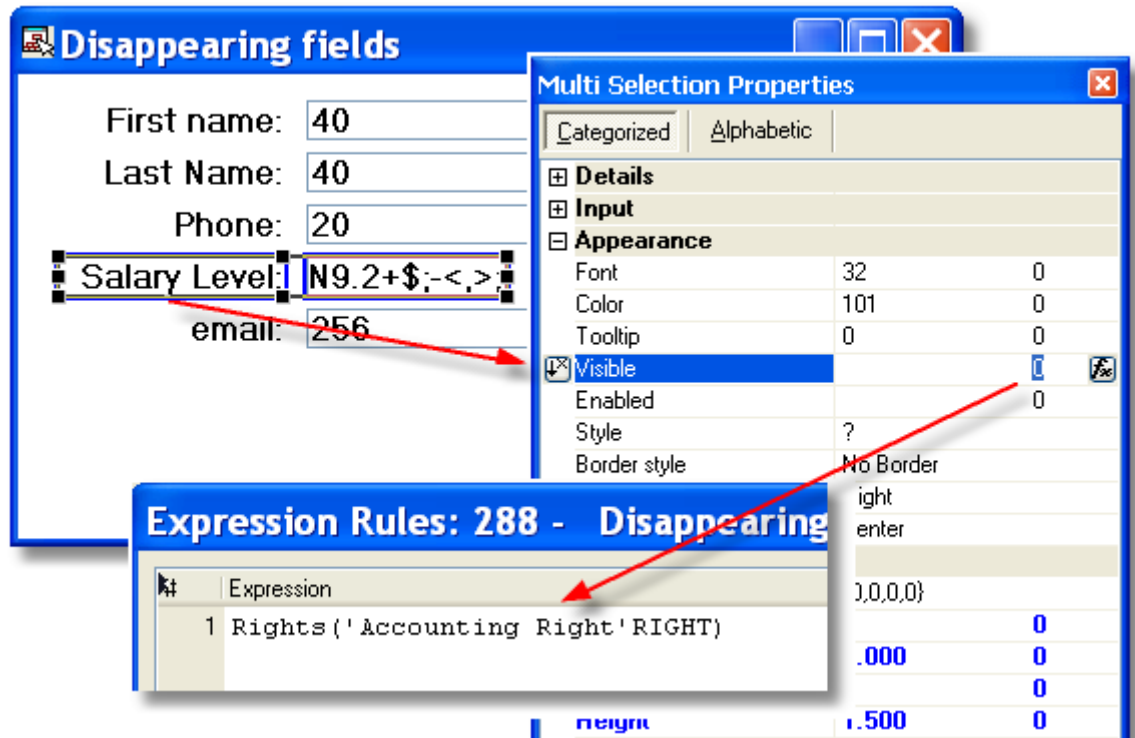
```
Rights(
```
5. From the Right-click menu, select *Rights*. Now you will see a list of all the Rights you have access to. Select the one you want (in our example, "Accounting Right"). This will automatically bring the text name of the Right into your Expression, so it will be:  

```
Rights ( 'Accounting Right 'RIGHT
```
6. Now just type the closing paren and you are done.

You can use Boolean logic to add two or more Rights, or add other conditions to the expression.

Now that we have our Expression, let's see how to use it to change the program functionality.

## Making a control disappear for unauthorized users



One of the most common things you need to do with security levels is to make data disappear from a screen. In this example we will make the “Salary level” field disappear. The method also works for push buttons though, or any other control.

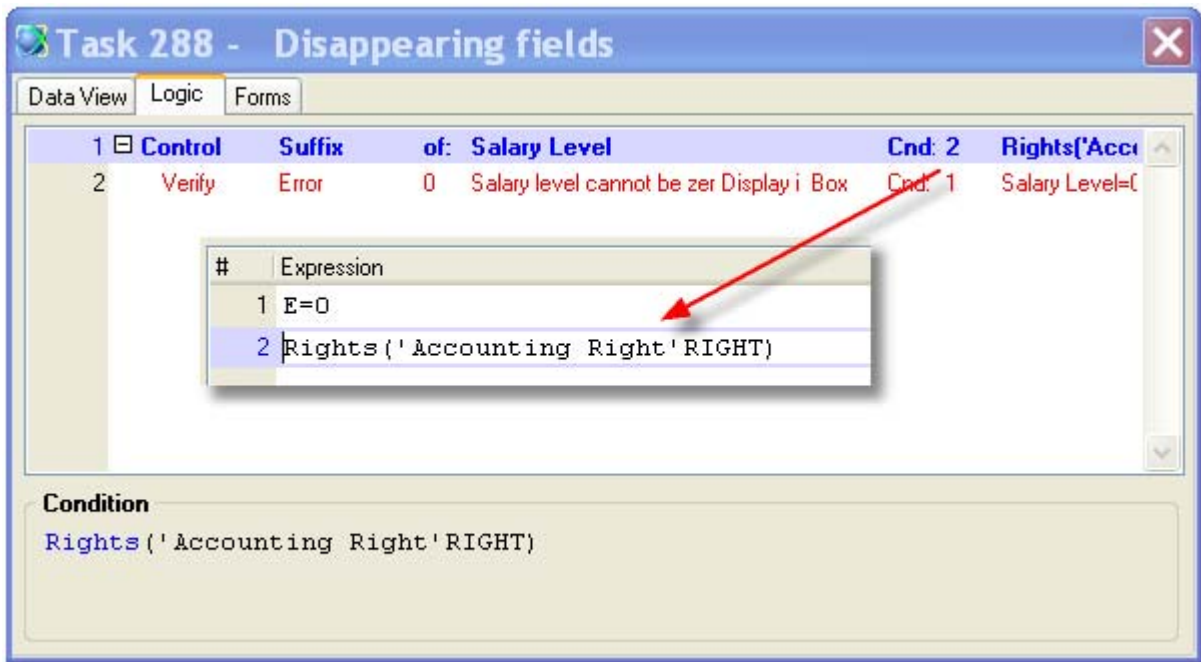
1. Select the control you want to work with. If you want to work with several controls at once, press down the Ctrl key and click on the ones you want one by one. Here we selected the “Salary Level” field and its field prompt.
2. Press **Alt+Enter** to bring up the *Properties* for the control(s).
3. Go to the *Visible* property, to the rightmost field.
4. Zoom to select (or create) your **Rights()** expression.

Now, the fields will be invisible for non-authorized users.

If you want the field to be disabled, but not fully invisible, use the **Rights()** expression on the *Enabled* property instead.

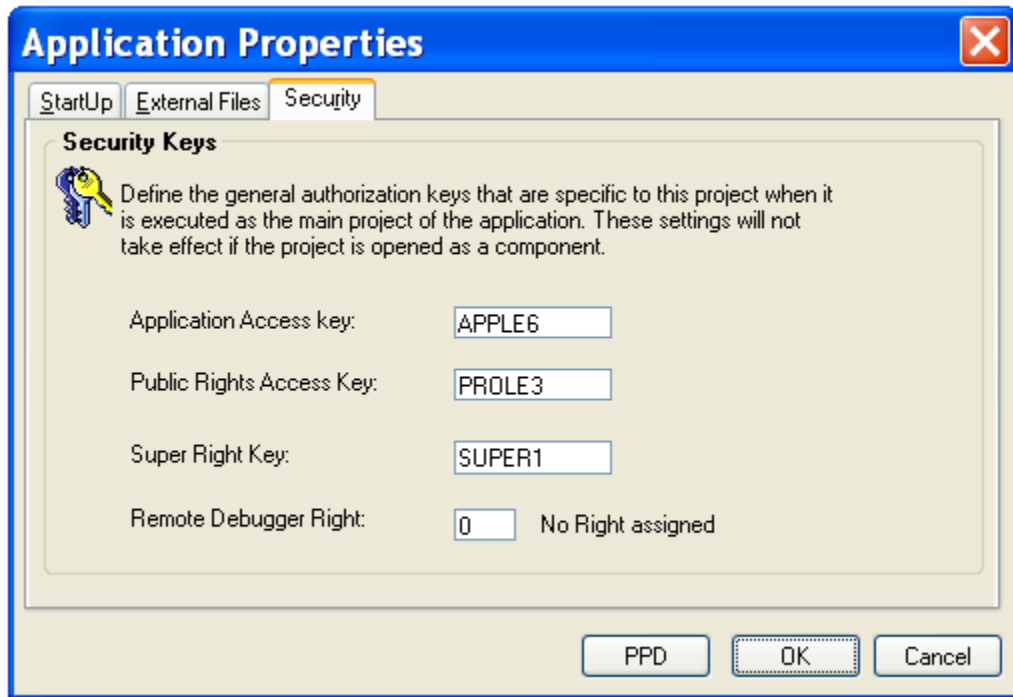
**Note:** When you make a field disappear, make sure you also turn off any validation logic for that field. For instance, suppose you have a field “salary level” that doesn’t show up for most users. You also have a Verify error operation that says “Salary level cannot be blank”. If the user can’t fix the field, they can’t escape the error message. So the error logic needs to have the same Rights() logic as the field. This isn’t an issue though, as long as the verification logic is in the Control logic unit, because that will be disabled when the control on the form is disabled.

## Preventing logic from executing for unauthorized users



To prevent logic from executing based on a Rights() expression, just use the expression as a Condition. Here, for example, we disabled the Control Suffix, which isn't technically necessary if you have the expression on the Control (but it might be good for documentation purposes). You can disable any operation or logic unit.

## How do I Restrict Access to the Entire Application?



You can restrict access to the entire application by using the *Security* features in **Application Properties** (**Ctrl+Shift+P**). There are four security features here, which are explained fully in the eDeveloper Help, but here is a summary.

Property	What it does	Effect
Application Access Key	Allows the user to access the application	If the user does not have this key, the user will get a message "Access denied, runtime engine failed to open the application".
Public Rights Access Key	Allows the Supervisor to select Rights to give to users.	Normally, the Supervisor can always select Rights, even if the Supervisor doesn't have that Right. If a Public Rights Access key is assigned, however, the Supervisor must first type in the key as a Supervisor Right.
Super Right Key	Allows the user to access in effect have all Rights.	This overrides all the individual Rights. It is good for testing, because it allows the programmer to run all programs without needing to be assigned each Right individually.
Remote Debugger Right	Allows the user to run the Remote Debugger.	

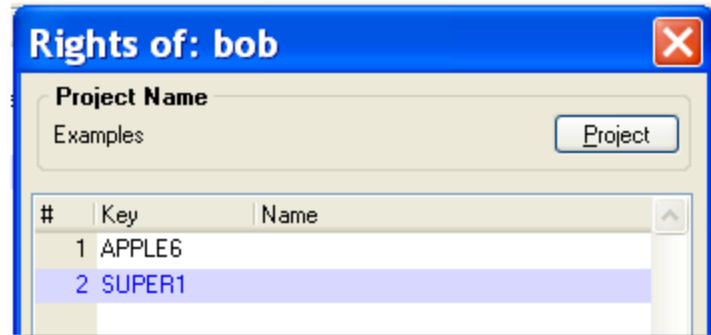
The *Remote Debugger Right* is set by zooming to the Rights repository. The others are plain text you type in, as you would set a password. Once they are entered into the field, no one can see them who doesn't have that key, so be sure you keep them safe somewhere for future reference.

### Using Security keys

To assign a security key to a user, do the following:

1. Close your application, if it is open.
2. Log in as Supervisor.
3. Go to the user you want to authorize.
4. Zoom from the Rights column. A list of Rights will appear.
5. Press F4 to open up a line. Type in the key, exactly as it was entered in Application properties.

Now, that user will have that particular key assigned.



## How do I Implement Roles?

Typically, when one is implementing a security system in eDeveloper, rights are granted according to the user's job function, which corresponds to an eDeveloper *Group*. That is, a user who is an Accountant will have a different set of menus and screens than a person who is an Engineer. Other rights, however, may only be granted to certain individuals, such as the ability to print paychecks or fix timecards.

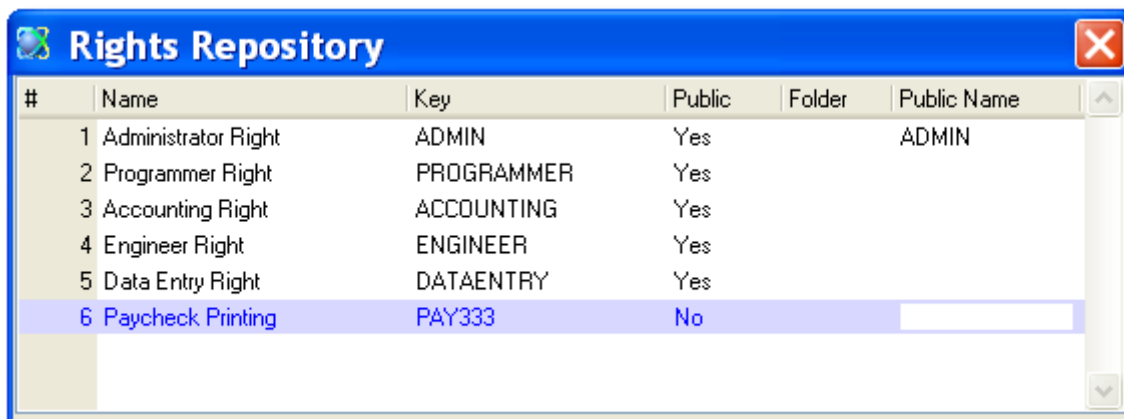
Some care must be taken when designing the system, because creating too many very specific rights makes the system unwieldy, but making it too general might not give enough flexibility.

In any case, the steps for setting up your user roles are as follows:

1. Set up your *Rights*
2. Set up your *Groups*
3. Set up your *Users*

Let's look at each of these steps.

### 1. Set up your Rights



The screenshot shows a window titled "Rights Repository" with a table of rights. The table has columns for #, Name, Key, Public, Folder, and Public Name. The rows are as follows:

#	Name	Key	Public	Folder	Public Name
1	Administrator Right	ADMIN	Yes		ADMIN
2	Programmer Right	PROGRAMMER	Yes		
3	Accounting Right	ACCOUNTING	Yes		
4	Engineer Right	ENGINEER	Yes		
5	Data Entry Right	DATAENTRY	Yes		
6	Paycheck Printing	PAY333	No		

Rights are set up within each application, in the *Rights repository*. The Rights can have Public Names and be used as part of a Component.

To enter a Right:

1. Press **F4** to open up a line.
2. Type in the *Name*. This can be any text you like. It will be what shows up when you access the Rights list to select a Right for an expression or authorization entry.
3. Type in the *Key*.
4. Select Public=No if required (See below for more explanation on this).
5. Give the Right a Public Name if needed.

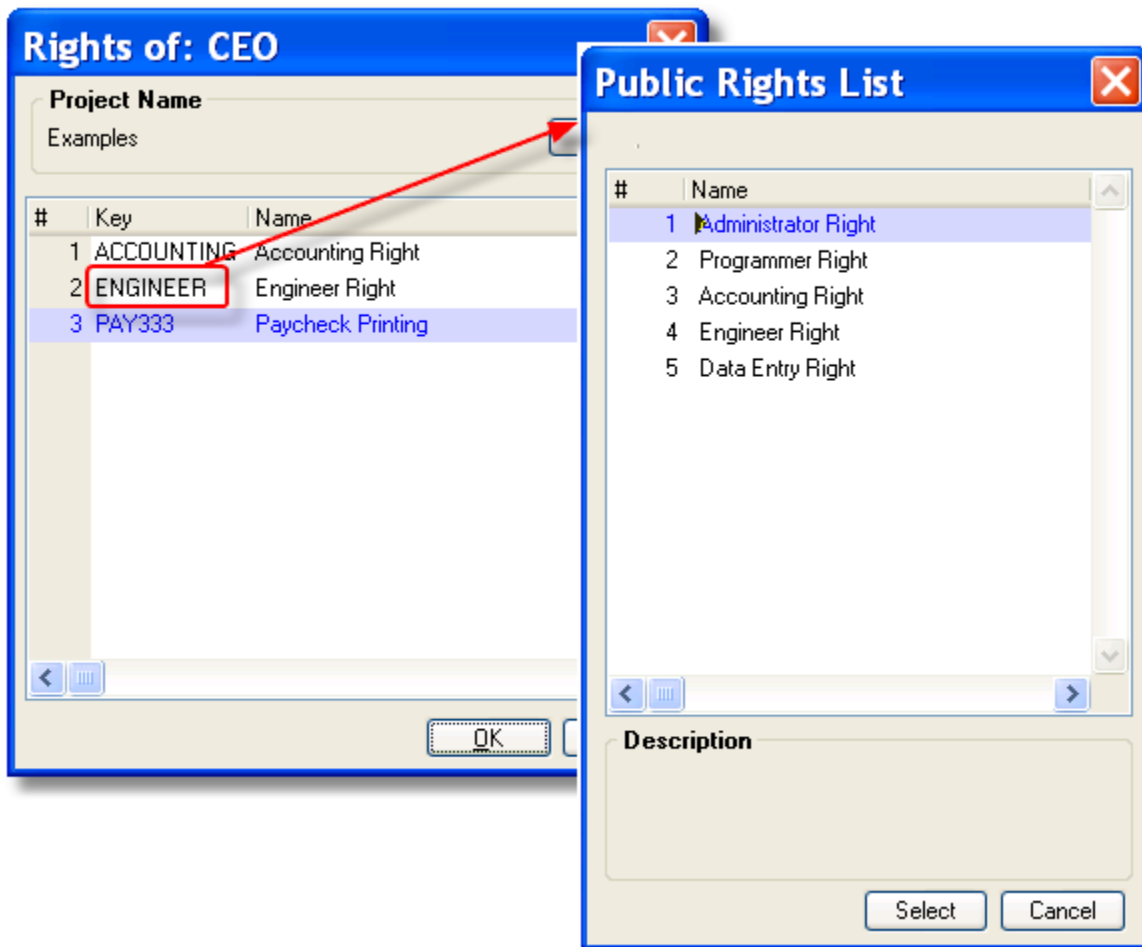
### Public rights

If you enter **No** in the *Public* column, then the Right will be effectively hidden from anyone who doesn't already have that Right. In our example, Paycheck Printing is a non-public right. If the Supervisor knows the key -- PAY333 -- then that person can enter that to give themselves or someone else the right to print paychecks.

This is necessary because eDeveloper allows a default Supervisor to log in and set up the initial rights, when no Rights file exists. If you want to keep certain items secret from even the Supervisor, then use a non-public Right.

However, you should use this feature with care, because if you forget your login and the key, you will be stuck.

## 2. Set up your Groups



Groups are not set up within your application. They are stored in the Security file. This file can be shared between several eDeveloper applications. You can check the location of the security file by looking in

**Options->Settings->Environment->Security file**, but you can't edit the file directly because it is encrypted. To set up your Groups, therefore, you need to use eDeveloper's tools.

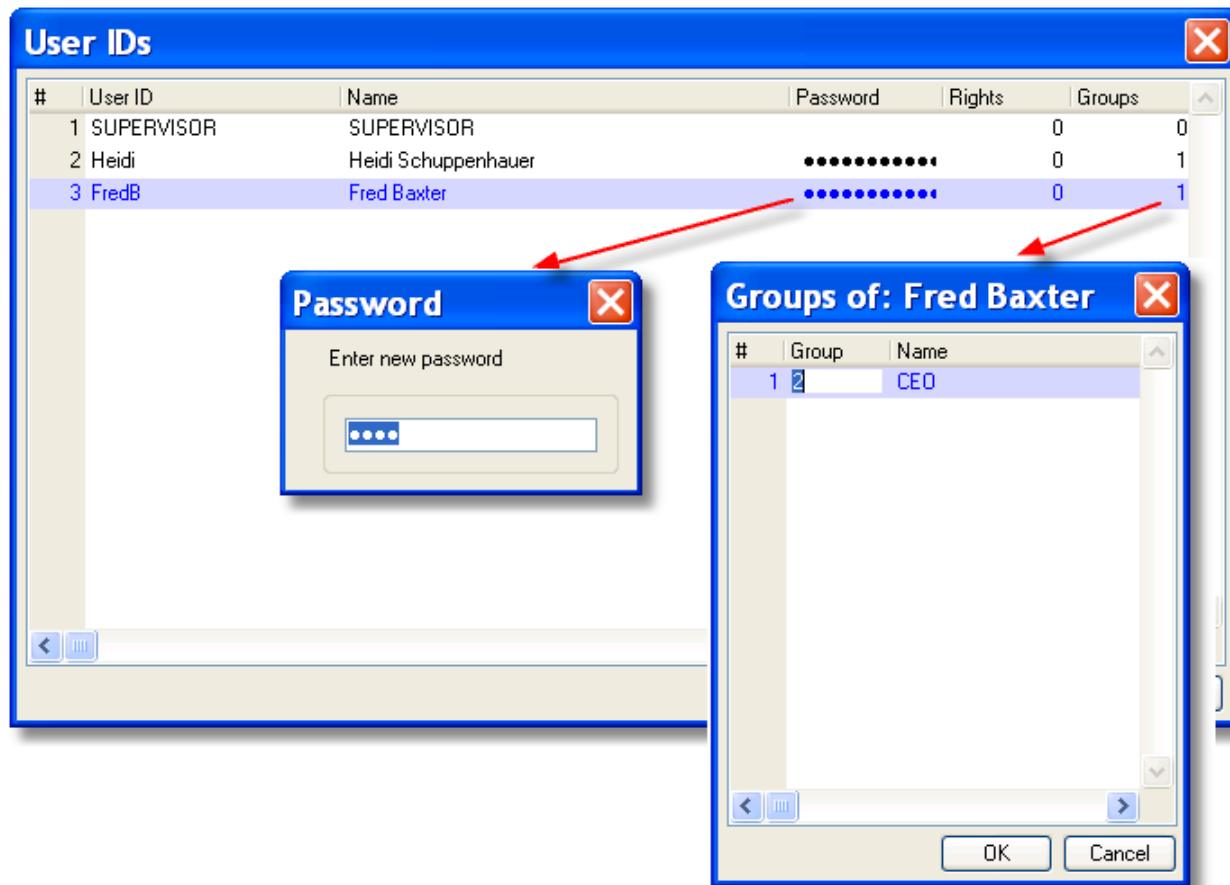
**Prerequisite:** First you need to be sure you are logged in as Supervisor. See Chapter 31, "How do I Declare Administrator Rights in an Application?" on page 760 for more information about that.

1. Select **File->Close Project**. Now you will be at the eDeveloper Startup screen.
2. Choose **Options->Settings->User Groups**. A list of User Groups will appear. There will always be one group, by default, the SUPERVISOR GROUP.
3. Press **F4 (Edit->Create Line)** to add a line.
4. Name the group whatever you like. We'll call ours "CEO".
5. Tab over to the Rights column, then zoom. You will see an empty list, because no Rights exist yet for this group. For each Right you want to add:
  - Press **F4 (Edit->Create Line)** to add a line.
  - Type in the Key, or zoom to select it from a list.
  - If the Right is a non-Public Right, then you can't select it from a list; you have to type it in. In our example we typed in our non-Public Right, PAY333.

Continue until you have created the Groups you think you will need (you can always add more later).



### 3. Set up your Users



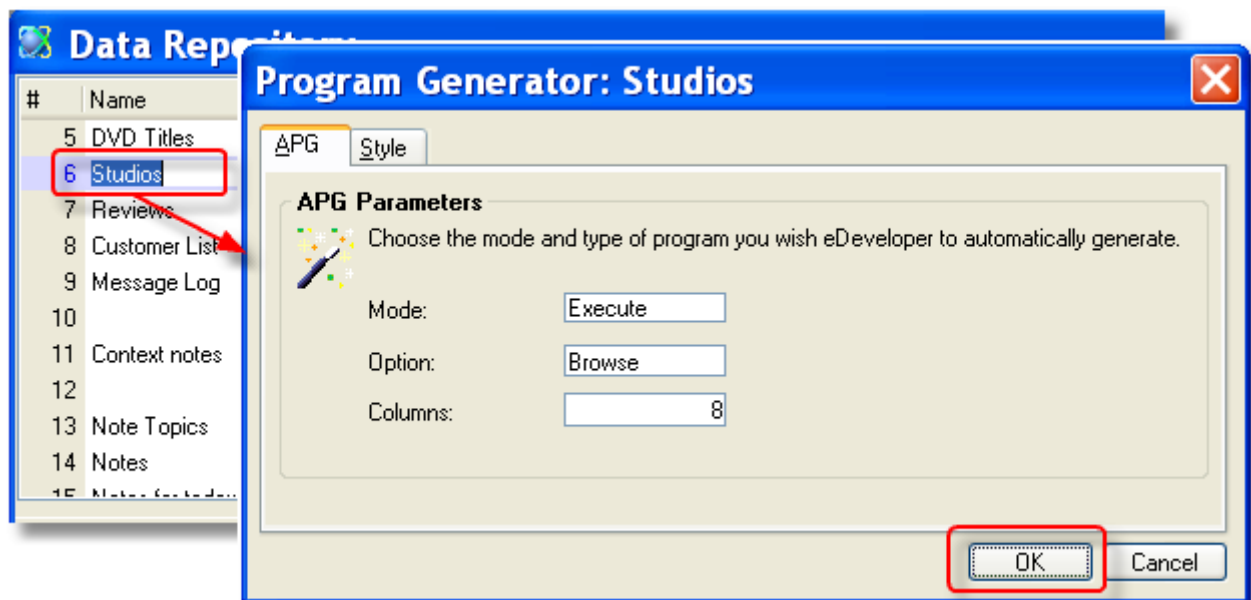
Now, while you still have your application closed, you can set up your Users.

1. Choose **Options->Settings->UserIDs**. A list of User IDs will appear. There will always be one User, by default, the SUPERVISOR.
2. Press **F4 (Edit->Create Line)** to add a line.
3. In the User ID column, type in the user's login id. Since this can be passed in from the operating system, using the network login is a good idea.
4. In the Name column, type the user's name. This field isn't used by eDeveloper, but you can use it to display the user's full name when you need to.
5. Zoom from the Password column to create the login password (you won't need a password if you are logging in via the network).
6. Zoom from the Groups column to assign the user to one or more Groups.
7. For each Group you want to add for this user:
  - Press **F4 (Edit->Create Line)** to add a line.
  - Zoom to select the group from a list.

If you planned your Groups carefully, you probably won't need to add individual Rights to one user. But if you do, you can add them from the Rights column.

## Chapter 32: Utilities

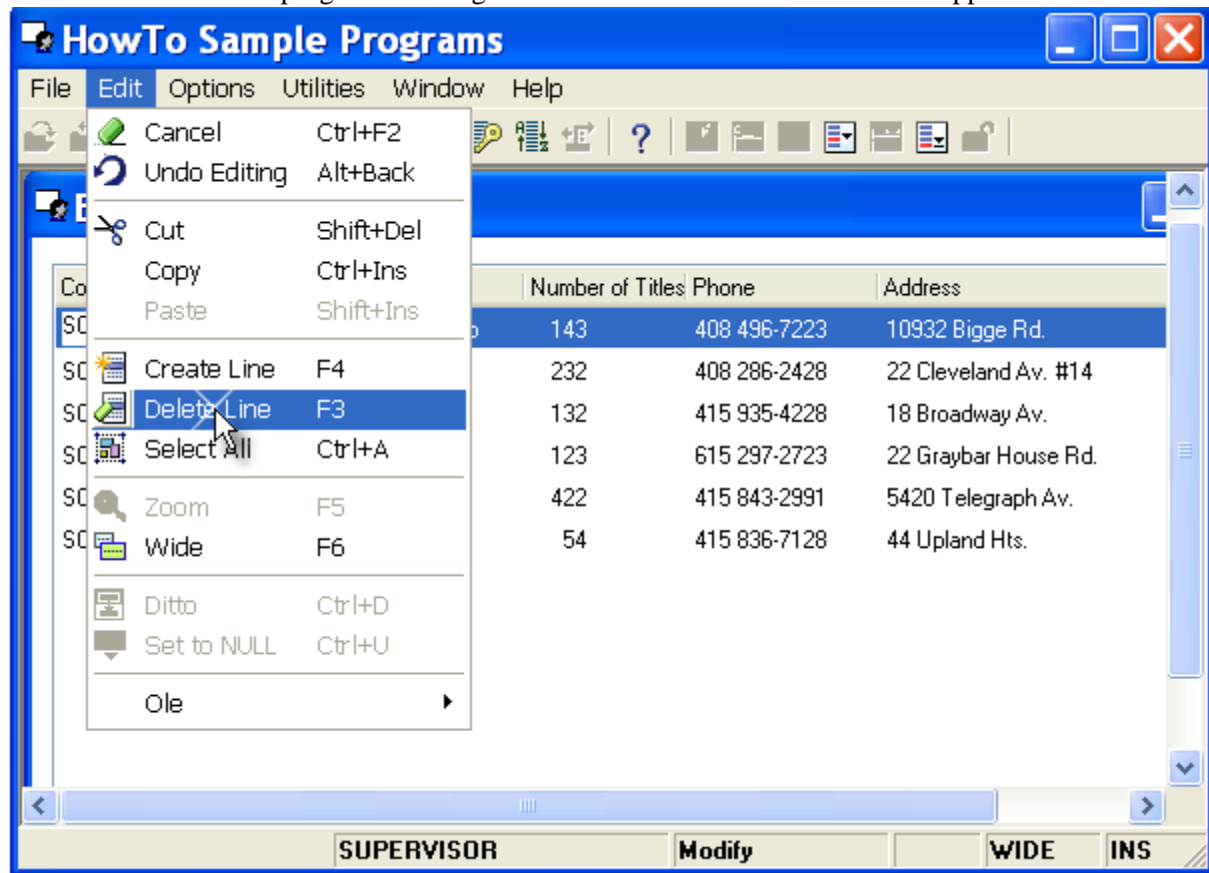
### How do I Browse a Data Source?



When you are developing an application, it's often useful to be able to quickly take a look at the data you are working with. Here is how to do it:

1. Go to the *Data Source Repository* (**Shift+F2**).
2. Go to the *Data Source* you want to browse.
3. Press **Ctrl+G** (**Options->Generate Program**). The Program Generator dialog will appear.

4. Press **OK**. A browse program showing all the records in the Data source will appear.



This only takes a couple of seconds, and from the resulting browser you can use the runtime Locate and Range utilities to easily find specific records. You can also add, delete, and edit records, and other options, which you can see on the pulldown menu.

There are more options in the Program generator you can use to make your browser even more useful. These are explained in Chapter 32, “How do I Create Simple Browse Program for a Data Source?” on page 777.

**Hint:** Before your browse program starts, Task prefix of the Main program will execute. If you have code in Task prefix that will be irritating to you while you are trying to run browser programs, such as a user timecard, you can disable it by using the RunMode() function. See Chapter 19, “How do I Skip Initialization Code?” on page 489 for details).

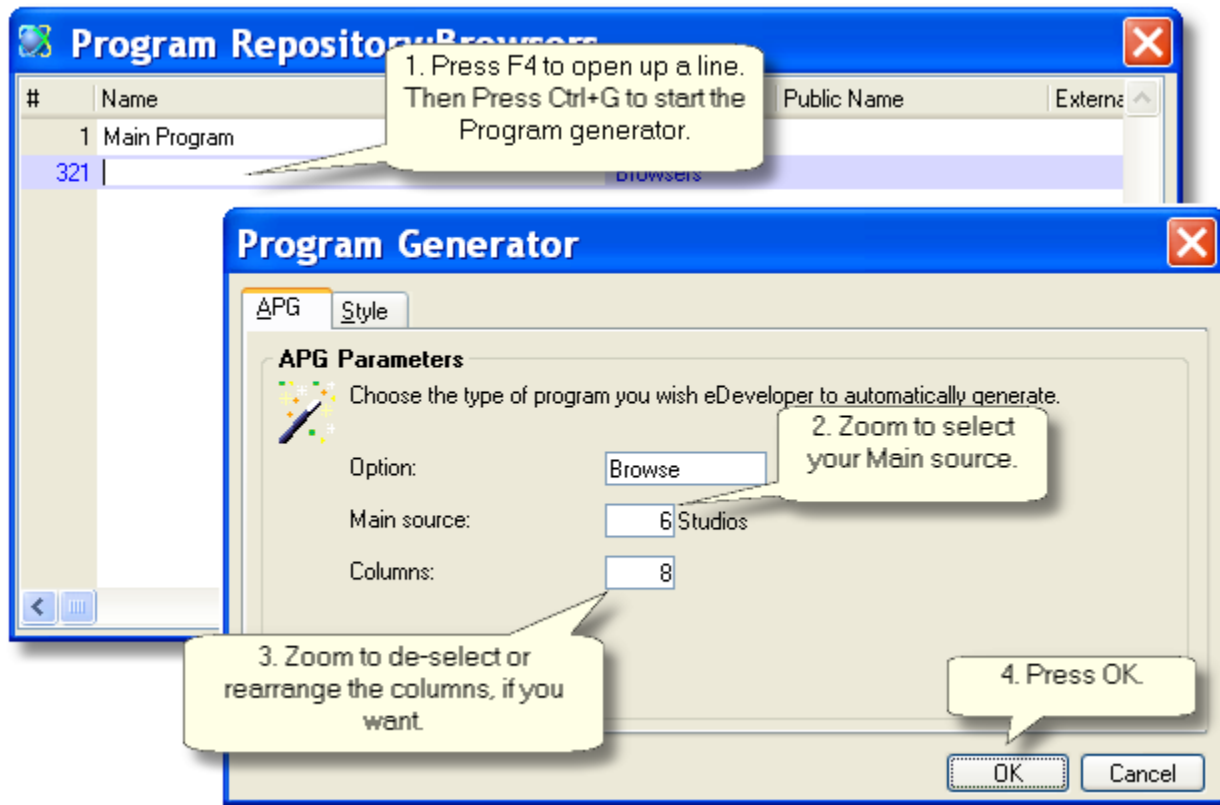
**Note:** If you want to create a permanent program to browse the Data source, the process is very similar. See Chapter 32, “How do I Create Simple Browse Program for a Data Source?” on page 777.

## How do I Create Simple Browse Program for a Data Source?

It is very easy to create a simple browse program for any Data source. These simple browse programs are useful for debugging and they are actually quite powerful, as they have the ability to add, delete, and modify records. You can also use them as the base programs for running the eDeveloper report generator or exporting data into other formats, such as XML. They can also be used as a starting point to create more complex programs.

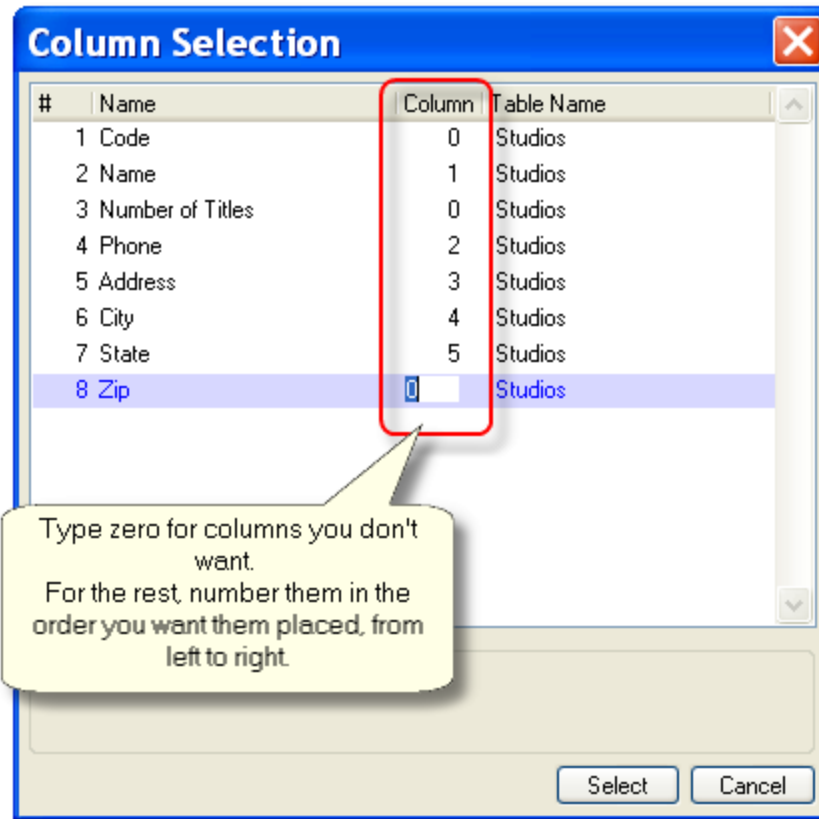
Let's see how to do it.

### Creating a browse program



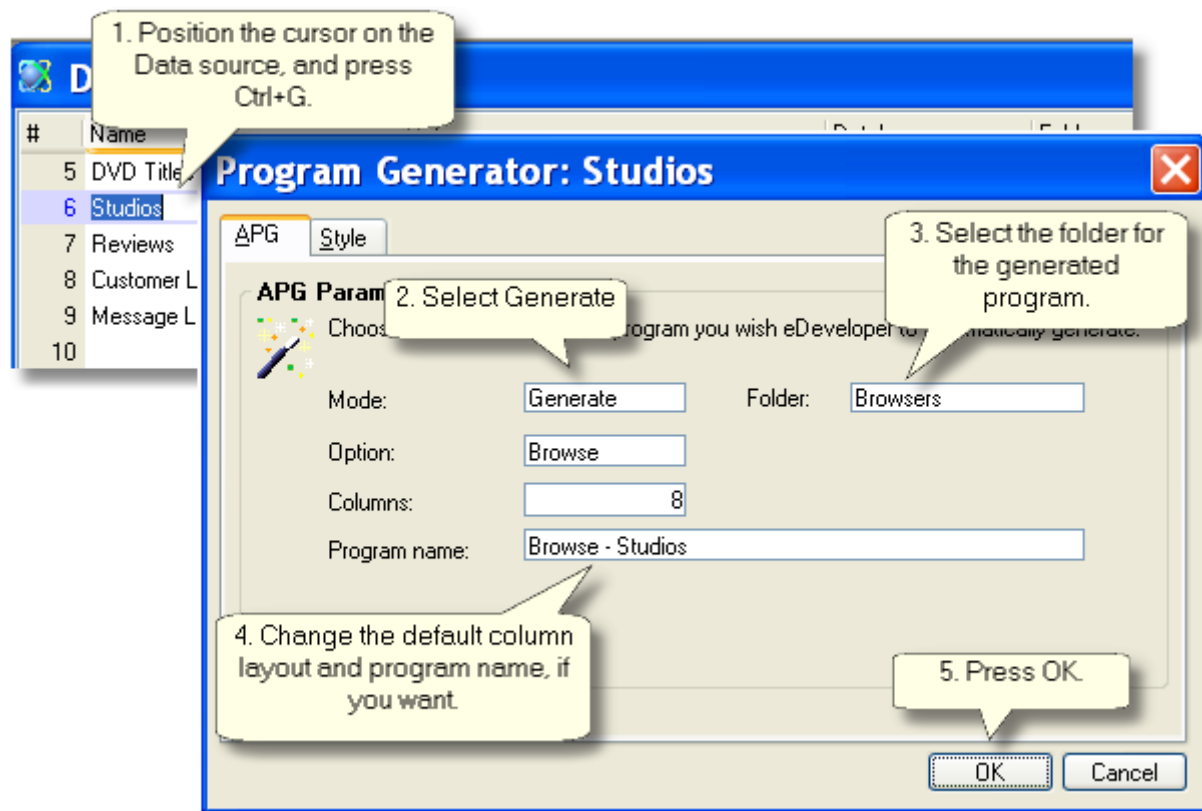
1. Press **F4** (**Edit->Create Line**) to open up a line in the Program repository. Then press **Ctrl+G** (**Options->Generate program**) to bring up the *Program generator*.

2. From the *Main source* field, zoom (F5, or double-click) to select your Main source from a list of Data sources.



3. From the *Columns* field, zoom to de-select some of the columns or to reorder the columns. By default, all the columns are selected, in the order they are listed in the Data source.
4. Press OK. A browse program will be generated.

Alternatively, you can also create programs in the Program repository while parked on the program in the Data source repository. This works similarly to the case above, except you don't need to select the data source.



The Style tab allows you to choose whether your browser program will show a table of records or only one record at a time, whether it is 3D or 2D, the screen size, and whether or not to use a Model.

After you press OK, a program will be generated. You can run the program by pressing F7, or go in to edit it.

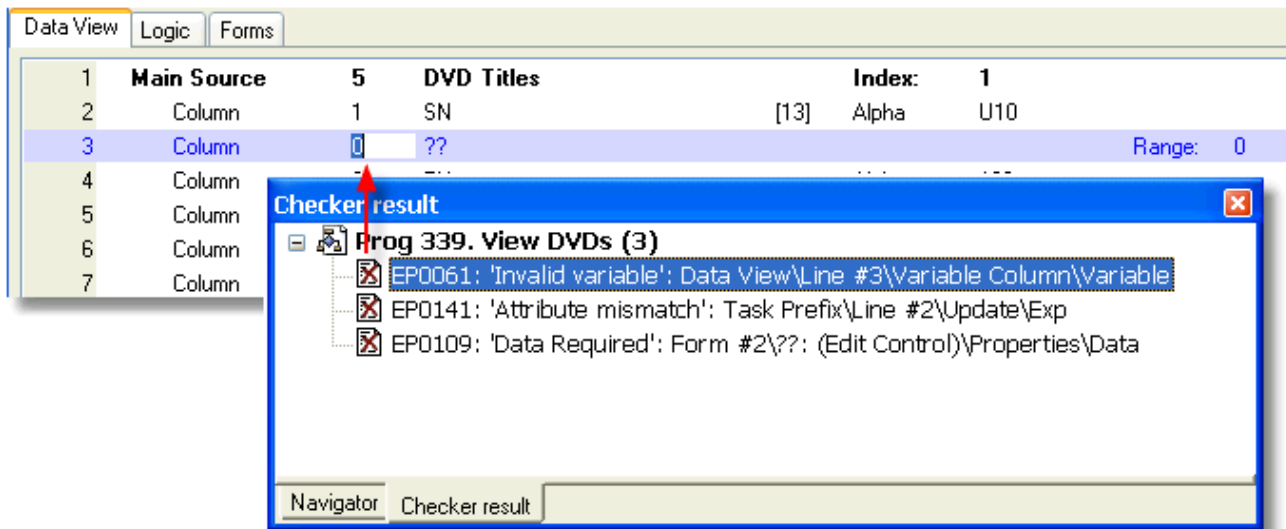
## How do I Syntax Check a Program?

Before you run a program in any language, you should check it for correct syntax. Since eDeveloper does not force you to compile a program before running it, the syntax check is optional, but if the program has errors, it will run incorrectly.

It only takes a few seconds to syntax check an eDeveloper program. When the syntax check is completed, you will be presented with a list of errors (if there are any). Clicking on an error will cause you to jump to that error so you can fix it.

You can also syntax check a list of programs all at once. This is a good thing to do before putting a new application into production, to ensure no errors have crept in.

### Syntax checking one program



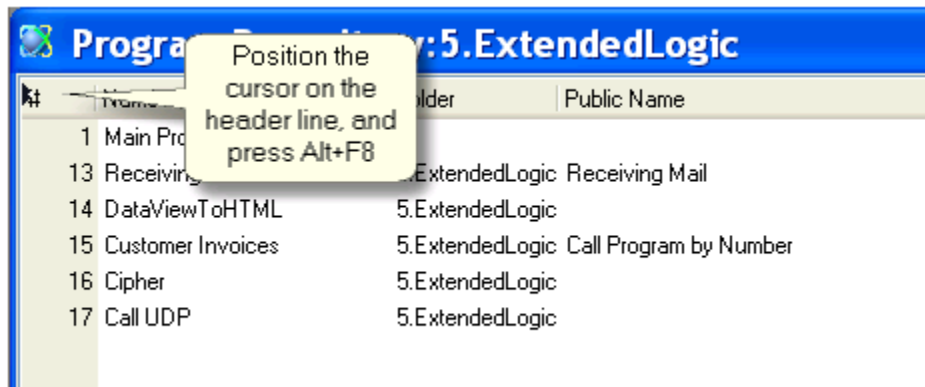
1. Position the cursor on the program you want to check.
2. Press **F8** (**Options->Check syntax**). You will see some screens flash, and then one of two things will happen:
  - You will see “Program is ok” on the status line, or
  - You will see “Check Syntax completed - Please refer to the Checker result pane” on the status line, and the Checker result pane will have some error messages in it. (If you don’t see the Checker result pane, select **View->Checker Result** (**Alt+F3**) to make it visible).

If you had errors, you can work through the list of errors using the Checker result list. See Chapter 32, “How do I Use the Checker Results?” on page 786 for details on how to do this.

You can also customize which errors appear and in which order. See Chapter 17, “How do I Control the Displayed Checker Messages?” on page 450.

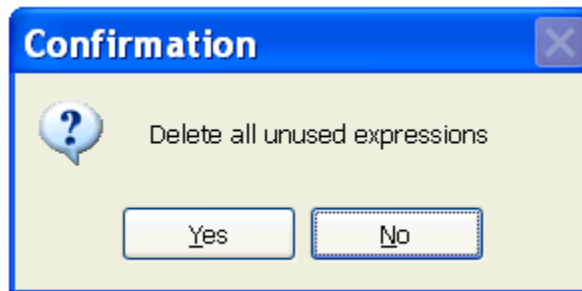


## Syntax checking many programs at once



You can check a series of programs at one time. The error messages will all be grouped in the Checker result pane, and you can work through the list the same way you would when working with only one program.

1. Position the cursor on the first program to check, or on the header line to check all the programs in that section. If you are working with only one folder, only that folder will be checked.
2. Press **Alt+F8** (**Options->Check to end**).

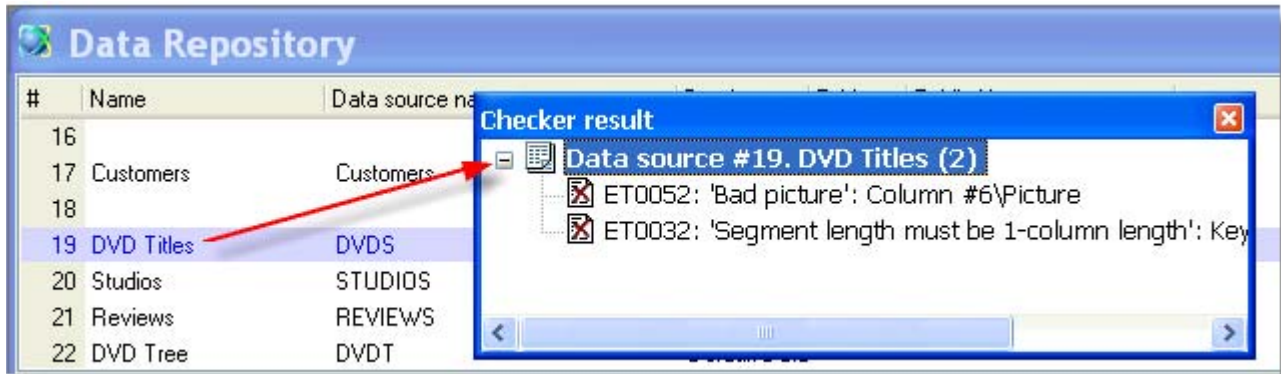


3. You will be prompted with a confirmation box "Delete all unused expressions". If you click on Yes, then expressions that may exist, but are not referenced, will be automatically deleted without prompting you first.
4. The syntax checker will run through all the programs in the list, and the errors will be listed in the Checker result pane.

## How do I Validate a Data Source Structure?

When you create a new Data source, it is a good idea to check that it has a valid structure before using it. This takes only a few seconds.

### Syntax checking one Data source

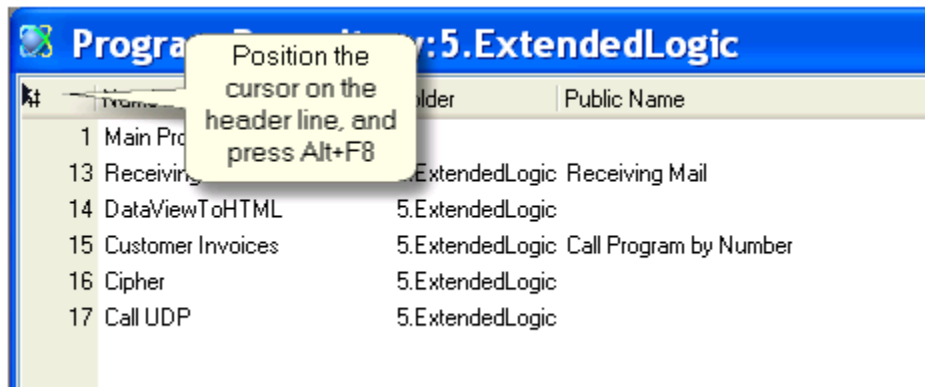


1. Position the cursor on the Data source you want to check.
2. Press **F8** (**Options->Check syntax**). You will see some screens flash, and then one of two things will happen:
  - You will see “Data source is ok” on the status line, or
  - You will see “Check Syntax completed - Please refer to the Checker result pane” on the status line, and the Checker result pane will have some error messages in it. (If you don’t see the Checker result pane, select **View->Checker Result** (**Alt+F3**) to make it visible).

If you had errors, you can work through the list of errors using the Checker result list. See Chapter 32, “How do I Use the Checker Results?” on page 786 for details on how to do this.

You can also customize which errors appear and in which order. See Chapter 17, “How do I Control the Displayed Checker Messages?” on page 450.

## Syntax checking many Data sources at once



You can check a series of Data sources at one time. The error messages will all be grouped in the Checker result pane, and you can work through the list the same way you would when working with only one Data source.

1. Position the cursor on the first Data source to check, or on the header line to check all the Data sources in that section. If you are working with only one folder, only that folder will be checked.
2. Press **Alt+F8** (**Options->Check to end**).
3. The syntax checker will run through all the Data sources in the list, and the errors will be listed in the Checker result pane.

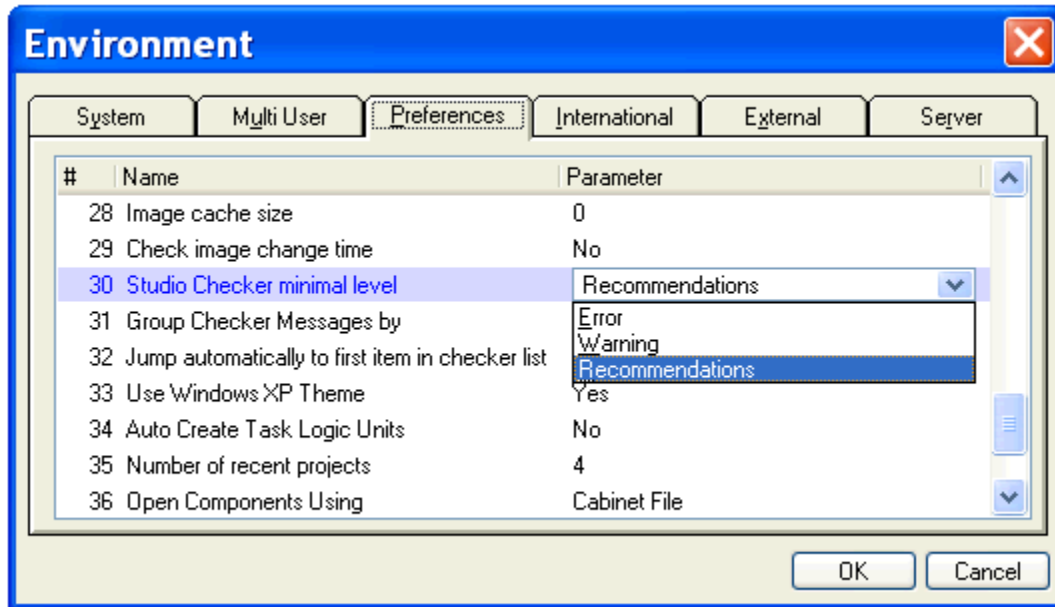
## How do I Filter the Messages Shown by the Checker?

You can choose which messages will show when you use the Checker. This is done in two ways:

- By setting the minimal level of errors shown in the Checker.
- By setting the Level of each individual message

Here is how to do it.

### Set the Checker Minimal Level

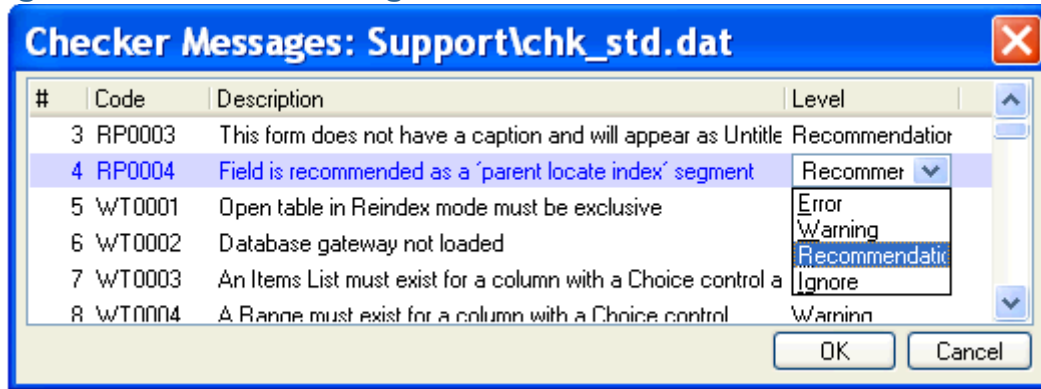


1. Go to **Options->Settings->Environment->Preferences->Studio Checker minimal level**.
2. Set this as follows:

Setting	Messages that show
Error	Errors
Warnings	Errors Warnings
Recommendations	Errors Warnings Recommendations

In other words, if you want all the possible messages to show, use the Recommendation level.

## Setting the Level of a message



1. Go to **Options->Settings->Checker messages**.
2. Set the Level to Recommendation, Warning, Error, or Ignore.

After customizing your error list and changing the minimal level, you will only see the messages that you want to see in the Checker.

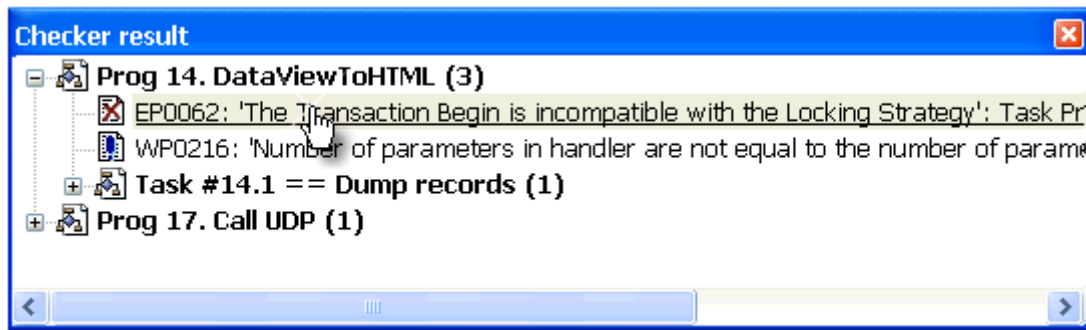
## How do I Use the Checker Results?

After you have completed a syntax check, you may have a list of results listed in the Checker Result pane. Now, you need to handle each of those messages. There are two basic ways to do this:

- Click on each message in the *Checker result pane*, or
- Use **Ctrl+F8** to move from message to message.

Let's see how to use either method.

### Using the Checker Results List



1. Position the cursor on the error you want to fix.
2. Double click.
3. The Checker will jump to the code in error.
4. Continue until you have fixed all the errors.

The Checker results may be grouped by object, as shown here, where each Task or Data source has its own node on the tree, or they may be grouped by error type, or both, depending on how the Checker is configured (See Chapter 17, “How do I Control the Displayed Checker Messages?” on page 450).

### Using Ctrl+F8

Alternatively, you can use Ctrl+F8 (Options->Next Checker Message)

**Prerequisite:** Settings->Options->Environment->Preferences->Jump automatically to first item in checker list must be set to Yes.

1. When you do the syntax check, the task referred to in the message will automatically open, and the cursor will be located on the item that is the issue.
2. When you want to go to the next error, press **Ctrl+F8**, and you will jump to the next item.

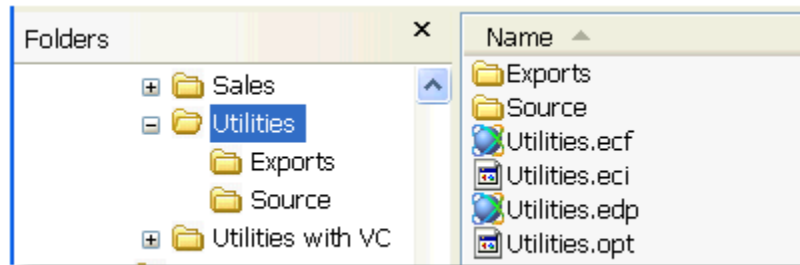
## How do I Back Up the Project?

It is always important to back up your work. You may have a good source control product in place that will handle backups automatically, but even without one, you have some easy options for doing good backups:

- Make backups of the operating system files
- Use the eDeveloper Export to create a packaged version of the XML files
- Make copies of programs as you work

We'll take a look at each of these below.

### Backing up the operating system files



When eDeveloper creates a new project, the structure of the project consists of a root level, which has the **.edp** (project) file, and some subdirectories. The actual program source is held in a series of XML files in the Source subdirectory. Usually, other supporting files, such as images or HTML templates, would also be located in subdirectories.

So, the quickest and easiest way to back up the entire project is to simply compress the folder at the top layer ("Utilities" in our example). You can keep multiple compressed versions of the project, if you want. This method ensures that you have a complete snapshot of your current project.

You can also copy this directory onto other media (such as a CD or an external hard drive), which is good insurance in case of hardware failure.

**Hint:** The *Magic.ini* file may or may not be in this directory, depending on how you have it installed. It is a good idea to keep a backup of the *Magic.ini* file too, in any case, since by the time you have been working on a project for awhile the *Magic.ini* may be highly customized and it can take awhile to restore it if it gets lost.

### Creating export files

You can also export objects from your eDeveloper project, or export the entire project. This allows you to create a “bundled” version of the project.

1. Select **File->Export/Import** (**Ctrl+Shift+E**).
2. By default, the Operation is Export. Leave that as it is.
3. By default, the Type is Entire Project. Leave that as it is to back up the entire project. However, if you want, you can choose to back up just parts of your project, such as only the Models or Data sources. You can further refine the export by choosing a folder or selecting only a certain range of sequence numbers.
4. Choose a file name. This is where the file will be created. You can zoom to choose the location. The suffix “.xml” will automatically be appended when the file is created.
5. Press OK.

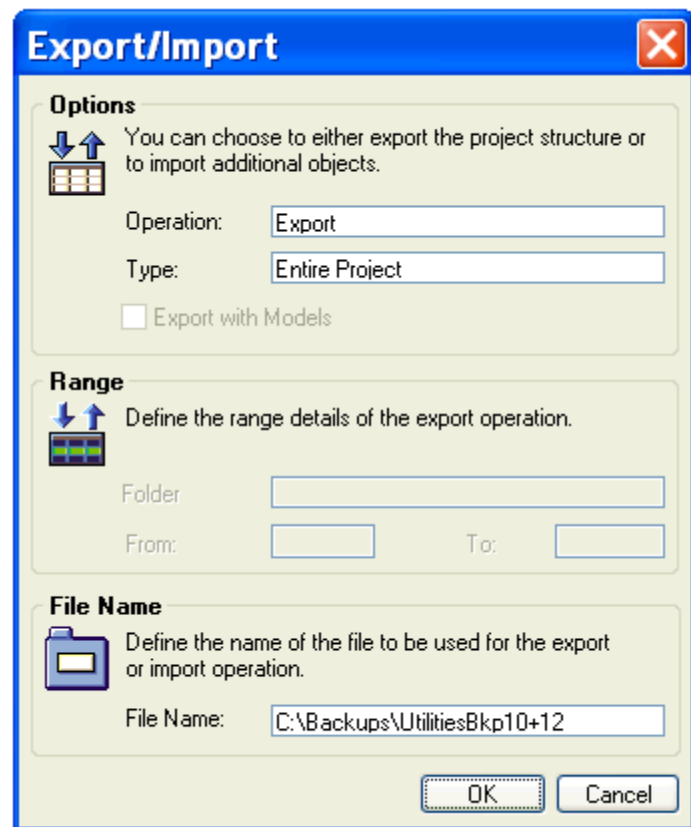
When the export is finished, the file will be created.

**See also:** Chapter 2, “How do I Transfer Objects From One Project to Another?” on page 33.

### Making copies of programs as you work

Before you start working on a program, you can always make a backup copy of it, using **Edit->Entry->Repeat Entry (Ctrl+R)**. This isn’t as comprehensive as backing up the entire project, but it is a quick way to make sure you can get back to your original copy. This is especially useful if you are trying something experimental.

How to do this is explained in detail in Chapter 1, “How do I Repeat an Entry in the Studio?” on page 12 and Chapter 1, “How do I Replace an Entry in the Studio with Another Entry?” on page 14.



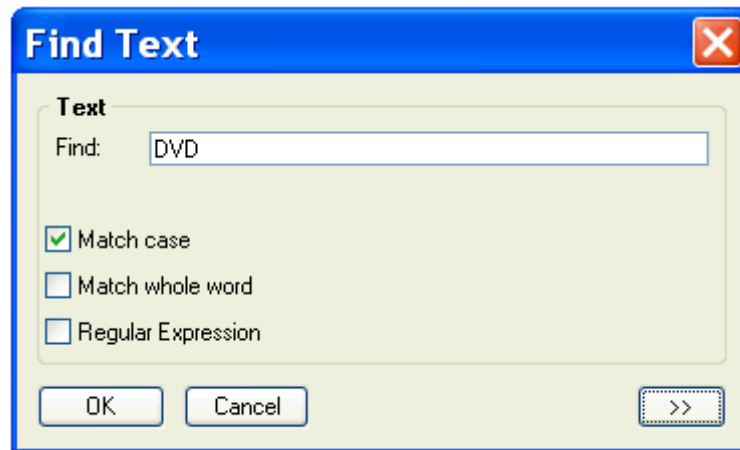


## How do I Search and Replace Text in the Project's Objects?

Sometimes it is useful to find text within your project, and perhaps change it. For instance, you might have a copy of a report or screen print, but have no idea which task was involved. Or, you may have a situation where it has been decided to change the name of a certain piece of data: using “Salesrep” instead of “Salesman” for instance, or if some hard-coded company name is changed.

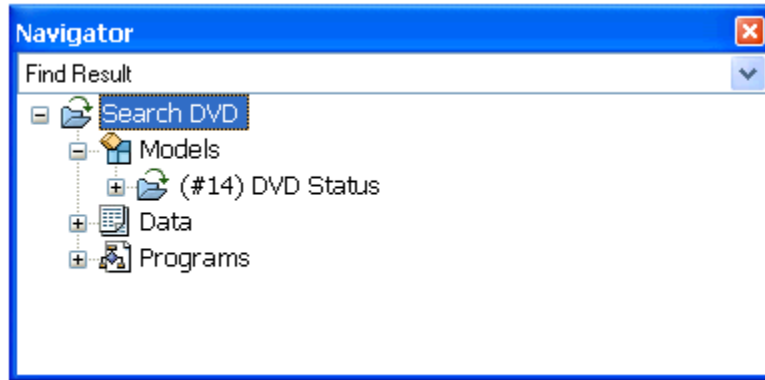
eDeveloper has good facilities for both finding and replacing text, as we will see below.

### Finding Text



1. Select **Edit->Find and Replace->Find Text (Ctrl+Shift+F)**
2. The *Find Text* dialog will appear.
3. Type in the text you want to find. In our example, we chose “DVD”.
4. Check *Match case* if you want the search to be case-sensitive.
5. Check *Match whole word* if you want only whole words that match.
6. Check *Regular Expression* if you want to use masking characters. These are explained in the eDeveloper Help file.
7. If you click on the >> button at the bottom, you will be able to refine your search, selecting which objects you want to search.

8. Click OK.

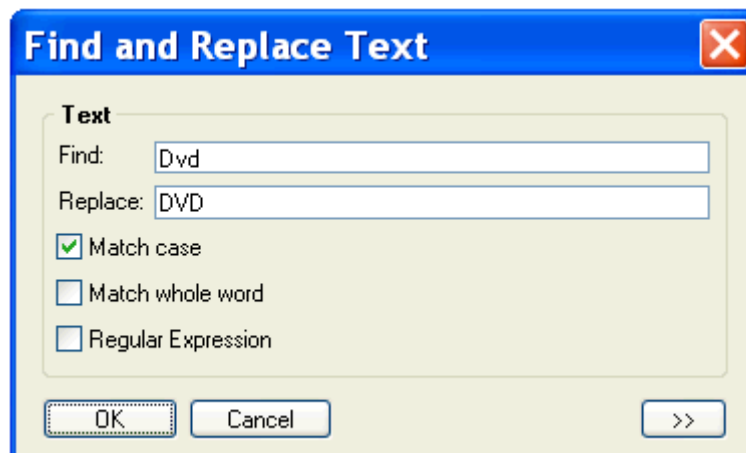


When the search is done, you will have a list of all references to the text in the Navigation pane. You can click on the results to go to the object.

Now, once you have a list of where the text is, you can save or print the list (see Chapter 32, “How do I Save or Print the Search Results?” on page 792).

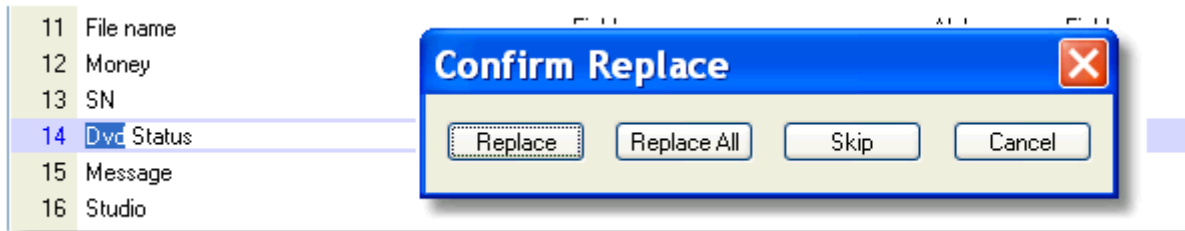
**Hint:** While you are working with the Find Result list, you can delete the items on the list if you want (**F3**, or **Edit->Delete Line**). This is useful when you are working on a long list; just delete the items as you fix them.

## Replacing Text

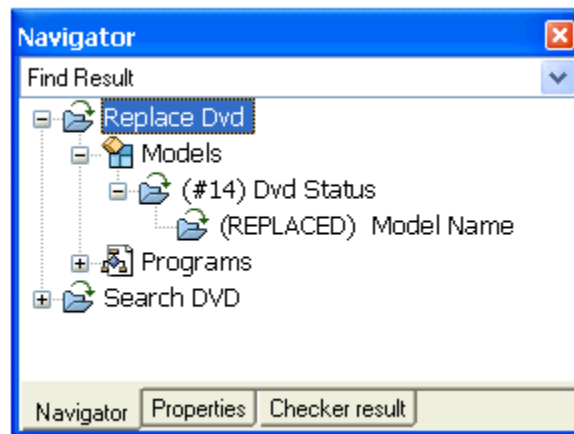


1. Go to **Edit->Find and Replace->Replace Text**.
2. Enter the text to find, in the *Find* field. Here, we are looking for the text “Dvd”, and we want to match the case.

3. Enter the text to replace the found text with in the *Replace* field.
4. Set the other options as you would if you were just finding text as described in the previous example.
5. Click OK.



6. If the text is found, you will be positioned on the first occurrence of the found text, with a *Confirm Replace* box. Here you have the following options:
  - **Replace**: Replace this instance of the text, then jump to the next instance.
  - **Replace All**: Replace all instances of the text, with no more prompting.
  - **Skip**: Leave this text as it is, and jump to the next instance.
  - **Cancel**: Stop the find and replace.



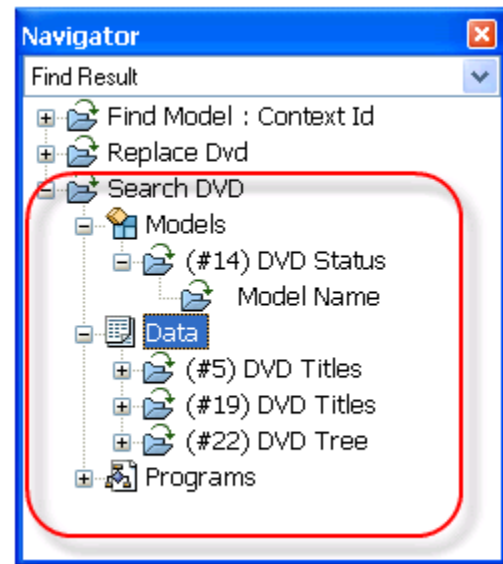
7. When the Replace is finished, you will see a list of all the items that were replaced, in the Navigator. You can click on these entries if you want, to check that you replaced the correct text.

## How do I Save or Print the Search Results?

After you have done a Find Text, Replace Text, or Find Reference, the results will be listed in the Navigator pane, under the Find Result list. You might find it useful to save these results to a file, or print them out. This is useful, for instance, when estimating how long some set of changes will take.

### Saving the search results

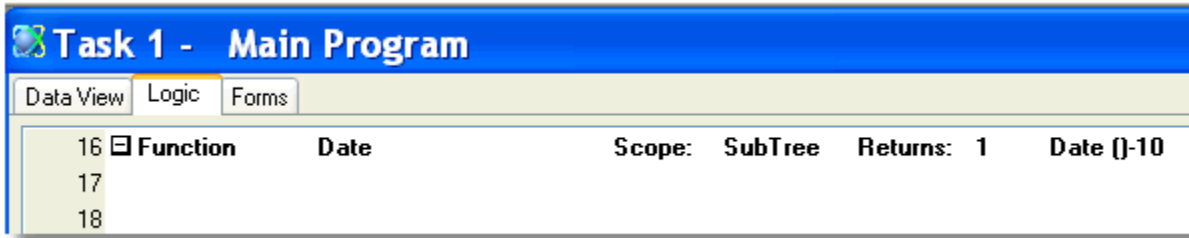
1. Park the cursor on the item you want to save. This can be at the top node of the item, such as “Search DVD”, or within the tree. In Our example, the cursor is parked on the “Data” node, so the entire “Search DVD” section marked in the rectangle will be what is saved.
2. Select **Edit->Find and Replace->Save Find Result**.
3. You will be prompted with a Windows *Save As* dialog. Choose the folder and file name where you want to save the file.
4. Click *Save*.
5. The section you chose will be sent to a text file.



### Printing the search results

1. Park the cursor on the item you want to save. This can be at the top node of the item, such as “Search DVD”, or within the tree, as in our example. Either way, that section will be printed.
2. Select **Edit->Find and Replace->Print Find Result**.
3. You will be prompted with a Windows print dialog. Choose the printer you want to use, and press Print.
4. The section you chose will be printed.

## How do I Override eDeveloper Functions?



eDeveloper has a large number of good, useful functions. However, you can override any of these functions easily, to create your own customized versions.

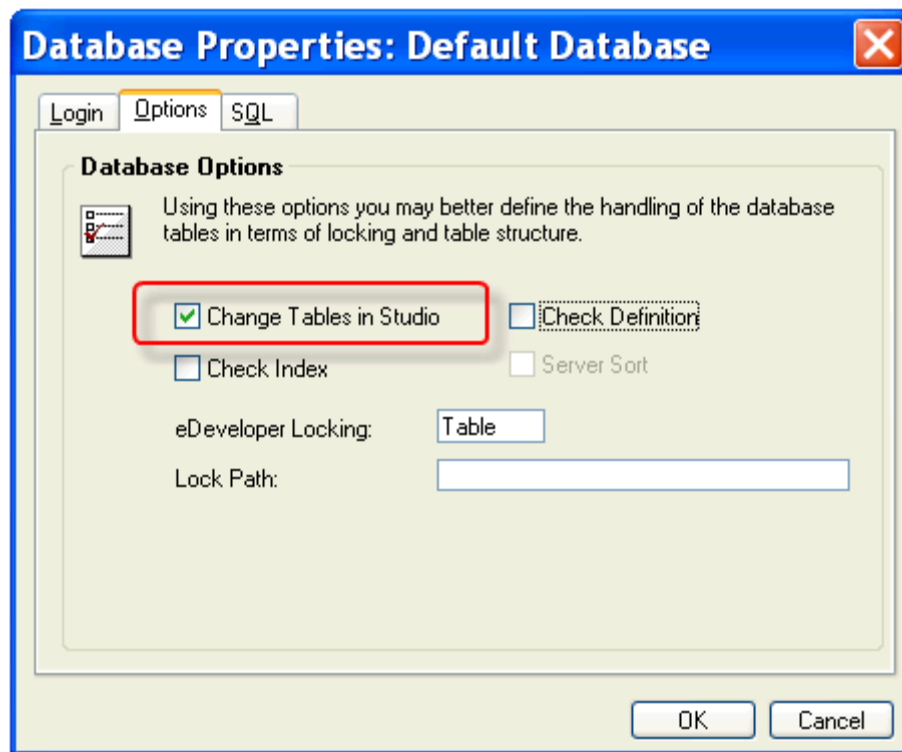
To do this, follow the instructions on Chapter 11, “How do I Create a Function That is Available For the Entire Project?” on page 248. However, make the function name be identical to the function in eDeveloper.

For instance, in this example we overrode the **Date()** function, so it will return a day 10 days ago.

## How do I Account for Incompatibilities in Table Structure Between eDeveloper and the Database?

When you are working with an ISAM table, there is always the possibility that the table as it exists does not match the table definition in eDeveloper. This can happen if changes were made in eDeveloper and the changes were not made in the DBMS, or vice versa, or if two different eDeveloper projects are not kept in sync with each other.

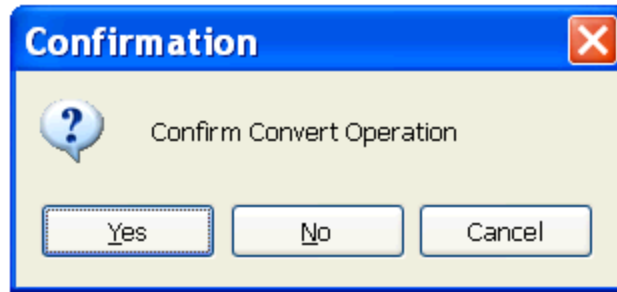
### Allowing eDeveloper to automatically convert the table



1. First, close your project, if a project is open.
2. Then go to **Options->Settings->Databases**
3. Select the database you want to work with, and press **Alt+Enter** to bring up the Database Properties.
4. Click on the **Options** tab.
5. Make sure **Change Tables in Studio** is checked.

If **Change Tables in Studio** is checked, then when you make some change to the table definition in the Data source repository, eDeveloper will automatically take care of converting the actual table. eDeveloper will convert the actual data (if any exists) and also the definition of the table within the DBMS.

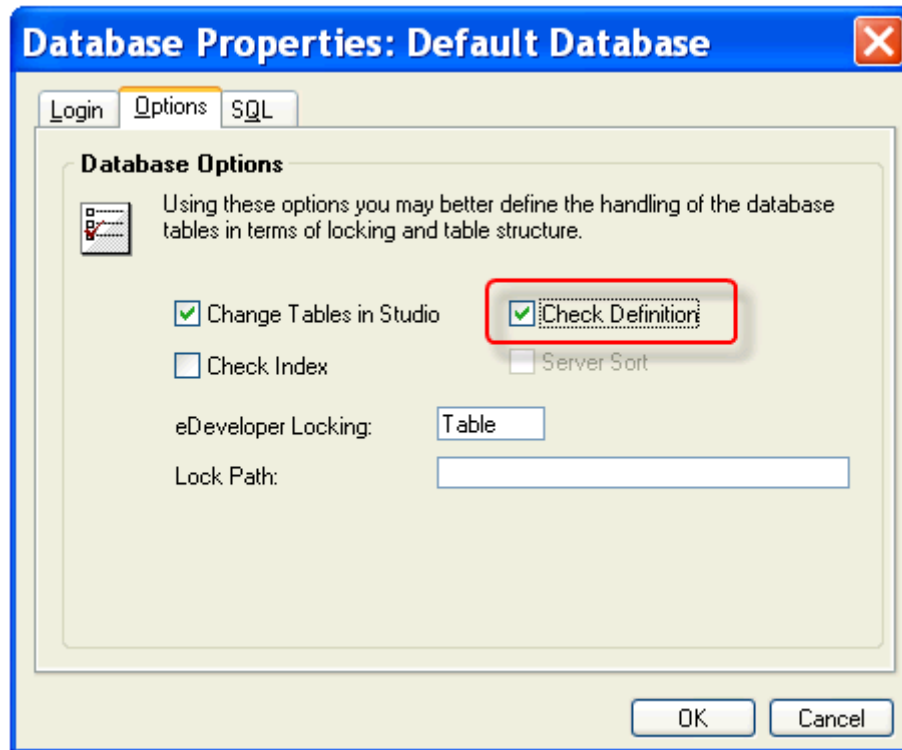
For instance, if you were to change a field definition from Numeric to Alpha in a given table, then exit, you would receive a message such as:



When you click on Yes, eDeveloper will convert the numeric data 3 to the alpha text '3'. eDeveloper will also handle the changes if you shuffle the field positions, or change the indexes.

It is, however, important that if you change the table, you allow eDeveloper to convert the data each time. If the Confirmation box appears, and you click "No", then the table definition will be out of sync with the data.

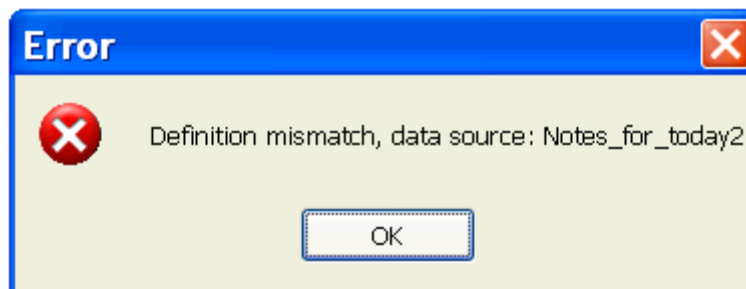
## Checking for compatibility



While you are testing, you can allow eDeveloper to automatically test that the data is synchronized, for ISAM tables. To do this:

1. First, close your project, if a project is open.
2. Then go to **Options->Settings->Databases**
3. Select the database you want to work with, and press **Alt+Enter** to bring up the Database Properties.
4. Click on the **Options** tab.
5. Make sure **Check Definition** is checked.

Now, if you try to access a table that is not in sync, you will get a message such as:





Also, an error event will be raised, which you can trap to log and give a meaningful message to the user.

## How do I Export a Pervasive ISAM Table Structure to be Used with External Tools?



Pervasive tables can be used with other tools, such as Crystal Reports, if there is a file called a DDF available to the other tool.

Creating the DDF files is done using the DDF Maker Wizard. This tool converts the internal eDeveloper data structure into correct DDFs. To create the DDFs:

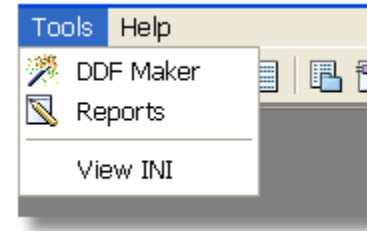
1. Create your Data sources in eDeveloper, using Pervasive as the underlying DBMS.
2. Syntax check the tables to make sure they are correct (**Options->Check Syntax**; see Chapter 32, "How do I Validate a Data Source Structure?" on page 782).
3. Select **Tools->DDF Maker**.
4. Follow the prompts to select the files you want to create DDFs for, and where you want the DDF files to be located.

You will need to be familiar with how your external tool uses DDFs, and what data types it will support. For instance, some tools will only accept date fields that are in certain formats.

## How do I Add External Tools to the Studio?

eDeveloper includes a very powerful feature called the *Tools Infrastructure*.

Using the Tools Infrastructure, you can make your own customized tool menu for your application, using external tools, or tools built in eDeveloper. In fact, some of the Wizards that are used in eDeveloper are created using eDeveloper.



The tools are added to the Studio in the Magic.ini file, in the **[TOOLS\_MENU]** section. In our example, we used:

```
[TOOLS_MENU]
Menu1 = A,&DDF Maker,,Add_On\DDFMaker\DDFMaker.ecf,,,Add_On\DDFMaker\DDF+
Maker_suf.opr,ImageFor = B ToolNumber = 60 ToolGroup 1
Menu2 = A,&Reports,,Add_On\ReportGenerator\ReportGenerator.ecf,,,+
,ImageFor = B ToolNumber = 23 ToolGroup 1
Menu3 = S,,,,,
Menu4 = O, &View INI,,notepad.exe %WorkingDir%magic.ini,,,,
```

There are 4 different types of items that can be put on the menu:

- An **Application**, which calls an eDeveloper *.ecf* file.
- An **OS command**, which executes any operating system command (batch file, executable, etc.)
- A **Submenu**, which is a header for more entries beneath
- A **Line separator**

The latter three are fairly self-explanatory. However, the first type, calling an *.ecf* file, is where you can develop (or use) very powerful tools. When an *.ecf* file is opened, there can be pre- and post- scripts which allow you to do macro-type processing.

The syntax varies slightly between the menu types, since not all entries are required for each.

Menu Entry	Syntax
Application (formatted for readability)	<b>&lt;menu name&gt; = A , &lt;caption&gt; , &lt;parent menu&gt; , &lt;ECF path&gt; , &lt;access key&gt; , &lt;pre-operation command file&gt; , &lt;post-operation command file&gt; , &lt;icons&gt;</b>
OS Command	<b>&lt;menu name&gt; = O ,&lt;caption&gt; ,&lt;parent menu&gt; ,&lt;command&gt; , &lt;access key&gt; , , ,&lt;icons&gt;</b>
Submenu	<b>&lt;menu name&gt; = M ,&lt;caption&gt; ,&lt;parent menu&gt; , , , ,</b>
Line separator	<b>&lt;menu name&gt; = S , ,&lt;parent menu&gt; , , , ,</b>

Below is a summary of the menu items. These are explained more fully in the eDeveloper Help .

Menu Parameter		
<b>menu name</b>	Name of this menu entry	This is the name that is used for the Submenu entries. It doesn't show up on the menu
<b>menu type</b>	<ul style="list-style-type: none"> <li>• A = Application</li> <li>• O = OS Command</li> <li>• M = Submenu</li> <li>• S = Separator</li> </ul>	
<b>caption</b>	Name that will show up on the menu	This is what shows up on the menu. Can include & for an accelerator key.
<b>parent menu</b>	The name of the parent menu	This is blank for top-level menus. Parent menu should be of type M.
<b>ECF path\command</b>	<ul style="list-style-type: none"> <li>• ECF: Points to an eDeveloper cabinet file.</li> <li>• OS Command: any valid operating system command</li> </ul>	This points to an <i>.ecf</i> file or contains an operating system command, depending on the menu type.
<b>access key</b>	Accelerator key combination, such as Ctrl+1	
<b>pre-operation command file</b>	Script file of commands to be executed before the ECF or OS command	The command files have their own powerful set of commands. These are explained in the eDeveloper Help file.
<b>post-operation command file</b>	Script file of commands to be executed after the ECF or OS command	
<b>Icons</b>	<b>ImageFor</b> = B <b>ToolNumber</b> = 60 <b>ToolGroup</b> 1	Icons to be used on the menu. The syntax here is the same as that used on the Menu entries in eDeveloper.

## Adding an external tool

OK, so let's look at how to add one external tool to the current menu. Let's add a call to the registry editor.

1. Close your current application.
2. Open the Magic.ini file you are using in an editor.
3. Go to the [TOOLS\_MENU] section.
4. Add the OS call to Regedit according to the syntax described above:

```
[TOOLS_MENU]
Menu1 = A,&DDF Maker,,Add_On\DDFMaker\DDFMaker.ecf,,Add_On\DDFMaker\DDF+
Maker_suf.opr,ImageFor = B ToolNumber = 60 ToolGroup 1
Menu2 = A,&Reports,,Add_On\ReportGenerator\ReportGenerator.ecf,,,+
,ImageFor = B ToolNumber = 23 ToolGroup 1
Menu3 = S,,,,,,
Menu4 = O, &View INI,,notepad.exe %WorkingDir%magic.ini,,,,
Menu5 = O, Edit Registry,,regedit.exe,F12,,,
```

In our example, we made the **F12** the accelerator key.

5. Save the *Magic.ini* file, and restart the Studio. Now you will see your new menu entry. Selecting this item will bring start *Regedit*.



## How do I Run External Processes Automatically?

When eDeveloper starts up, you can launch a file of command that will execute automatically. These commands can be macro commands that will do things within eDeveloper, such as importing or exporting objects, moving to a certain part of the Studio, or typing text.

To run external processes, you need two things:

- To create an external command file
- To set up eDeveloper so the command file is executed when eDeveloper starts

### Creating the command file

The command file has its own syntax and set of commands. These are the same commands that can be used with the Tools Infrastructure in Chapter 32, “How do I Add External Tools to the Studio?” on page 799 for the Pre- and Post- command files. The exact syntax is in the eDeveloper Help, but basically each command replicates what goes on when you do the same operation manually from within eDeveloper. Here is an overview of the commands:

- *Export*: Exports an application, or pieces of an application. This is useful if you want to back up all projects at midnight, for example.
- *Import*: Imports an application, or pieces of an application
- *ECF*: Creates a cabinet file from an application. This is useful if you want to create a cabinet file of all applications in one batch process.
- *Get definition*: Imports the definition of an SQL table.
- *Project*: Opens a project
- *Simulate*: Simulates going to menu items or typing on the keyboard

There are also Global Parameters you can query, to get specific information about an application, such as how many programs or models are in the application.

### Setting up the command file to automatically execute

Once you have the command file in place, you need to set up eDeveloper to automatically execute it. Since the command file can, in fact, create the project you are going to open, the command file cannot be in the project. So it is set up in the Magic.ini file. Adding the following line:

```
AutomaticProcessingSequenceFile=%Path%Filename.txt
```

to the [MAGIC\_ENV] section will cause the command file **Filename.txt** in the directory pointed to by **%Path%** to be executed.

### Setting up the automatic processing in batch

There may be instances where you want the processing to happen in the background, and not open up the application at all. For instance, you might want to simply export an application into a document. In this case, you can set up the Magic.ini so it executes the commands, then quits. You do this by adding the line:

```
AutomaticProcessingMode= B
```

to the [MAGIC\_ENV] section.

## How do I Limit the Use of my Application Using the eDeveloper License Mechanism?

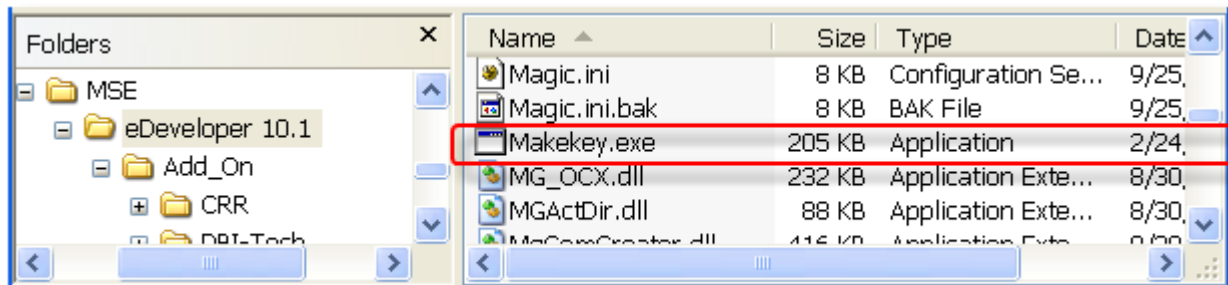
When you use eDeveloper, eDeveloper uses an internal licensing mechanism to regulate the total number of concurrent users. You have the option of using that same licensing mechanism in your own application, for licensing the number of users who purchase licenses from you.

This is done in three steps:

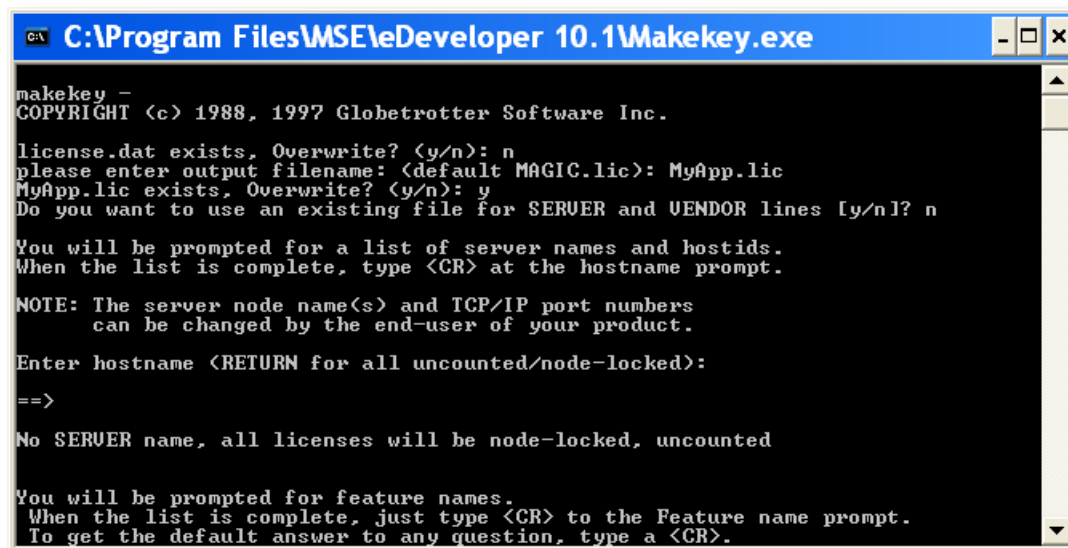
1. Use the *MakeKey* utility to generate a license key for your user
2. Use **LMChkOut()** to check out a license as each user logs in
3. Use **LMChkIn()** to check the license back in

These are explained below.

### Running the MakeKey utility

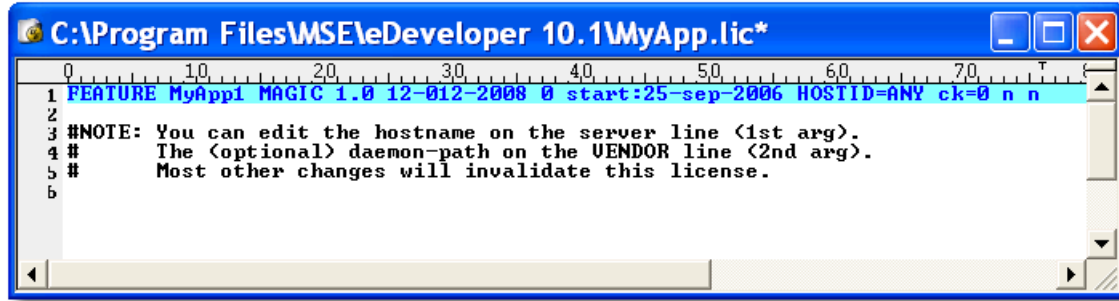


1. You will find the Makekey.exe utility in the eDev 10 installation directory. Click on it.





2. You will be prompted through a series of items. These are specific to the FlexLM product, and your answers will depend on what you are trying to do.



3. When you are finished, you will have a feature that your users can use with your application.

### Using LMChkOut()

1. In your application, when a user logs in, call the **LMChkOut()** function to check out a license. The syntax is:

```
LMChkOut ( <license file name> , <feature name> , <version> )
```

The return code will indicate whether the license is ok, not found, expired, and other conditions (see the eDeveloper help for details).

### Using LMChkIn()

1. In your application, when a user logs out, call the **LMChkIn()** function to check the license back in. The syntax is:

```
LMChkIn ( <feature name> )
```

The return code will be zero if it checked in successfully, 8 otherwise.

## How do I Find Which Computers Use a Specific eDeveloper License?

You can use the MGStations.exe utility from the Support directory to verify how many licenses are available on a specific license, the license code, how many licenses are actually used, and a list of the users that are using those licenses.

The syntax is:

```
MGStations <license> <license file>
```

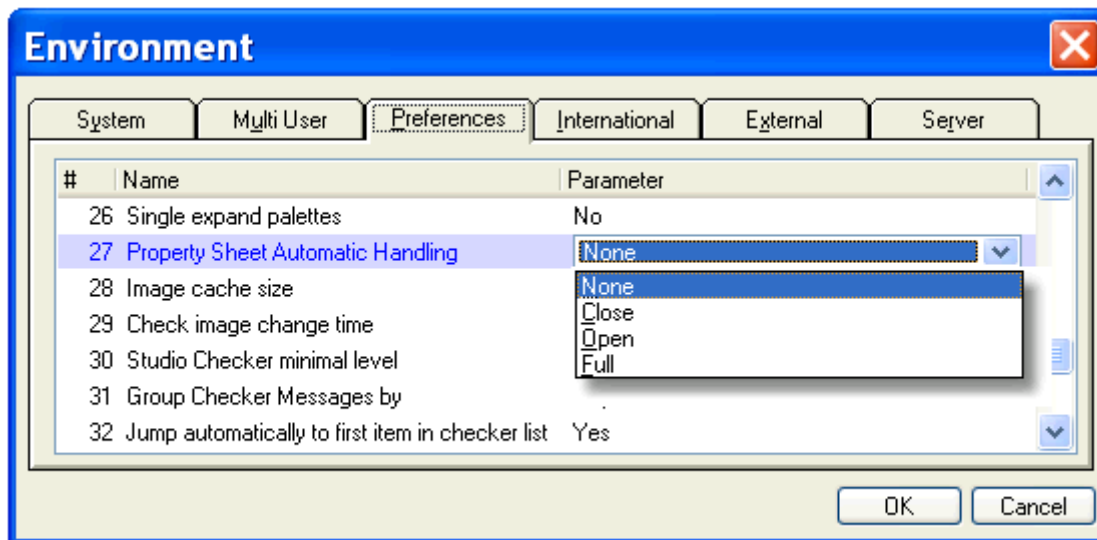
For instance:

```
C:\Program Files\MSE\eDeveloper 10.1\mgstations" MGC-  
STK C:\eDev10Projects\license.dat
```

Might return that there are two licenses available, and one is being used.

# Chapter 33: Studio Configuration

## How do I Prevent the Property Sheet and the Navigator From Appearing Automatically?



You can control when the Property sheet appears in the Studio in **Options->Settings->Environment->Preferences->Property Sheet Automatic Handling**. There are four basic modes:

- **None:** eDeveloper does not open or close the Property sheet.
- **Close:** eDeveloper closes the Property sheet when it isn't relevant, but doesn't reopen it.
- **Open:** eDeveloper automatically opens the Property sheet when it is relevant, but doesn't close it.
- **Full:** eDeveloper opens the Property sheet when it is relevant, and closes it again when it isn't.

If you don't want the Property sheet to open automatically, but you do want it to close automatically, select **Close**.

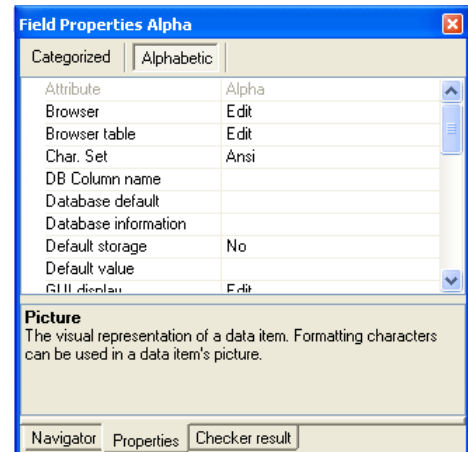
Otherwise, if you want complete control over the Property sheet, select **None**.

## How do I Separate the Palettes Once They are Merged?

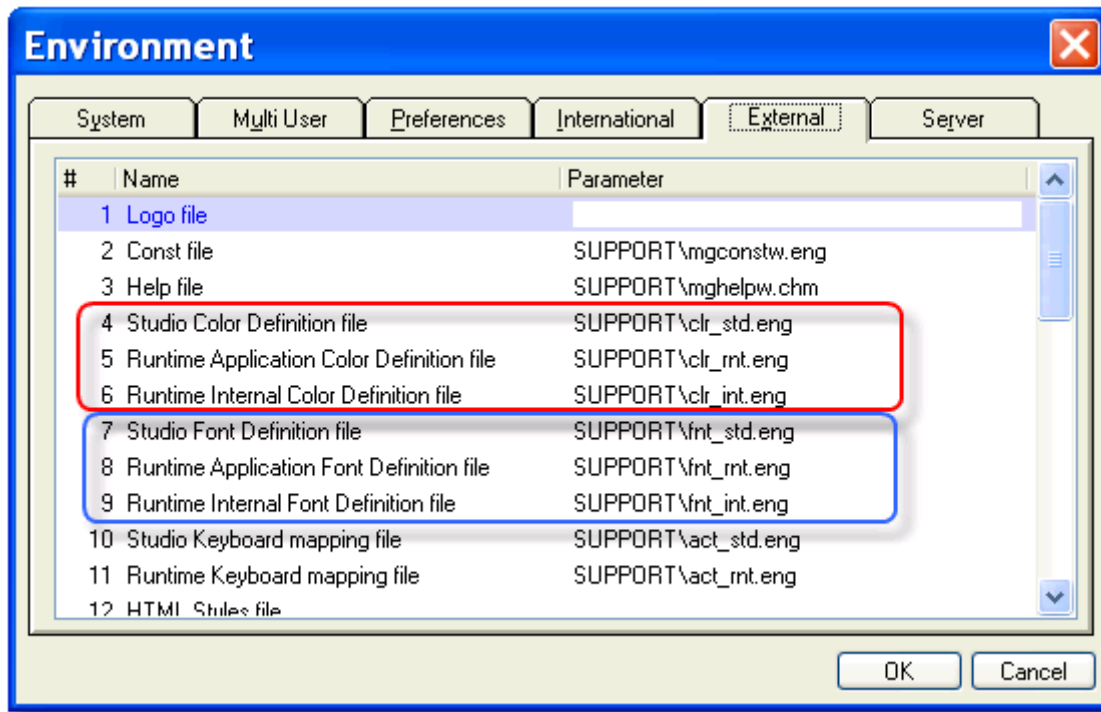
The Studio palettes are very flexible and can be resized, moved, or docked according to your work style. To save space, the palettes can also be combined as a single window. Sometimes they become combined accidentally; if you move one too closely to another.

### To separate combined palettes

1. Move the focus to the palette.
2. Holding down the **Ctrl** key, drag the title bar off in any direction.
3. Let go of the **Ctrl** key.
4. Repeat the process if you have three combined palettes. The palettes will now be separated.



## How do I Change the Studio Color and Fonts?



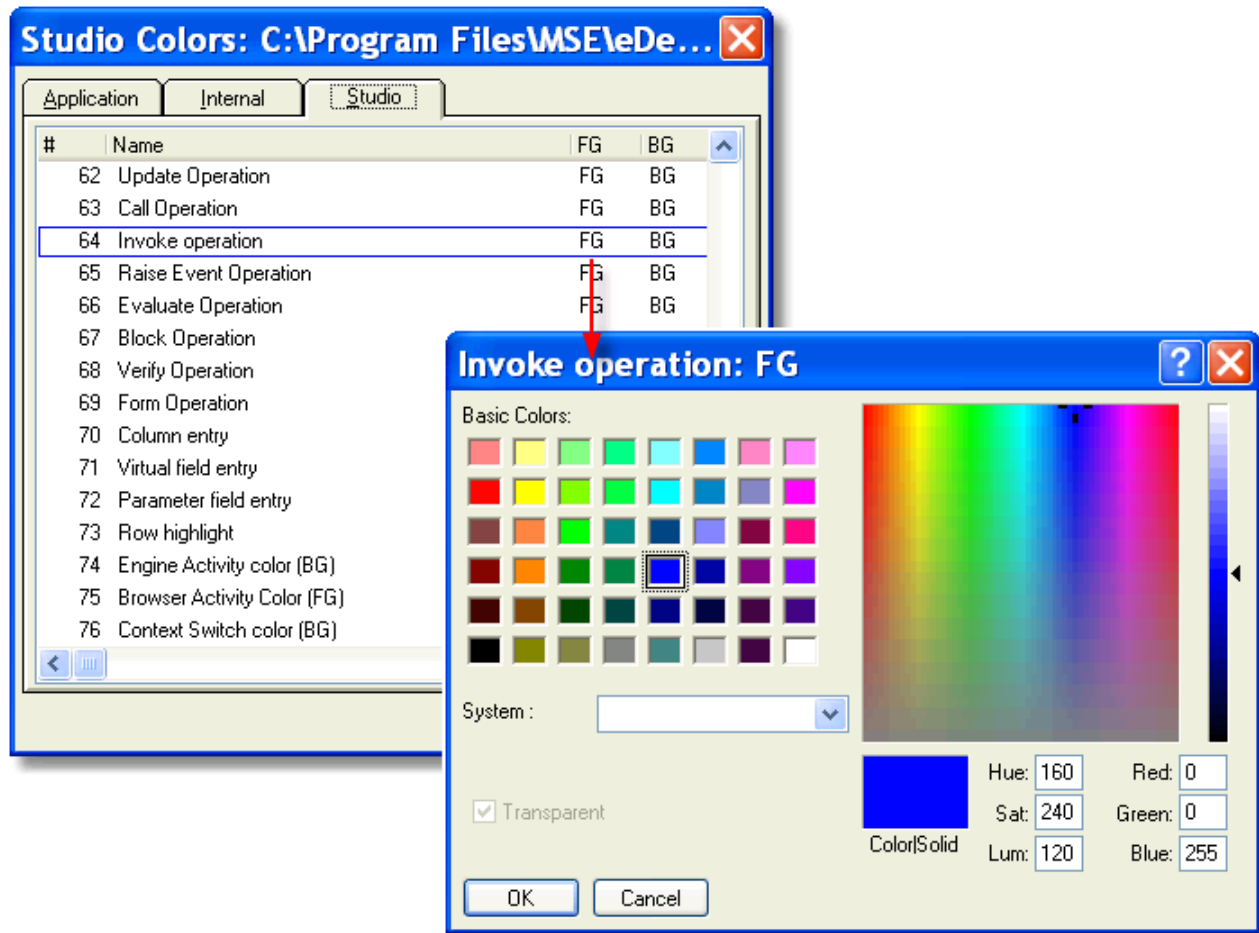
In eDeveloper, the colors and fonts are divided into 3 separate files:

- *Studio*: The colors and fonts you see while working in eDeveloper
- *Runtime Application*: The colors and fonts you use to create your application
- *Runtime Internal*: The colors and fonts used by eDeveloper at runtime (for items that are part of the basic structure of eDeveloper, such as pop-up dialogs and menus)

The files that are used to hold each of these color and font definitions are specified in the Magic.Ini. You can easily set the files used by changing them in **Options->Settings->Environment->External tab**, as shown above.

It can be very useful to customize the Studio color and font files to fit your work style, screen resolution, and screen size. You may even want separate color and font files depending on whether you are working on a laptop or desktop.

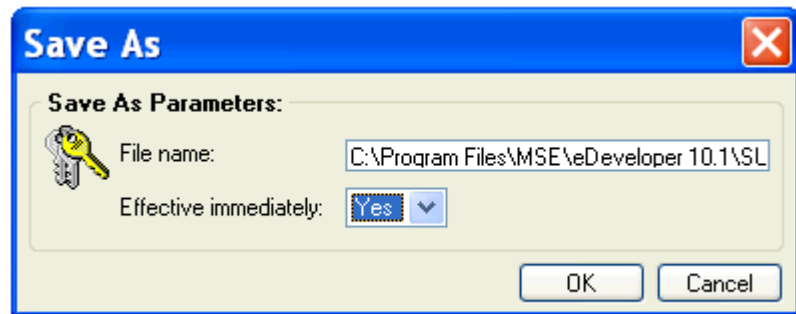
## Changing the Studio Colors



1. Go to **Options->Settings->Colors**
2. Click on the **Studio** tab.
3. Scroll to the values you want to change. The Operation colors, for instance, are useful ones to change because you can make your programs easier to read.
4. You can zoom from the foreground color (**FG**, which changes the text color) or background color (**BG**, which changes the color of the field the text sits on, the row color). When you zoom, the Windows color picker will appear.
5. If you choose a *System* color, then the color will be inherited from Windows.
6. Otherwise, if the System color is blank, you can choose a *Basic color* or choose a custom color.

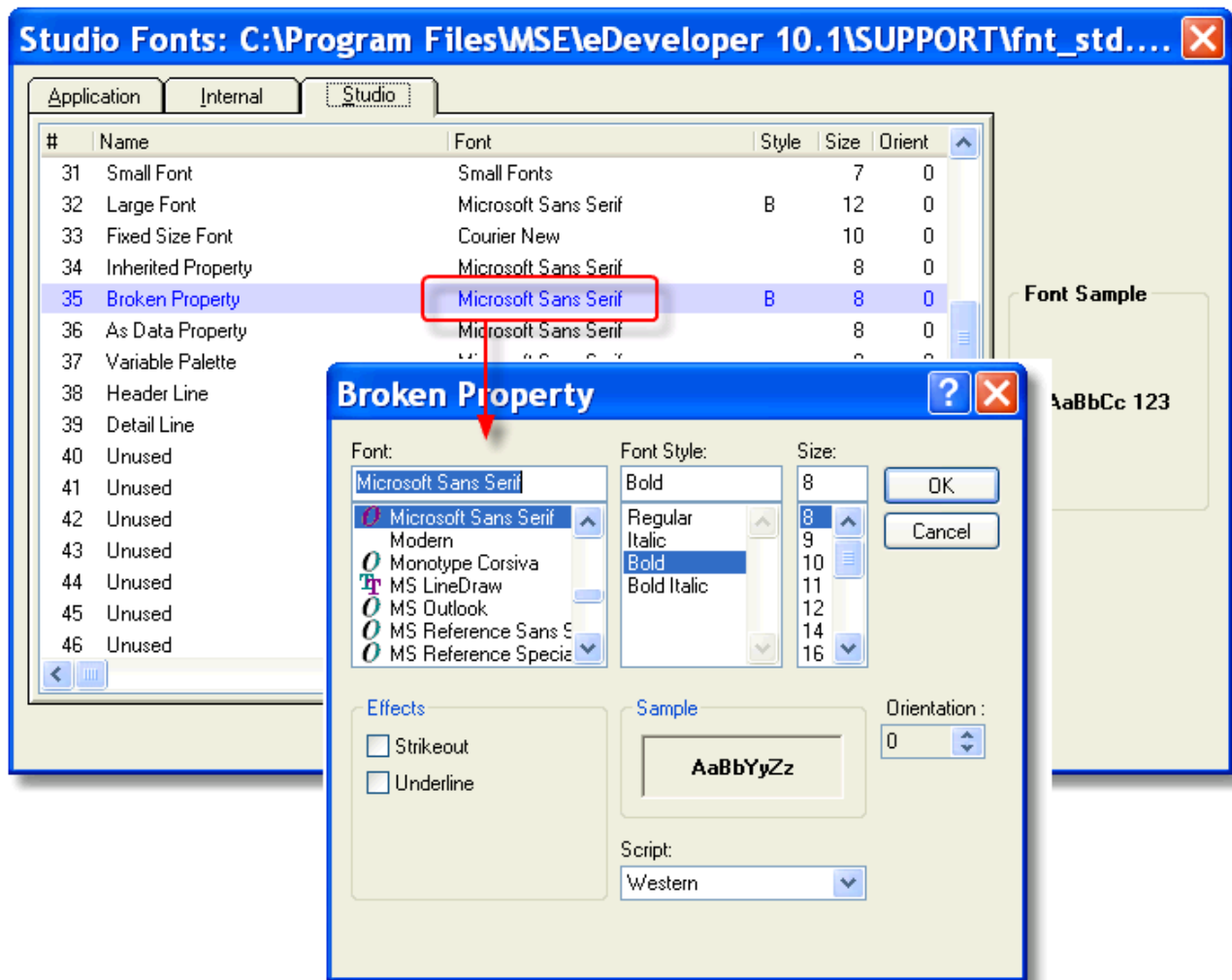
7. When you are done, click OK.
8. When you exit the Colors dialog, you will see a Save As dialog. Select *Effective Immediately* = **Yes**, and click OK.
9. Now your new colors will be in effect.

The colors will be saved in the text file specified in the *Magic.ini* and will be effective immediately.



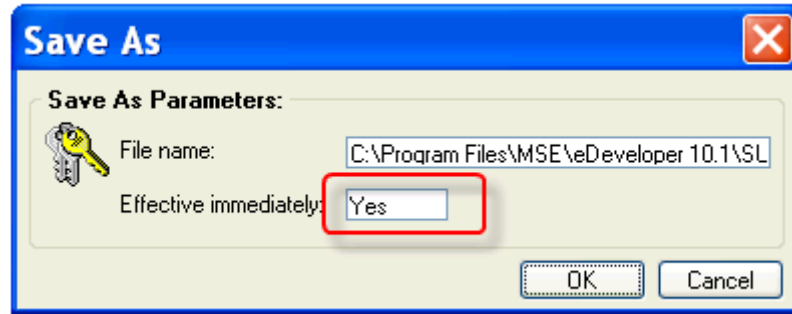


## Changing the Studio Fonts



1. Go to **Options->Settings->Fonts**
2. Click on the **Studio** tab.
3. Scroll to the values you want to change.
4. Zoom from the Font column to change the font. A font dialog box will appear. You can choose the font, font style, and size. You will see a sample of the selected font in the Sample box onscreen.

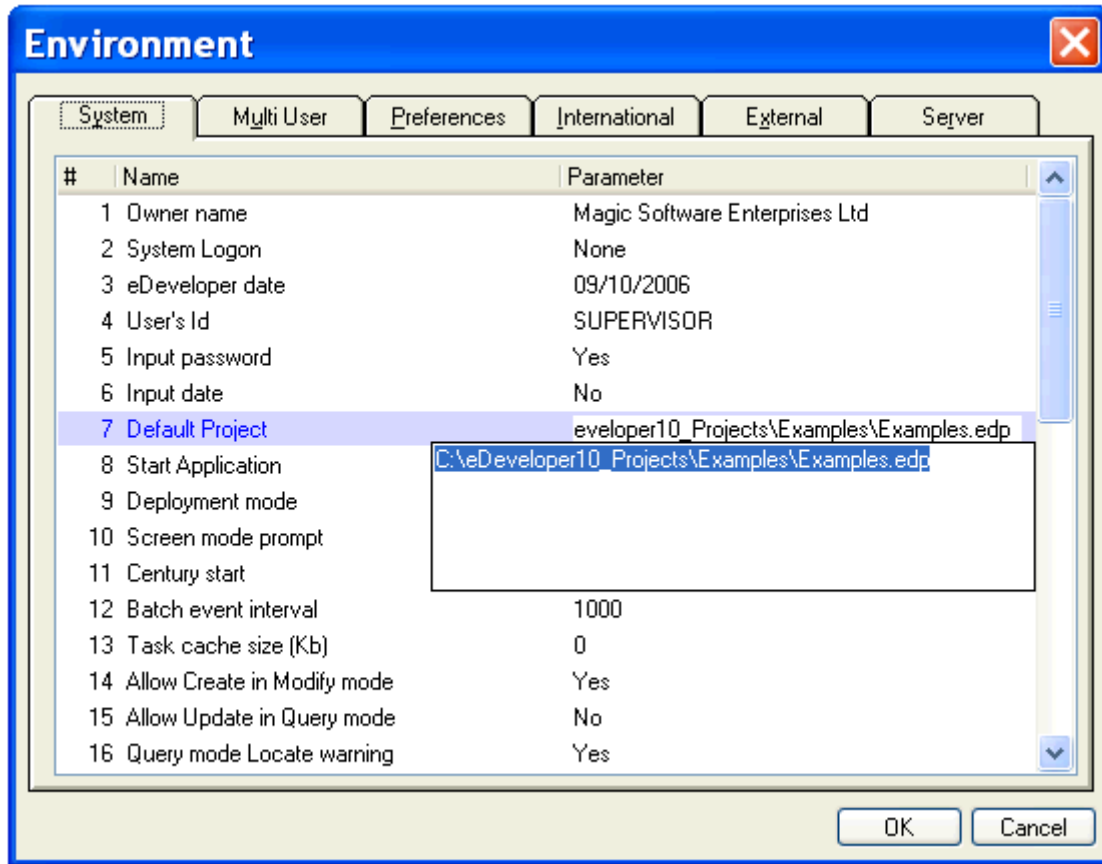
5. When you are satisfied, click OK.



6. When you are done changing the Studio Fonts, click OK. You will see a Save As dialog box, as shown above. Change *Effective immediately* to **Yes**, and click OK.
7. Now your new colors will be in effect.

The fonts will be saved in the text file specified in the *Magic.ini* and will be effective immediately.

## How do I Automatically Load a Project Upon Running the Studio?



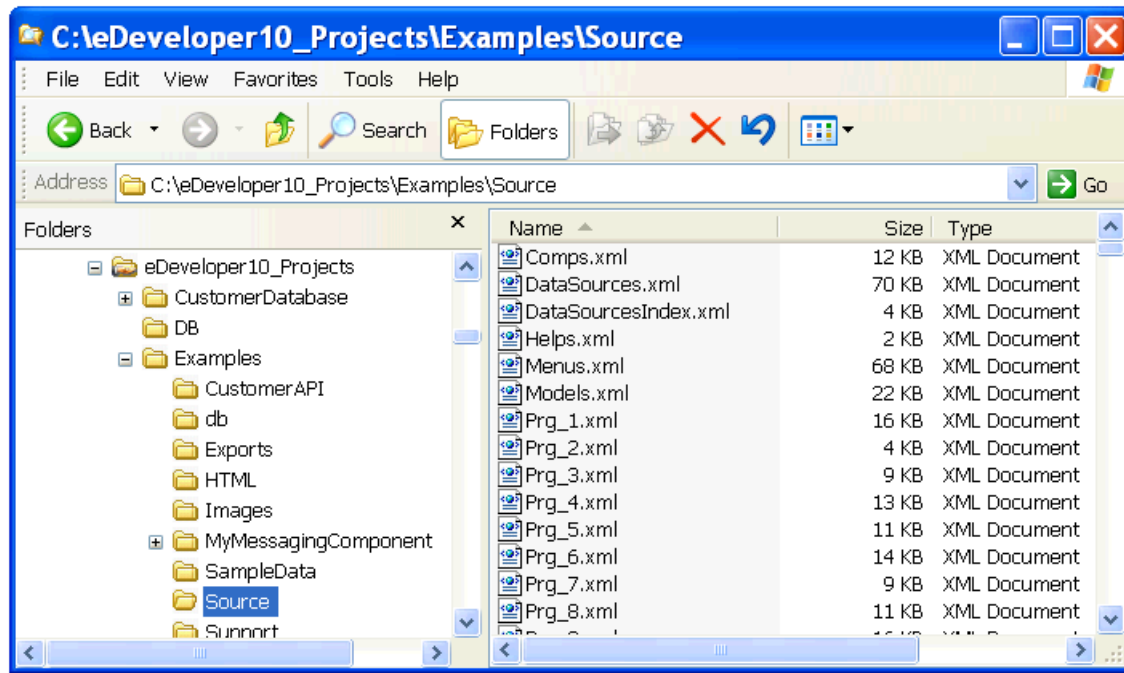
If you are working primarily on one application, you may want eDeveloper to automatically load that application without prompting you.

You can do this by setting the Default Project in the Magic.Ini file. The easiest way to do this is:

1. Go to **Options->Settings->Environment->System->Default Project**
2. Zoom to select the **.edp** file you want as your default project.
3. Press OK.

Now, when you start the Studio, this project will be opened automatically.

## How do I Control the Location of the XML Source Files of a Project?



eDeveloper source code is held in a series of XML files. By default, these are held in a subdirectory called *Source*, as shown here.

The XML source directory is always located relative to the *.edp* file. In this example, the *.edp* file is in **C:\eDeveloper10\_Projects\Examples**, so the source directory is **C:\eDeveloper10\_Projects\Examples\Source**.

### Changing the value of the Source directory in an existing *.edp*

Once the *.edp* file is created, the location of the source files is coded into it. The *.edp* file is, however, an XML file and can be edited. In this example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Application>

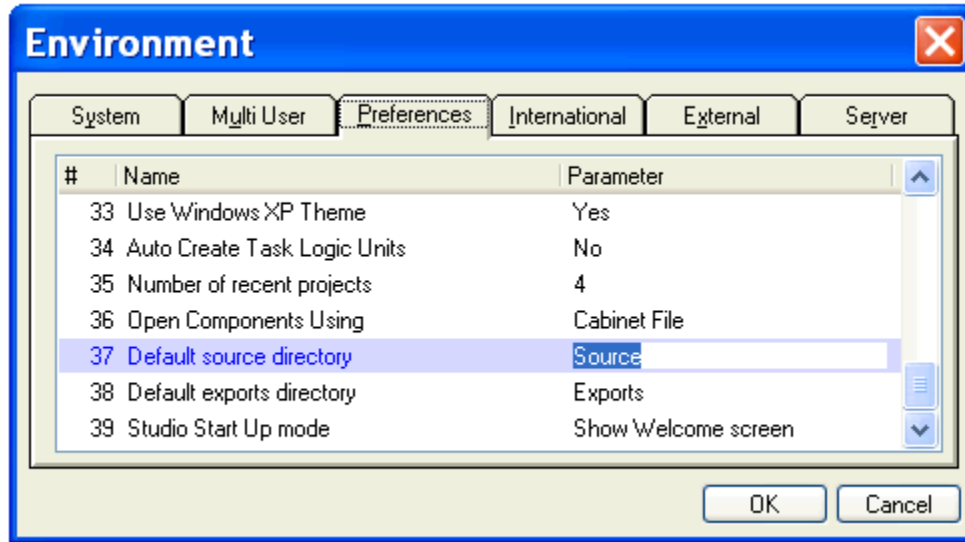
  <Project>
    <ProjectName val="Examples"/>
    <VCActive val="N"/>
    <WorkOffline val="N"/>
    <SourceDirectory val="Source"/>
    <ExportsDirectory val="Exports"/>
    <GUID val="{A0275EFF-1939-49CC-970A-98BA38F752A1}" />
    <ReferencedProjects>
      <RefProject FileName="Examples\MyMessagingComponent\MyMessagingComponent.Edp"
ProjectName="MyMessagingComponent" />
    </ReferencedProjects>
  </Project>

</Application>
```

You can see the name of the source directory. You can change the value of it, if you need to.

### Changing the default source directory

You can also change the value for the source directory for all future *.edp* files that you might create. this is done in the *Magic.ini* file, under the **DefaultSourceDir** setting.



Or, you can change it in **Settings->Options->Environment->Preferences->Default source directory**. Either way, the value will be saved in the *Magic.ini* and will be used the next time you open the Studio.

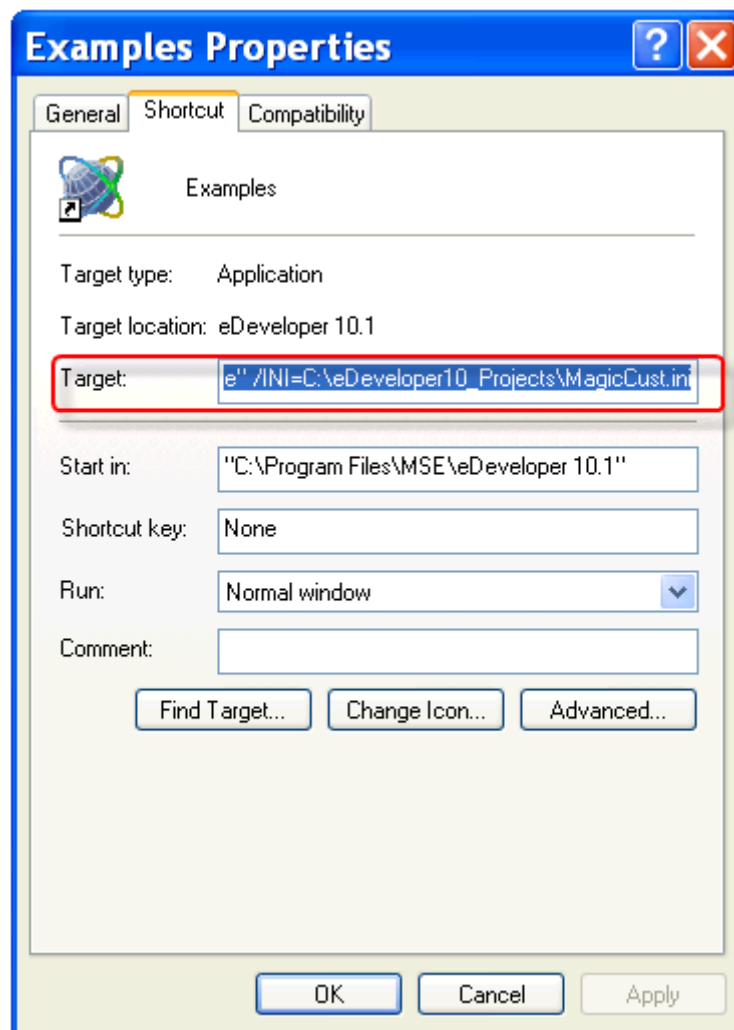
## How do I Define the Location of the .ini?

By default, when eDeveloper starts, it will use the file called “Magic.ini” located in the same directory as the *.edp* file. If that does not exist, it will use the “Magic.ini” file located in the eDeveloper installation directory.

However, you can override this behavior and locate the *Magic.ini* file wherever you like, and in fact use a different name if you like. This is often done to provide a specific login for certain users (such as testers or programmers) that is different from the default login.

Here is now to do it.

### Using a different .ini file



1. Create or copy your startup icon.

- 2.** In the Target field, add the location of your Magic.Ini file. This is done using the syntax:

```
<eDeveloper .exe file> /INI=<ini file name>
```

Which in our example is:

```
"C:\Program Files\MSE\eDeveloper 10.1\eDevStudio.exe"  
/INI=C:\eDeveloper10_Projects\MagicCust.ini
```

You can also do this using a .bat file or other script file.

## How do I Add Additional *.ini* Settings?

By default, when eDeveloper starts, it will use the file called “Magic.ini” located in the same directory as the *.edp* file. If that does not exist, it will use the “Magic.ini” file located in the eDeveloper installation directory.

However, you can override specific entries in the *.ini* file, or add entries, when you start up the Studio. This is useful when you want to use, say, a specific color file or use different files for testing.

Here is now to do it.

### Overriding *.ini* settings

First, create a file that has the settings you want to override. It is important that you use the forward slash in front of each item. For instance, in our example, we have:

```
/[MAGIC_ENV]InputPassword=N  
/[MAGIC_ENV]User = Supervisor  
/[MAGIC_ENV>Password =  
/[MAGIC_ENV]StartApplication=1  
/[MAGIC_ENV]ColorDefinitionFile=usrclr2.eng  
/[MAGIC_LOGICAL_NAMES]TEMP=C:\temp\  
/[MAGIC_LOGICAL_NAMES]Testing=Y
```

We have overridden several of the default settings, such as the userid and password. This saves time while we are programming.

Each item has the section of the INI in brackets, preceded by a forward slash.

### Using the overrides

Next, we need to instruct eDeveloper to use the override settings. We do this using the following syntax:

```
<eDeveloper .exe file> @<Override File Name>
```

Which in our example is:

```
"C:\Program Files\MSE\eDeveloper 10.1\eDevStudio.exe"  
@C:\eDeveloper10_Projects\Testing.ini
```

Now, the values in Testing.ini will override the values in the default Magic.ini.

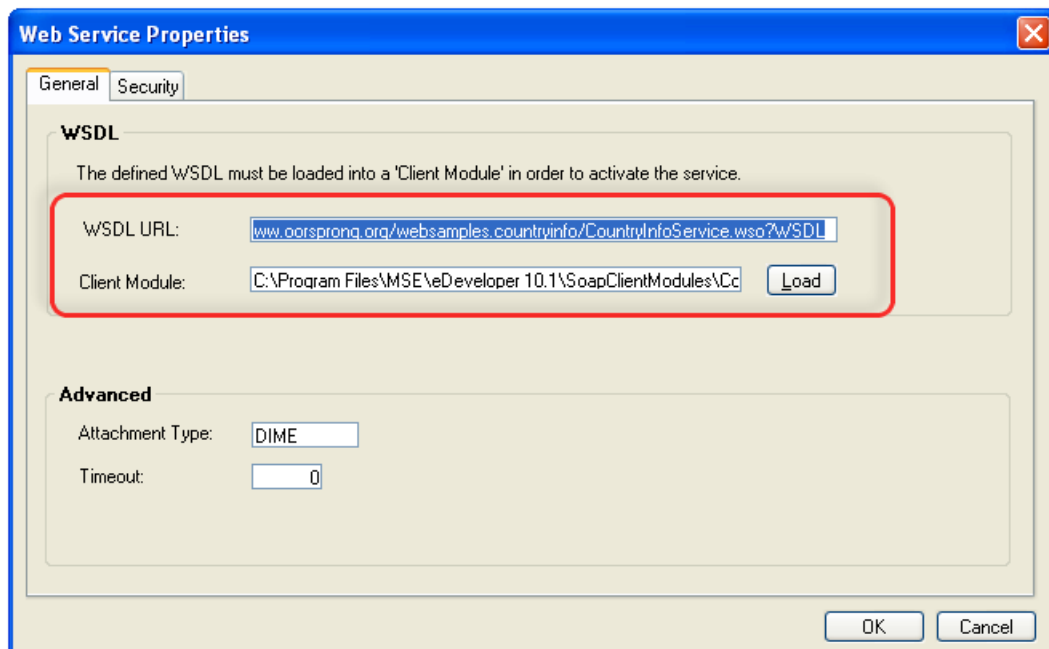


# Chapter 34: Consuming Web Services

## How do I Access a Web Service?

Web services are described in a special XML file called a WSDL (Web Service Description Language). The WSDL document describes the service, including the structure of the input and output. In order to access a web service, you must first find its WSDL entry, which will generally be a URL on the web. To access a Web Service from eDeveloper, you need to define a SOAP entry in the Services section and load the service WSDL. When the WSDL is loaded, a client module (JAR file) is generated and optionally XML schemas are generated as well.

## Creating a Web Service Client Entry

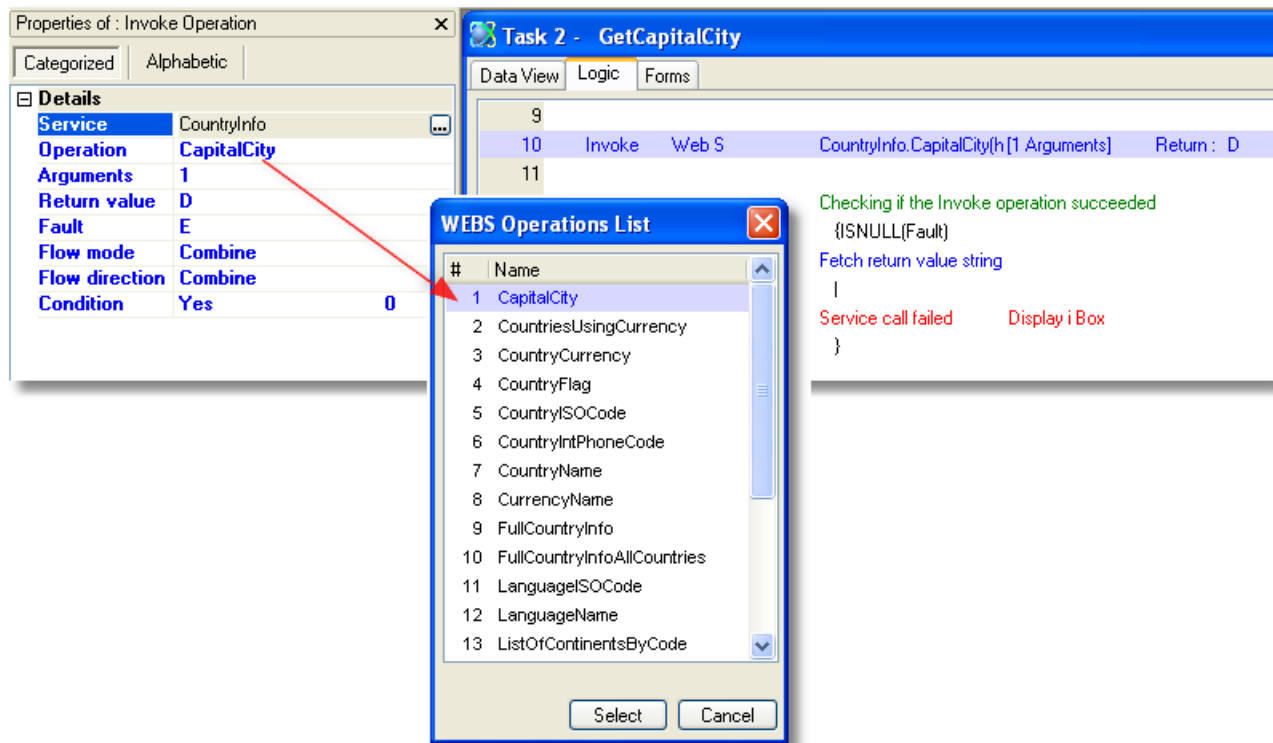


1. Go to **Options->Settings->Services**.
2. Press **F4** (or Edit->Create Line) to open up a line.
3. Give your web service a name. In our example, we used CountryInfo.
4. In the Server column, zoom to select **SOAP**.
5. Press **Alt+Enter** to access the Server properties.

6. In the *WSDL URL* field, enter the URL of the WSDL you are accessing. The Client Module field will fill in automatically.
7. Press the *Load* button. It will take a few moments for eDeveloper to create the Client Module and XML Schema files.

Also in Service Properties, you can set up attachment encoding type, and the security settings to be used.

## Accessing the Web service



Once the Web service is defined, you can access it easily from within eDeveloper.

1. Press **F4 (Edit->Create Line)** to open up a line.
2. Press **/** to select the Invoke operation. The cursor will move to the next field.
3. *Web service* will be selected by default. Tab to the next field.
4. Go to the Properties pane (**Alt+Enter**).
5. Zoom from the *Service* field to select the Service you want. In our example, that is CountryInfo.
6. Zoom from the *Operation* field to select which web service operation you want to access.
7. Zoom from the *Fault* field to select a variable (Blob or Alpha) to hold the error code description (if the operation completes successfully the variable will have an empty value).
8. Zoom from the *Arguments* field to set up the arguments for this Web service. The exact argument list will vary depending on the Web service, but eDeveloper will display a list of what the Web service expects. Often the argument will be a complex type argument represented as XML document, in which case the location of the XML Schema location will be listed.

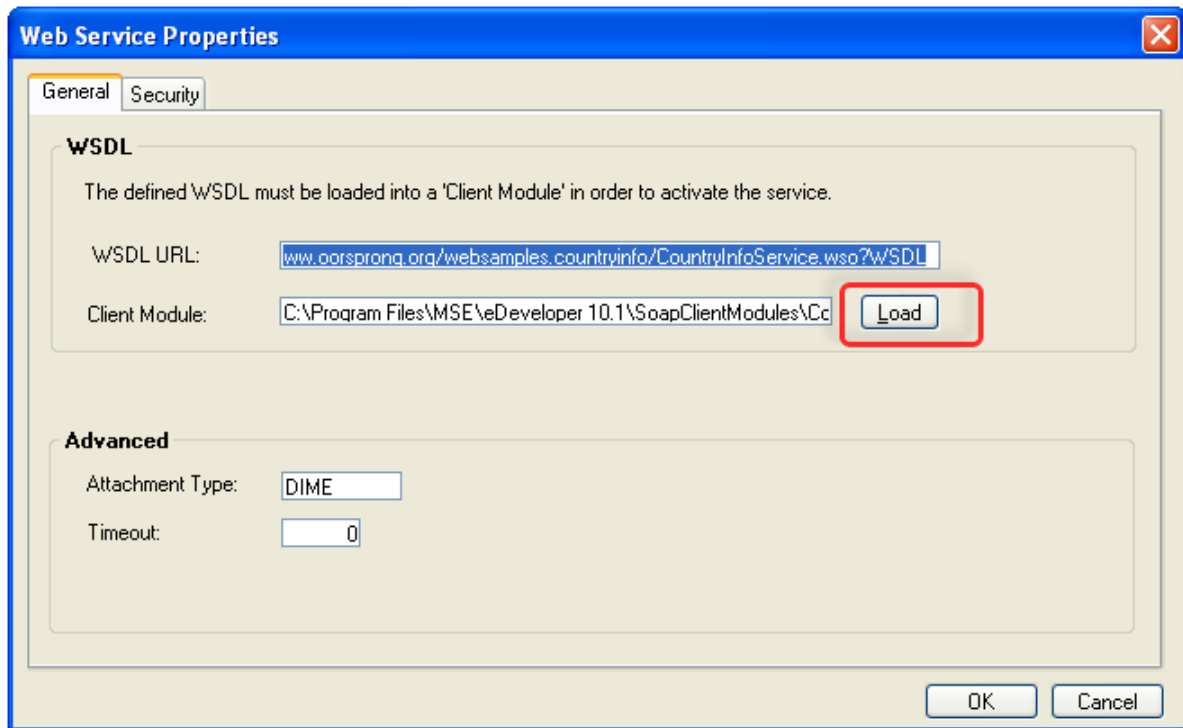
9. Zoom from the *Return value* field to specify a variable for the Returned value from the Web service.

That is all there is to it! If the Web service uses complex input or output, you will need to put together the XML Blob for the Arguments and/or Return value.

## How do I Reload Web Service Definitions?

If a web service changes, you may need to change your application to reflect the changes or take advantage of new services. However, you should not delete and recreate the service, as this will cause existing programs that use the service to break. What you want to do is to reload the service, which is easy to do.

### Reloading an existing service



1. Go to **Options->Settings->Services**.
2. Select the service you want to reload.
3. Open up the Properties by pressing **Alt+Enter**.
4. Press the **Load** button.

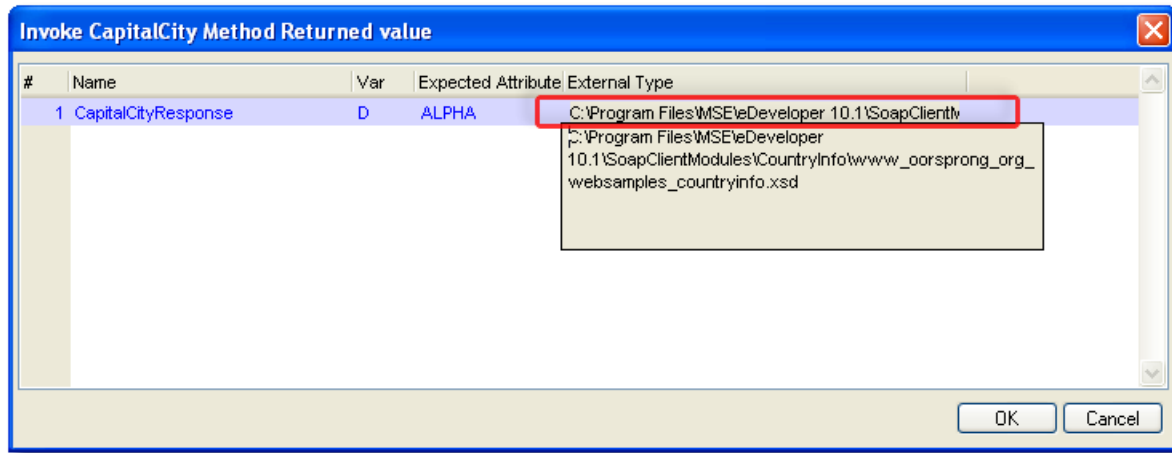
If any XML definitions also changed, you will have to reload the XML data source definitions from the XML Schema also.

## How do I Send or Receive Complex Arguments?

Some SOAP services send very simple data, which can be sent and received using simple eDeveloper variables, which will be translated by eDeveloper as needed. However, many SOAP services are quite complex, and you should use XML data sources to prepare and extract the XML documents.

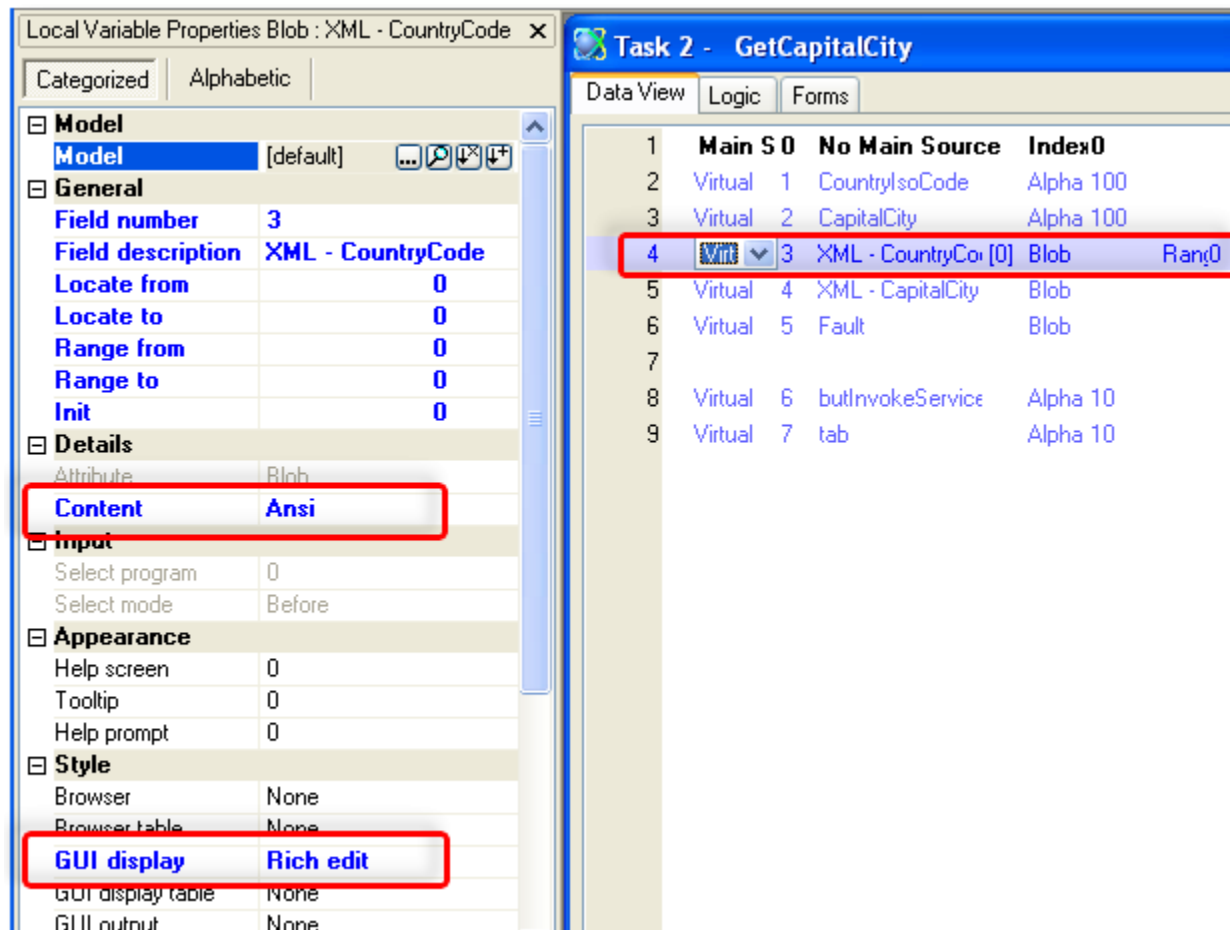
### How to know if a WSDL uses complex arguments

If a WSDL uses complex arguments, you will need to set up XML files to send or receive the data.



You can tell that XML data is expected because the External Type field will contain a path pointing to an XML Schema (XSD) file. The XML Schema file will be created on your computer by eDeveloper, when you declared the Web service. This XML Schema is what you can use to create the expected XML data source, as described in Chapter 14, “How do I Create an XML View?” on page 345.

## Declaring the BLOB Variable



First, you need to declare a variable that will hold the XML document. You would declare this as you would any other variable, only the attribute type is Blob.

The Content type should be Ansi or Unicode, as it will be containing text.

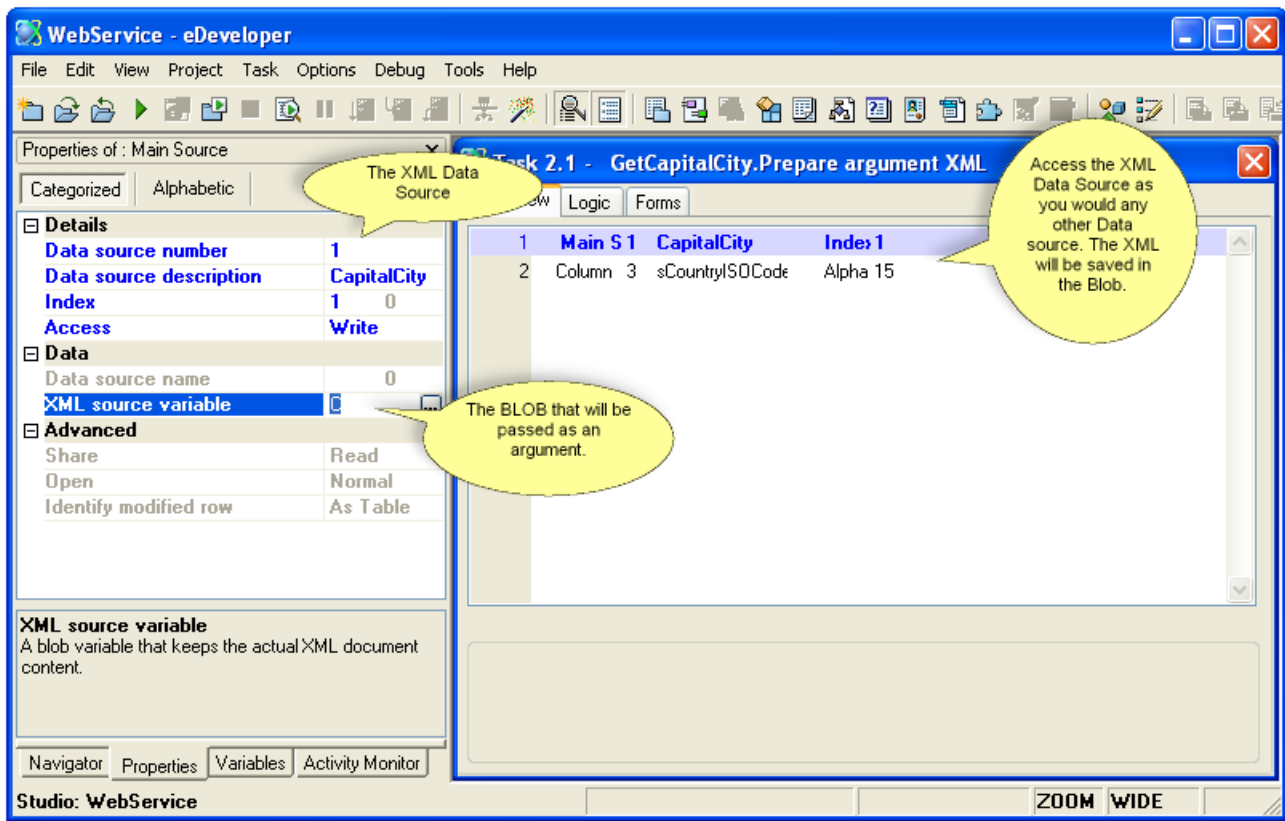
If the GUI Display field is set to Rich edit, then you can easily display the Blob onscreen in a Rich Edit field.

## Creating and Reading the XML Blob

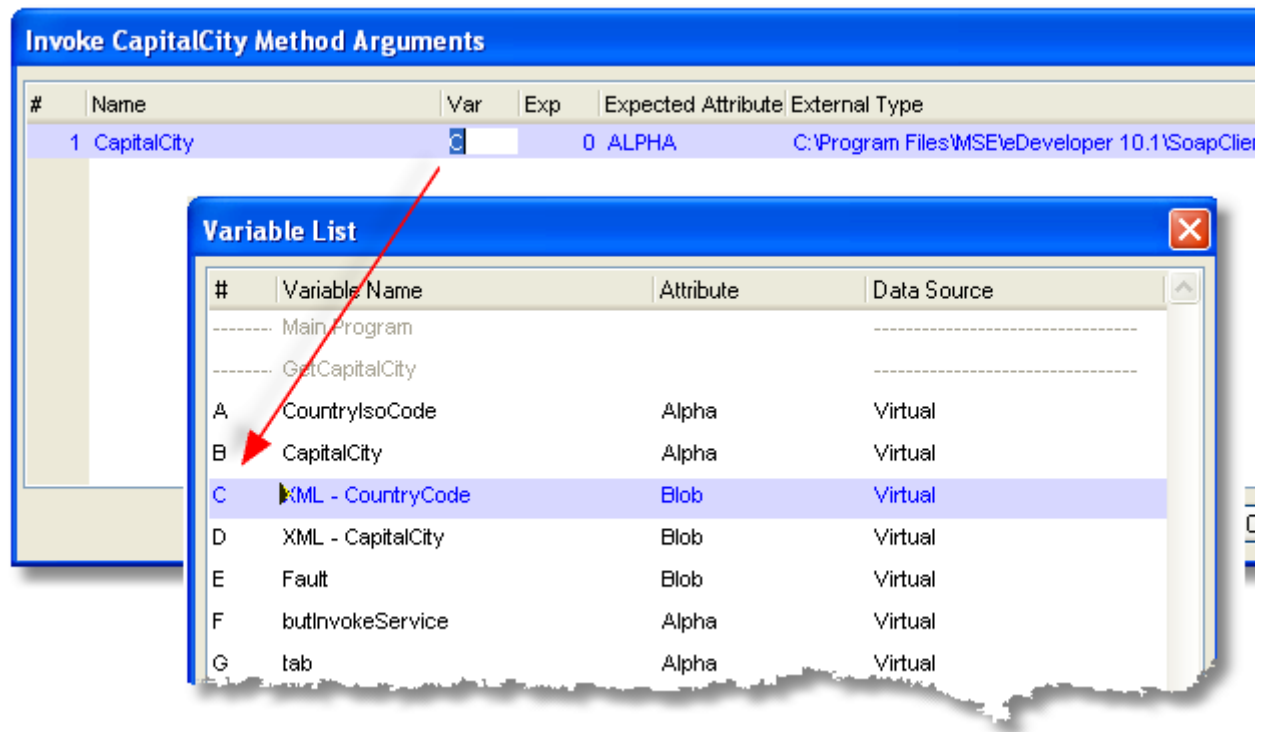
To create or read the XML Blob, you first need to set up an XML Data source, based on the XML Schema. This is explained in more detail in the XML chapter, Chapter 14, "How do I Create an XML View?" on page 345.

Once you have the XML Data source defined, you can move data to the XML file as you would any other Data source.

1. Before you call the subtask to create the XML file, update the Blob variable to NULL() to ensure it is empty.
2. Call a subtask to move data to the Blob. In this subtask, the Main data source will be the XML Data source. The XML source variable will be the XML Blob you are trying to read or write to.
3. Treat the Data source as you would any other data source. If you are only sending one “line” of data, the subtask should only execute once, moving data into one “record”. In our example, there is only one data field to update, the country code, so we move the country code into the country code field and exit.  
Many times the XML document is very simple, involving only a few simple data elements. In this case you would only update the corresponding data elements once.
4. The generation of the XML Blob is automatic from that point. eDeveloper will format the XML based on the XML Data source definition, and move it into the Blob you specified.



## Using the Blob variable as an argument



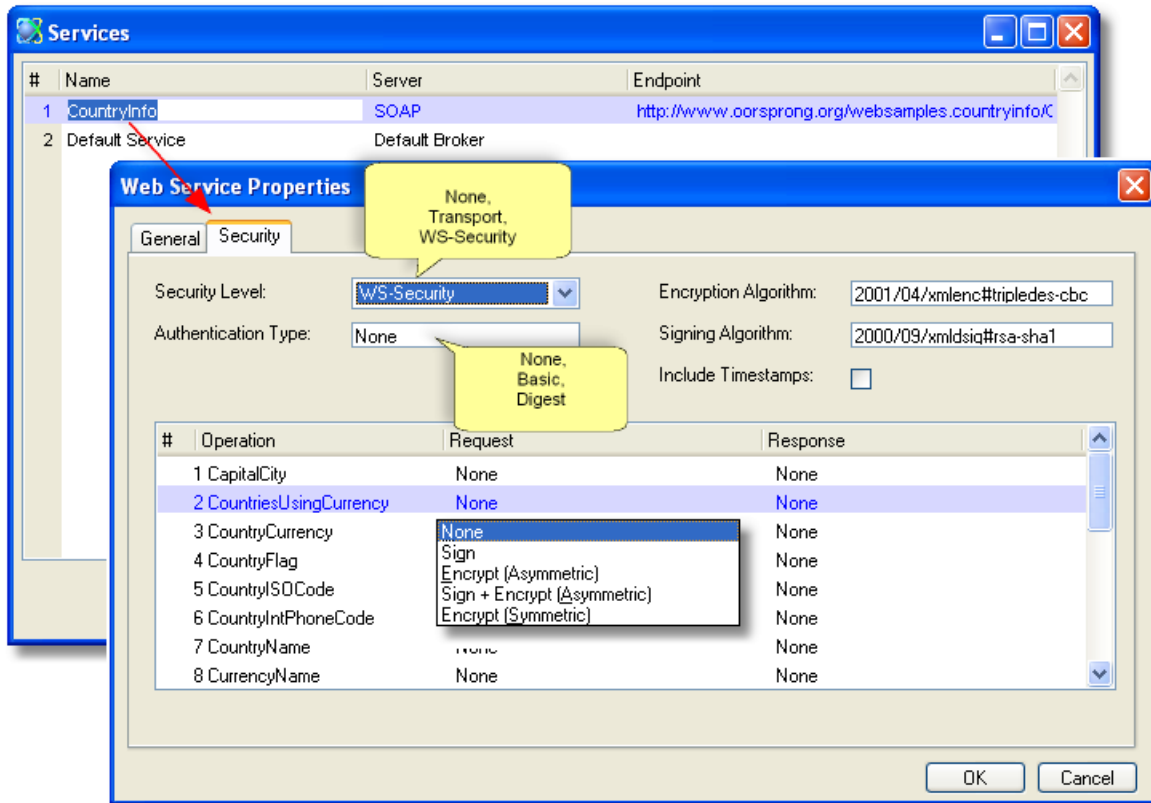
Now, it is a simple matter to assign the Blob variable to the Web service argument.



## How do I Securely Access a Web Service?

When you are accessing a Web service, you will need to set up the required security level (as defined by the Web Service provider) from within eDeveloper. There are two levels at which you will set up the security. First you will set up the security levels for the Service as a whole. Then, when you use the Web service in your program, you can specify the user name and password, if any, that is needed by the Web service.

### Setting Security for the Service

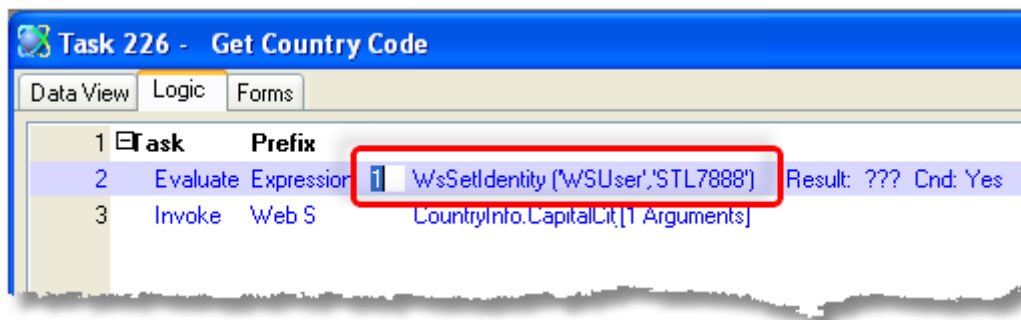


1. Go to **Options->Settings->Services**
2. Go to the Service for which you want to set the security level. Press Alt+Enter to access the Web service properties.
3. Set the security levels to what you need. Below is a table showing the options. If the Security level is set to Transport, then the communication channel is secured. Otherwise, if WS-Security is chosen, then the

messages are secured using encryption, or both encryption and digital signing. If the access point to the service (as defined in the WSDL) is secured (https URL), the communication will be secured as well.

Security Level	Authentication Type	Encryption Algorithm	Operation Security
None	Disabled	Disabled	
Transport	<ul style="list-style-type: none"><li>• Basic</li><li>• Digest</li><li>• SSL</li><li>• Kerberos</li></ul>	Disabled	Disabled
WS-Security	<ul style="list-style-type: none"><li>• None</li><li>• Basic</li><li>• Digest</li></ul>	Several choices for Encryption and Signing algorithms	<ul style="list-style-type: none"><li>• None</li><li>• Sign</li><li>• Encrypt (Asymmetric or Symmetric)</li><li>• Sign+Encrypt (Symmetric)</li></ul>

## Setting Security From the Task



When you access a Web service that requires authentication, your program can identify itself to the Web service provider using the `WsSetIdentity` function. Once the userid and password are set using the `WsSetIdentity` function, the same userid and password are used for all subsequent Invoke Web Service calls, until the function is used again.

**Note:** It is recommended to use variable instead of fixed values to avoid hard coding user credentials in the application.

## How do I Work With Web Service Attachments?

If the Web service has attachments defined in the WSDL, then they will show up in the Web service arguments list, and you can access them as you would any argument with “Expected attribute” set to “Blob” using Blob variable.

If the Web service receives and/or sends an attachments that are not defined in the WSDL, you can set attachments to be sent in the request using the `WsConsumerAttachmentAdd` and access the attachment received in the response using the **`WSConsumerAttachmentGet`** function. The eDeveloper Help shows the details for using these functions.



# Chapter 35: Providing Web Services

## How do I Provide Web Services With eDeveloper?

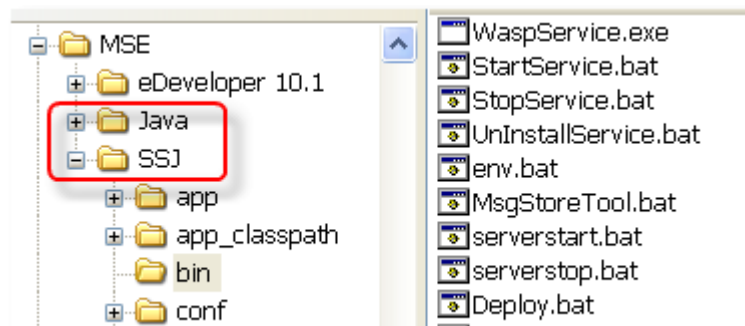
Web Services are the standard way to interoperate between applications on different platforms, frameworks etc. Web services are a good way to provide access to your application from the outside world. Using a web service, various types of applications, written in many different languages, anywhere in the world, can interface with your application in real time. As useful as they are, however, creating a web service manually can be very tedious and time-consuming. Fortunately, eDeveloper automates the creation of the service to make it quite easy.

You can provide Web services from any eDeveloper project. In fact, you create an eDeveloper Web service program as you would any other batch program, using the Parameters and the Return property to pass values in and out. eDeveloper will create the interface so that these can be used in a properly-formed Web service. The steps in providing a Web service are listed below.

## Checking the eDeveloper Setup

Before you can create a Web service, you need to have the appropriate setup for eDeveloper. When you installed eDeveloper, it should have installed the Web services framework.

As part of this installation, eDeveloper will install the Java SDK,

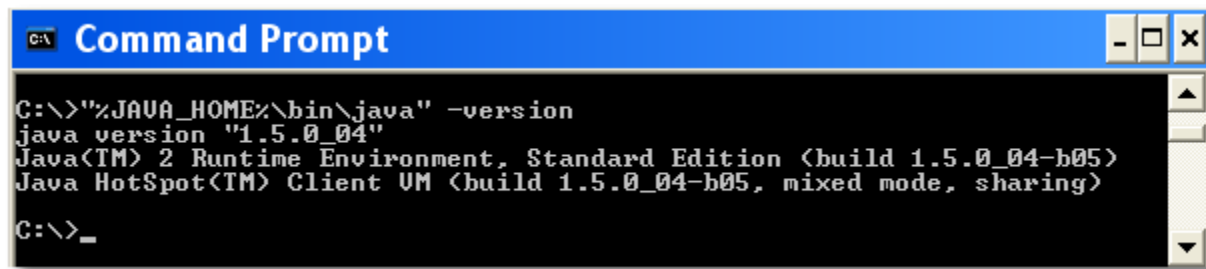


which eDeveloper uses to compile the Java components needed by the SOAP service.

eDeveloper also installs the Systinet Server for Java, which is what you will use to actually deploy the SOAP service.

### Verifying Java

If these products are already installed on your machine, then eDeveloper will use the versions already installed. Otherwise, it will create them as subdirectories in the MSE folder. In any case, to deploy Web services, you need to have these two products installed and operational.

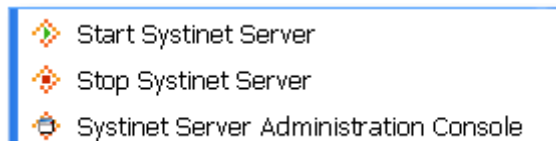


You can check the Java installation by issuing the command

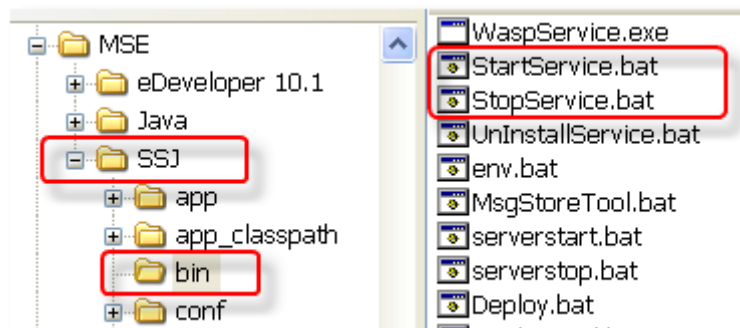
```
"%JAVA_HOME%\bin\java" -version
```

If the JAVA\_HOME environment variable is set correctly, and the Java SDK is installed, you will get a message indicating the Java version.

### Verifying Systinet



When Systinet is installed, you will have some entries on your **Start** menu which will allow you to start and stop the server, and to use the Systinet console.



You can also see the **StartService** and **StopService** .bat files in the SSJ directory, or by running the command line code:

```
"%WASP_HOME%\bin\serverstart.bat"
```

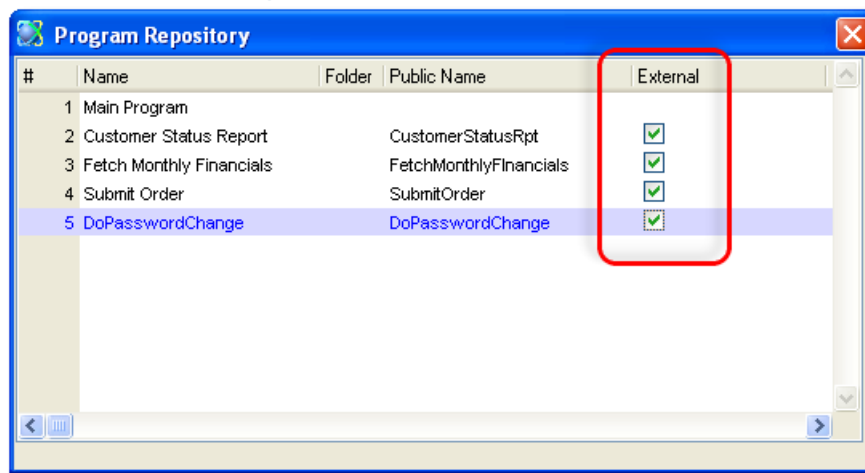
When the service is started, you can also start the console by entering the following in your browser:

```
http://localhost:6060/admin/console
```

We'll talk more about these later, but for now, just verify that they bring up a program.

Once you've validated that everything is installed correctly, you are ready to deploy your Web service.

## 1. Create your batch program



First, you need to make sure the programs you want to expose each have a unique **Public name**, and have the **External** box checked. These programs will be batch programs. They can accept and return values using Parameters in the Data view, and they can also return one value using the **Task Properties->General->Return value** property. These values can be simple data fields, or they can be Blobs containing complex XML data.

## 2. Start the Systinet server

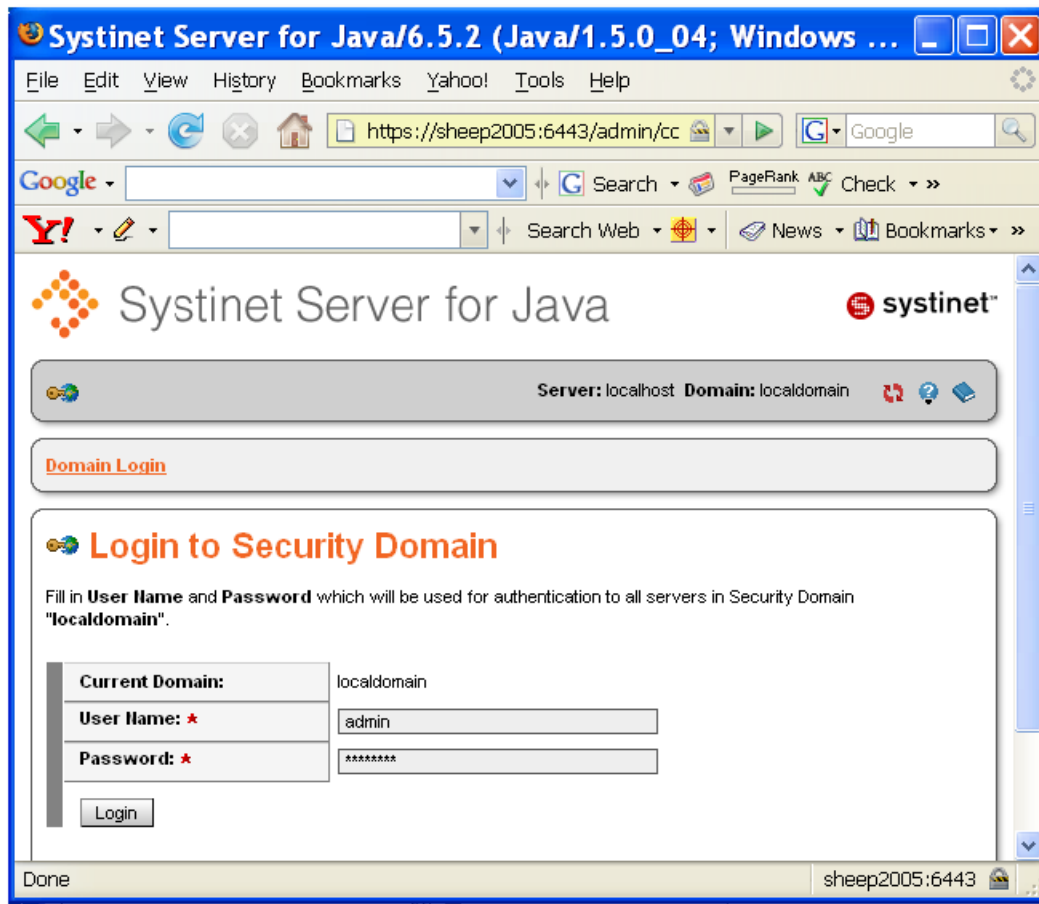
First you need to start Systinet. You can do this from the Start menu, using the Start Systinet option. Or, you can by run

```
%WASP_HOME%\bin\serverstart.bat
```

- From within you eDeveloper application, select **Options->Interface Builder->Web Service**. You will see a screen, "Welcome to the Web Service Interface Builder". Click Next

Next, start the console. You do this in your browser, by entering the URL:

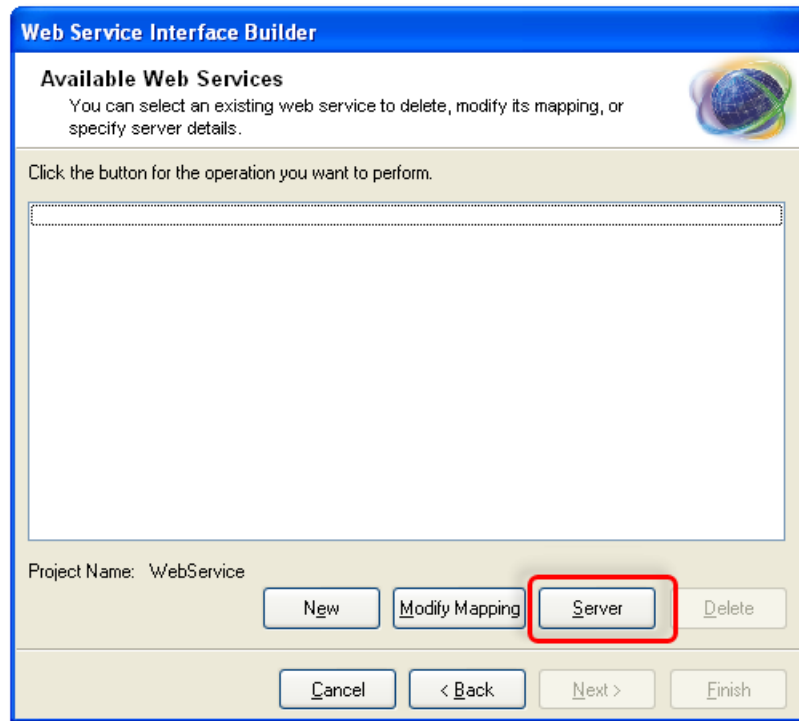
`http://localhost:6060/admin/console`



The default user is *admin*, and the default password is *changit*.



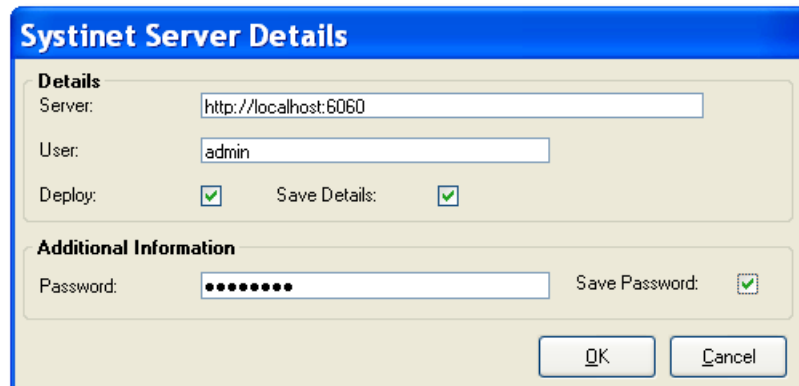
### 3. Start the Webservice Interface Builder



From within the eDeveloper Studio, select **Options->Interface Builder->Web Service**. You will then see the Web Service Interface Builder start screen.

Now you'll see a list of available Web services. If you haven't created any yet, this list will be empty. Click on the **Server** button.

### 4. Server Details



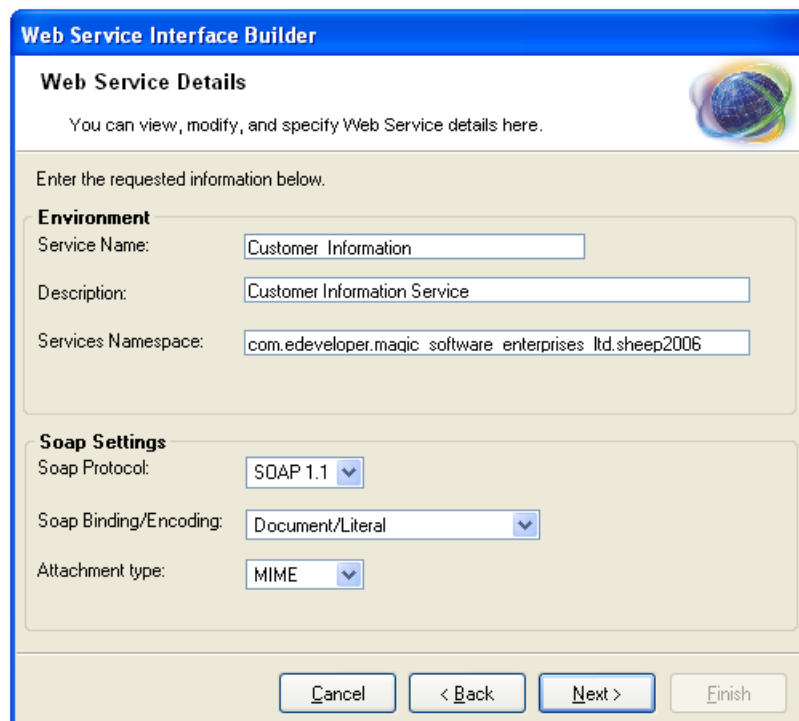
For the server details, you need to enter the location and login details. The default Systinet settings are to localhost:**6060**, user **admin**, and a password of **changeit**.

Note: After installing eDeveloper you are required to enter the password and to avoid entering it each time you define a service you should check the “Save Password” checkbox.

If you check **Deploy**, the the service will be automatically deployed after it is produced.

Now, from the Available Web Services list, select **New**. You will be presented with the Web Service Details screen.

## 5. Web Service Details



The screenshot shows the 'Web Service Interface Builder' window with the 'Web Service Details' tab selected. The window has a blue title bar and a light beige background. At the top right is a small globe icon. Below the title bar, the text 'Web Service Details' is followed by 'You can view, modify, and specify Web Service details here.' Below this is the instruction 'Enter the requested information below.' The form is divided into two sections: 'Environment' and 'Soap Settings'. The 'Environment' section contains three text input fields: 'Service Name' (filled with 'Customer Information'), 'Description' (filled with 'Customer Information Service'), and 'Services Namespace' (filled with 'com.edeveloper.magic software enterprises ltd.sheep2006'). The 'Soap Settings' section contains three dropdown menus: 'Soap Protocol' (set to 'SOAP 1.1'), 'Soap Binding/Encoding' (set to 'Document/Literal'), and 'Attachment type' (set to 'MIME'). At the bottom of the window are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

Here you provide the **Service name**, **Description**, and **Namespace**. These should be text that is meaningful to the user.

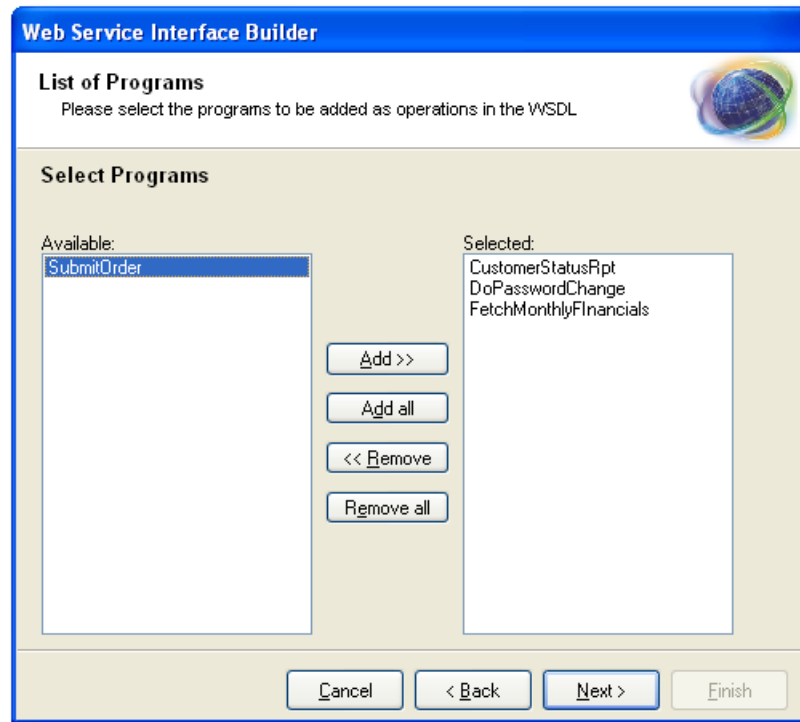
For **Soap Protocol**, SOAP 1.1. is the most compatible with most clients.

**Soap Binding/Encoding** can be Document/Literal, wrapped/literal, RPC/literal, or RPC/Encoded.

**Attachment type** can be MIME or DIME. MIME is the most compatible with most clients.

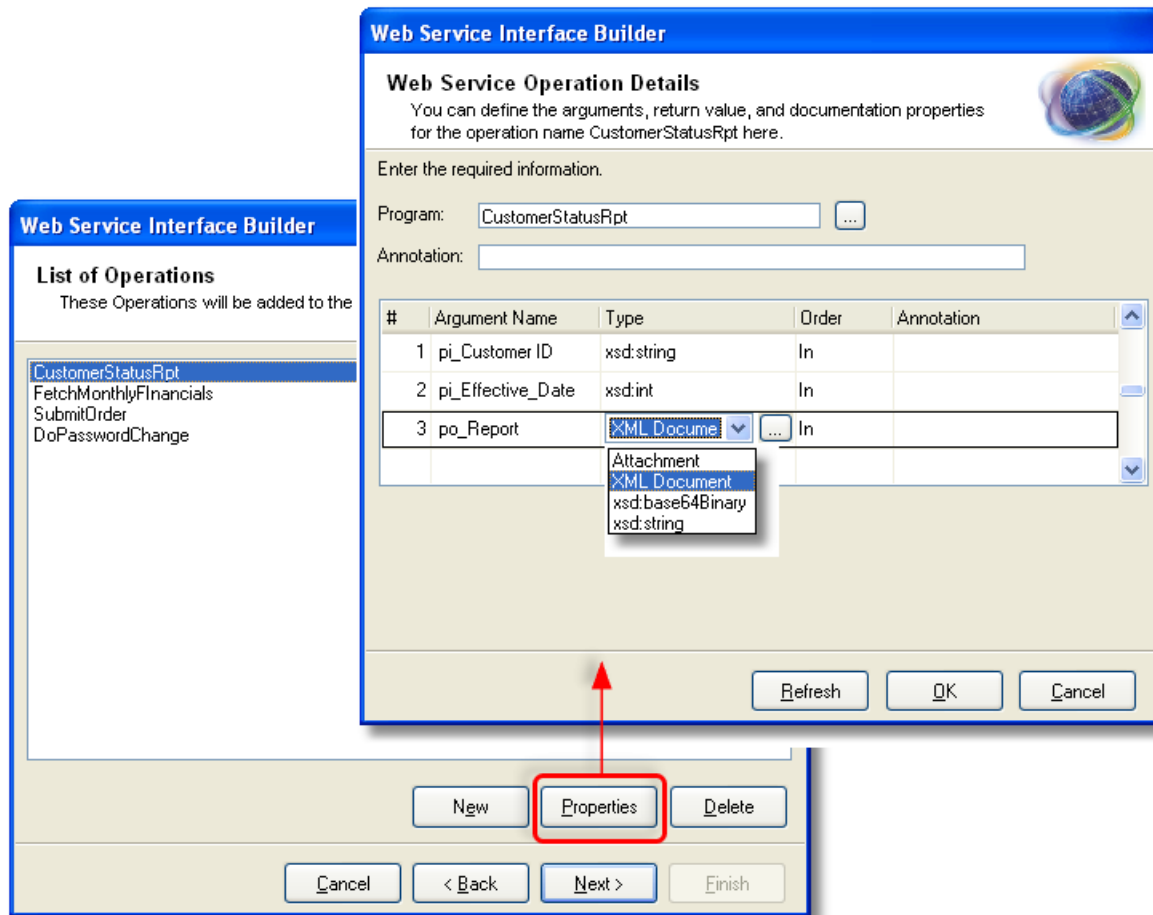
When you are finished, press **Next**.

## 6. Select Programs



You will be presented with a list of all the batch programs which are defined with Public names and with the External box checked. You can use the buttons in the middle to move the programs you want to expose in services to the Selected column. Then press **Next**. You will be presented with a list of each of the programs you selected.

## 7. List of Operations

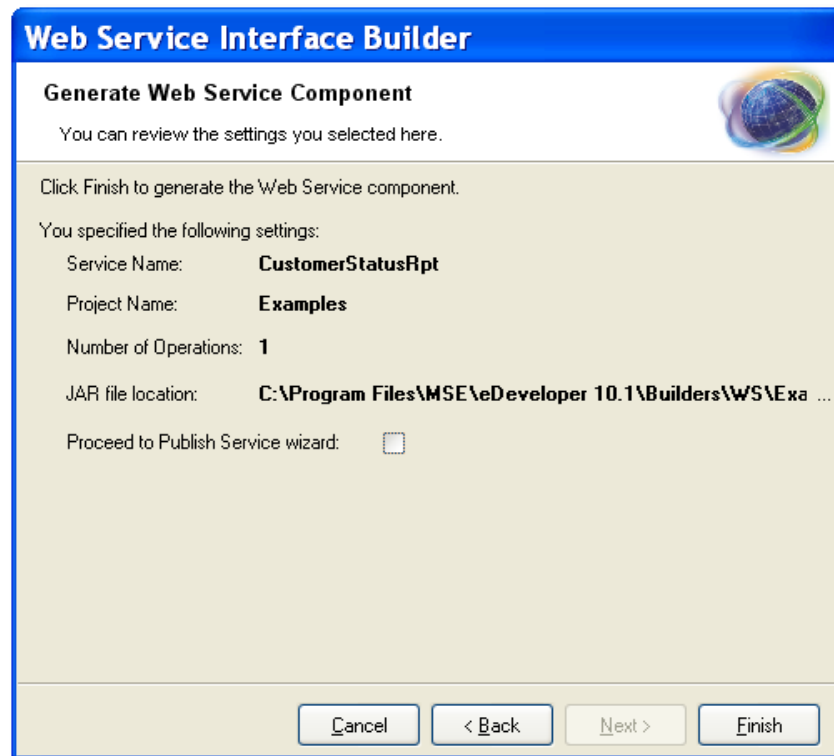


Now, each of the programs you chose will show up on the List of operations. For each of these, you can view and change the Properties, or Operation details. These affect how the Web service appears to the consumer of the service.

If the argument is of type Alpha, Unicode or Blob and it represents a complex type, then if you select XML Document here, you can click the ellipse button to select an XSD to describe the XML.

When you are finished, press **Next**.

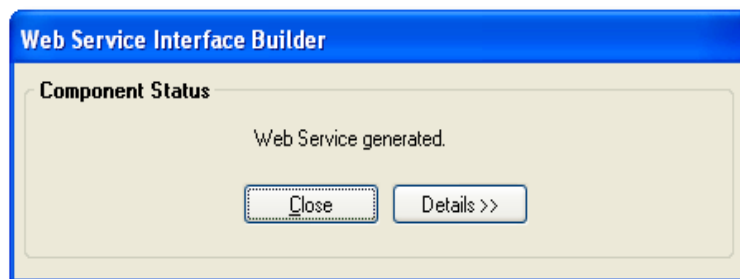
## 8. Summary screen



You'll be presented with a summary screen. Press **Finish** to continue.

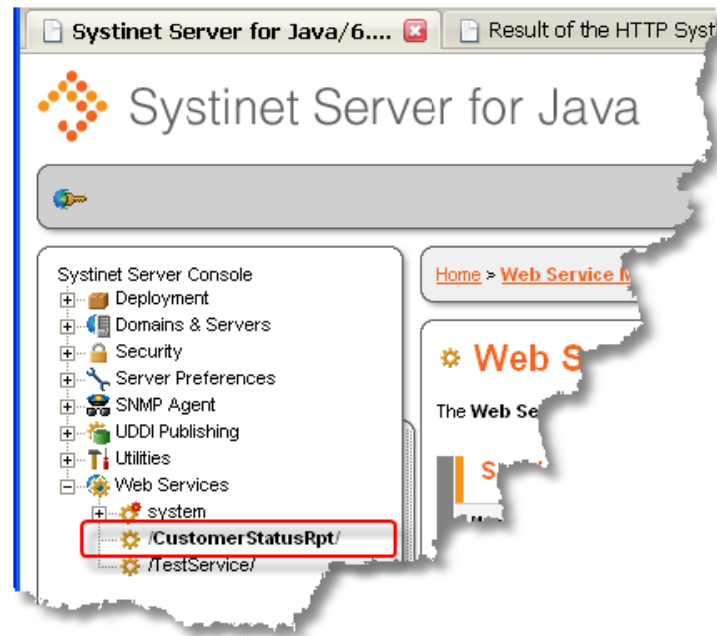
The “Publish” checkbox should not be checked unless you would like to publish the service in a UDDI registry after it is deployed.

## 9. Finished



When the process is complete, you will get a status screen. If there were errors, you will get an error message and you can see the error details by pressing the Details button. If there were no errors, the Details button will show you the JAR file location. You can copy and paste the JAR file location to use later, if you want to manually deploy the service.

## 10. Deployment



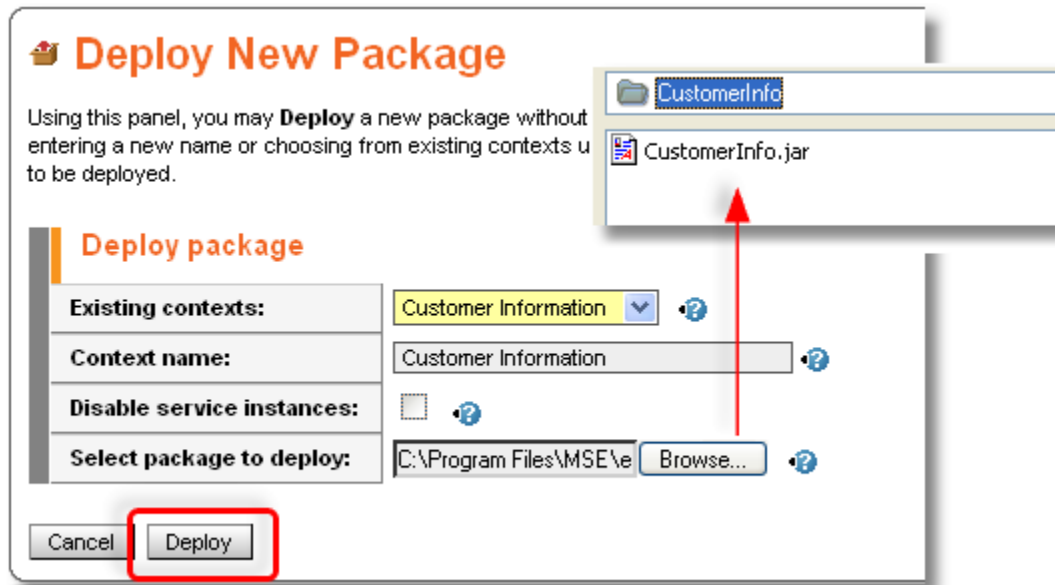
If you checked the Deploy box in “4. Server Details” on page 837, then the service will be automatically deployed, and you’ll be able to see it and test it using the Systinet console, as described in “How do I Test a Web Service?” on page 843..

Otherwise, you can manually deploy the service, as described in Chapter 35, “How do I Deploy a Web Service Module?” on page 842.

### How do I Deploy a Web Service Module?

When you create a Web service in eDeveloper, you can have eDeveloper deploy it within Systinet automatically by checking the Deploy box on the Server Details screen, as shown in Chapter 35, “” on page 833.

Alternatively, you can deploy the module manually from within Systinet, by doing the following steps:



1. Open the Systinet console
2. Click on **Deployment->Deploy New Package**
3. Fill in the context and context name.
4. Select the package to deploy. This is the .jar file created by eDeveloper.
5. Click on the Deploy button.

Now the Web service is available for use.

## How do I Test a Web Service?

You can test a Web Service by writing a program to access it, but you can also test it directly from the Systinet management console.

1. Open the **Systinet console**.
2. Under the **Web Services** node to the left, select the Web service you want to test.

3. A number of choices will appear in the pane to the right. Scroll down a bit and click on the **Invocation Console** button.



### Web Invocation Console

#### Service URL:

http://AMIRLP:6060/Calculator/

#### Operation:

Service	Port	Operation	Style	Encoding
CalculatorImpl	CalculatorImpl	Calculator(string, int, int)	doc	literal

P_Operation: <b>string</b>	<input checked="" type="checkbox"/>	+
P_num1: <b>int</b>	<input checked="" type="checkbox"/>	7
P_num2: <b>int</b>	<input checked="" type="checkbox"/>	4

**Perform call**



4. A new window will appear. Here you will see fields to enter test values for the service. Make sure the eDeveloper runtime engine is running, then click on the **Perform call** button.

### Input message

Following message was sent to the service at <http://AMIRLP:6060/Calculator/>

```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:e="http://schemas.xmlsoap.org/soap/envelope/">
  <e:Body>
    <ns0:P_Operation xmlns:ns0="http://systinet.com/xsd/SchemaTypes/">+</ns0:P_Operation>
    <ns0:P_num1 xmlns:ns0="http://systinet.com/xsd/SchemaTypes/">7</ns0:P_num1>
    <ns0:P_num2 xmlns:ns0="http://systinet.com/xsd/SchemaTypes/">4</ns0:P_num2>
  </e:Body>
</e:Envelope>
```

### Output message

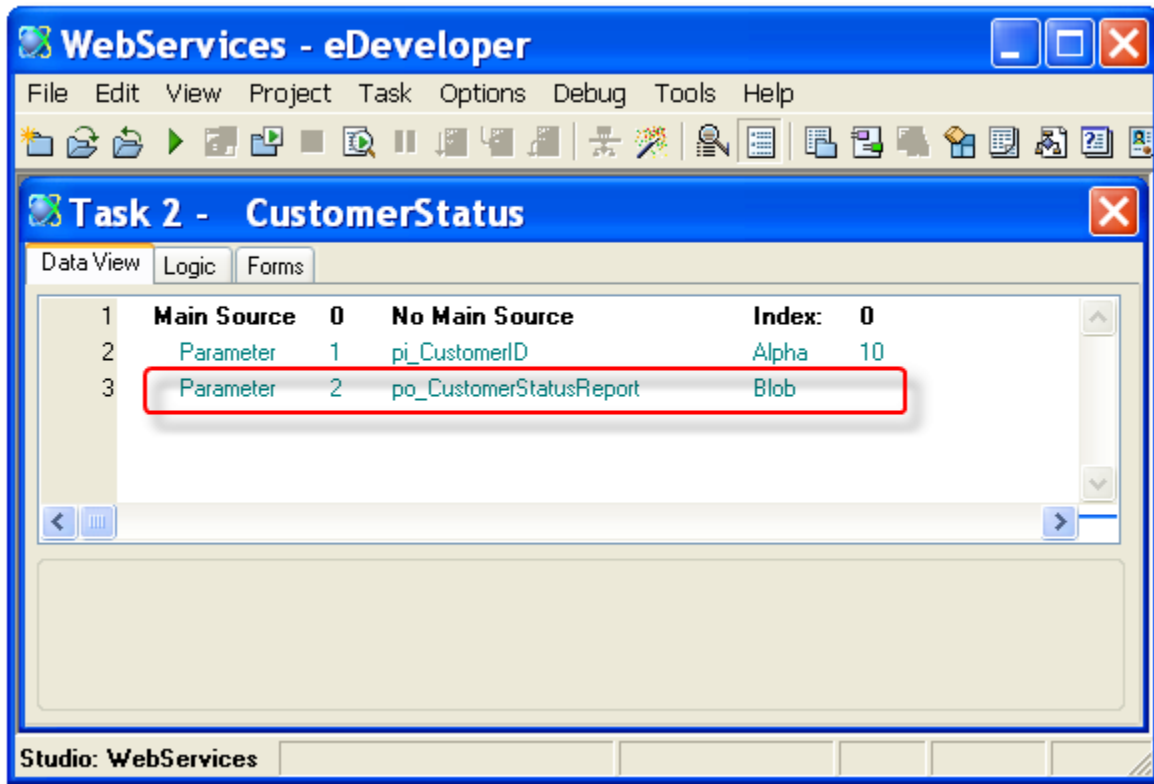
Following message has been received

```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:d="http://www.w3.org/2001/XMLSchema" xmlns:e="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns:vm0="http://systinet.com/xsd/SchemaTypes/"
  xmlns:vm1="http://idoox.com/interface">
  <e:Body>
    <vm0:int_Response i:type="d:int">11</vm0:int_Response>
  </e:Body>
</e:Envelope>
```

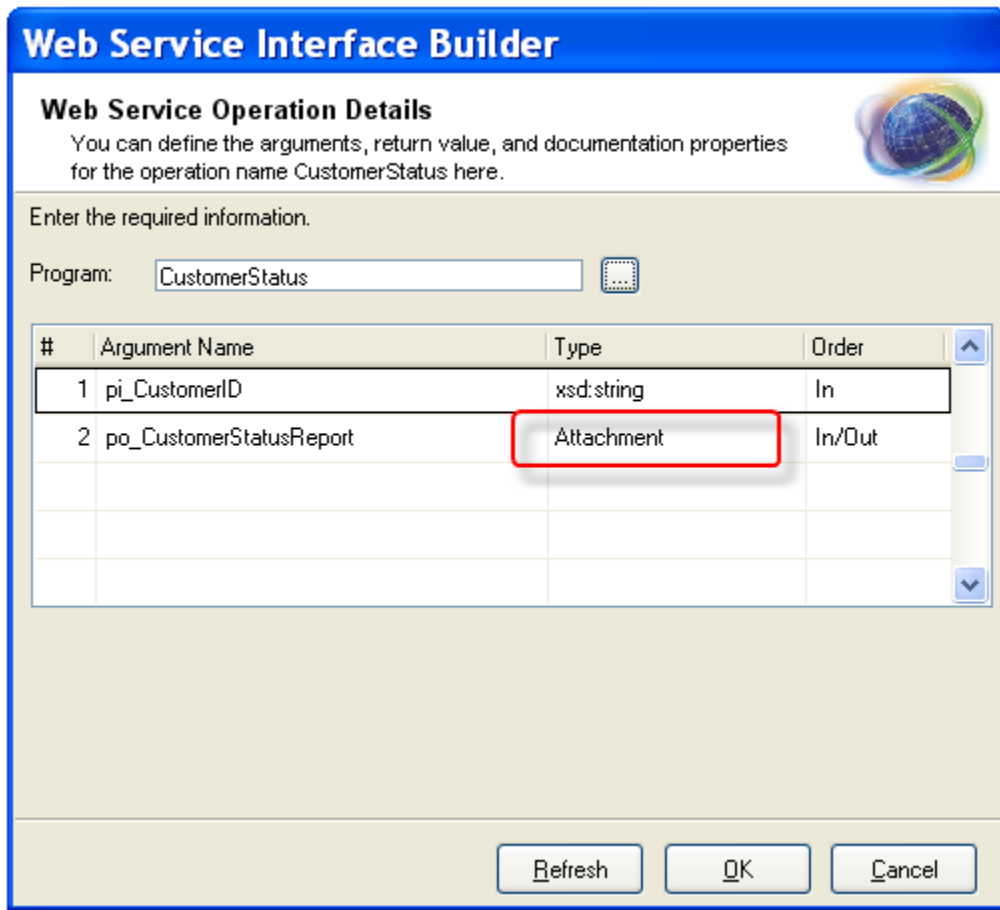
Return to: [Service List](#)  
[Operation List](#)  
[Operation Invocation](#)

5. You will see the input and output calls displayed. You can see our two input, 7 and 4, above, and the output, 11, below.

## How do I Provide Web Services With Attachments?



You can send and receive files in a Web service by encapsulating them in a Blob parameter. You can use the function Blb2File: **Blb2File (<Blob>,<File Name>).**



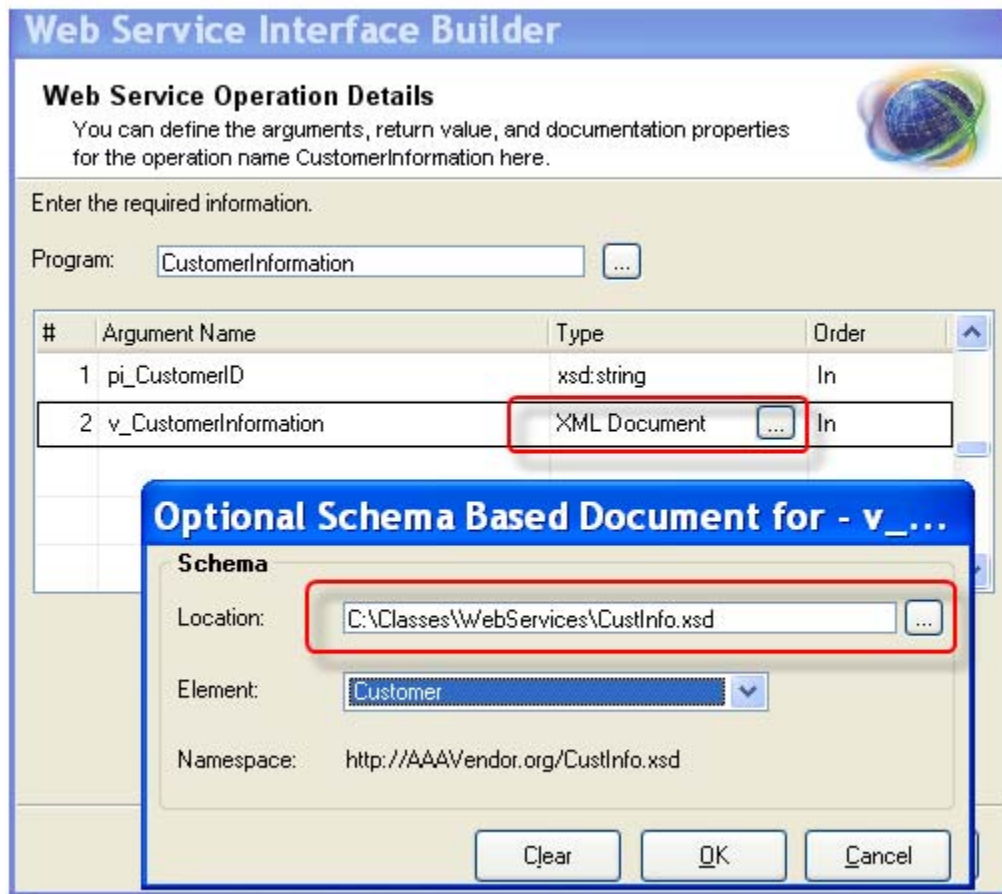
The image shows a 'Web Service Interface Builder' dialog box. It has a blue title bar and a white header area. The header contains the title 'Web Service Interface Builder' and a sub-header 'Web Service Operation Details' with a small globe icon. Below the header, there is a text area with the instruction: 'You can define the arguments, return value, and documentation properties for the operation name CustomerStatus here.' and 'Enter the required information.' Below this, there is a 'Program:' label followed by a text box containing 'CustomerStatus' and a small icon. The main part of the dialog is a table with four columns: '#', 'Argument Name', 'Type', and 'Order'. The table has two rows. The first row has values: '1', 'pi\_CustomerID', 'xsd:string', and 'In'. The second row has values: '2', 'po\_CustomerStatusReport', 'Attachment', and 'In/Out'. The 'Attachment' text in the second row is highlighted with a red rectangle. At the bottom of the dialog, there are three buttons: 'Refresh', 'OK', and 'Cancel'.

#	Argument Name	Type	Order
1	pi_CustomerID	xsd:string	In
2	po_CustomerStatusReport	Attachment	In/Out

Then, when you are building the Web service, set the parameter type as **Attachment**.

Now the file will come across as an attachment.

## How do I Send or Receive Complex Arguments From a Web Service as a Provider?



To send complex arguments in a Web service, you need to use an XML file. You can do this as follows:

1. Create the XML schema (XSD) that will be used to describe the XML file.  
The XML schema must include a namespace definition (targetNamespace). To avoid interoperability problems the namespaces of schemas of different arguments/return value must differ (i.e. do not use the same schema for all arguments of the same operation)
2. Use this XML schema to create the XML Data sources, which you will use to format the data to send (this is covered in Chapter 14, "How do I Create an XML Doc from Scratch?" on page 341).
3. When you are creating your Web service, make one of the parameters a Blob data type, and use this to hold the XML data.
4. When you are creating your Web service, select XML Document as the Type for this parameter. Click on the ... to bring up the schema. Choose the same XML schema file you used in step 2.
5. Press OK.

Now, your WSDL file will contain the XML schema needed by the consumer.

To access the Service WSDL use the Systinet Web Console (<http://localhost:6060/admin/console>).

Locate the service in the “Web Service Runtime View” screen (opened by clicking on the “Web Service” entry in the tree view on the left side of the console screen) The WSDL URL is in a hyper link named “Url” and it is in the form:

```
http://servername:6060/ServiceName
```

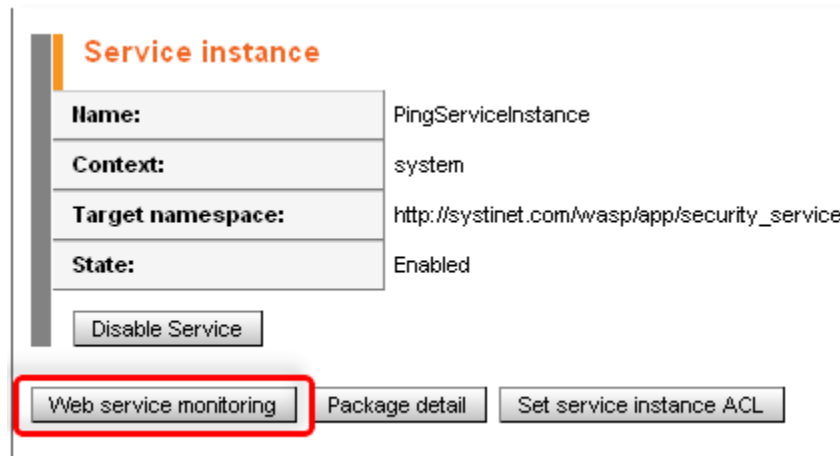
## How do I Trace SOAP Requests?

Sometimes it is useful to trace the SOAP requests that come in to your service. You can do this from the Systinet console.

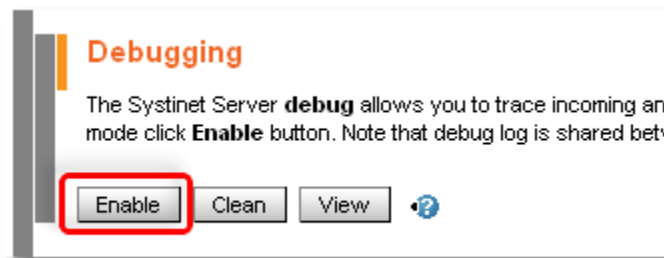
1. Start the Systinet console, using the shortcut or the URL

```
http://localhost:6060/admin/console
```

2. Expand the **Web Services** node on the tree on the left.



3. Select the service you want to trace.
4. Click the Web service monitoring button.



5. Then, scroll to the **Debugging** section. Click **Enable**. Now the Web service is under watch
6. After each call (or any activity) press the View button to see the HTTP and SOAP activity. Access to this page is username and password protected: the username and password are the same as your Systinet login.

## How do I set up Authorization for a Deployed Service?

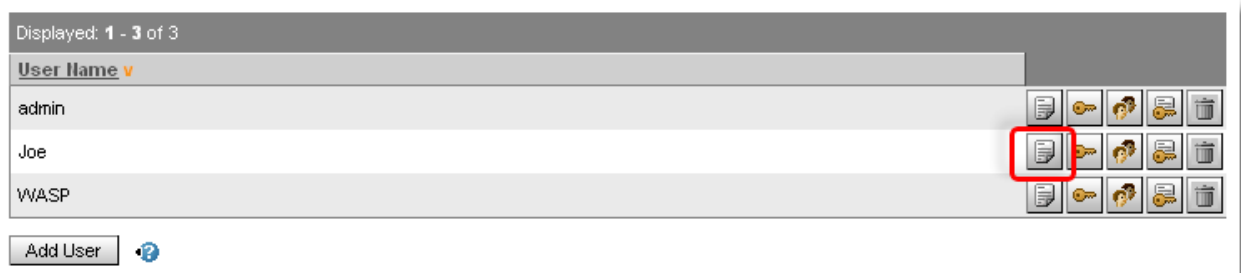
For any service you are deploying, you have the capability to allow only specific users access to the service. This is done in three steps:


- First, you must define the users.
- Second, you must define which users have access to the Web service.
- Third, you must define which authentication method will be required to access the service.

These two steps are explained below.

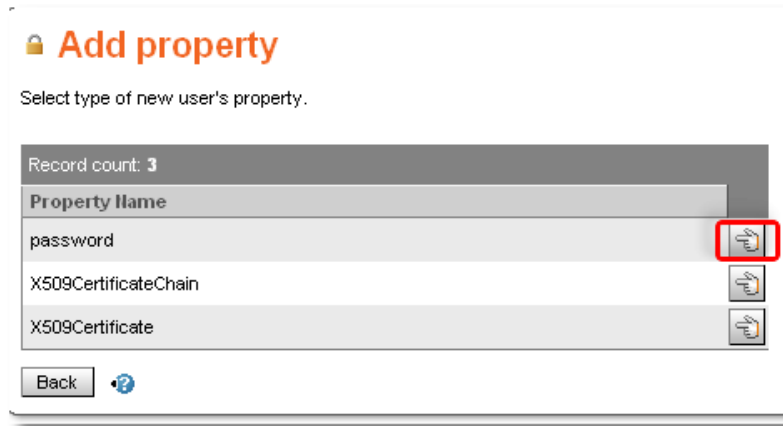
### Defining the User and Password




1. Open the Systinet console.
2. Select **Security->Database of Identities** on the tree to the left.




3. You will see a list of existing users on the right. Click on the **Add User** button below the table.
4. You will get a box prompting you for a user name. Enter the name, and press Add User.
5. Now you will see the new user on the user table. You can use the buttons to the right of each user id to manage this user.
6. To set the password, first click on the  first button.

7. A list of properties for this user will appear. Click the **Add property** button.



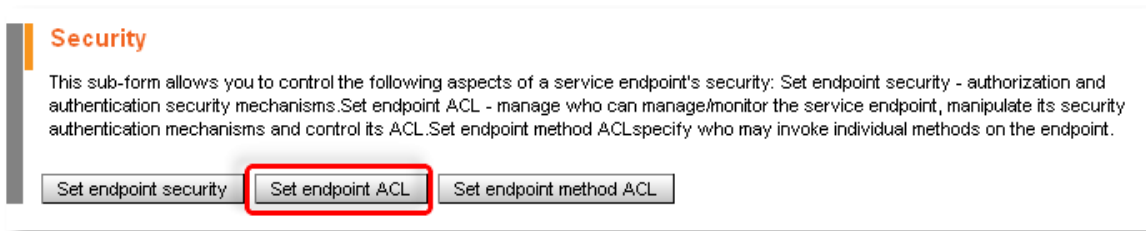
Property Name	
password	
X509CertificateChain	
X509Certificate	

8. You will see a list of properties. One of them is password. Click on the  to the right of the password property.
9. You will be prompted to enter the new password, and to confirm it. Then press the Enter property button.

## Allowing the User Access to the Service

Now, you need to change the Service to be authenticated and define the user as allowed to consume it.

1. Select **Web Services** from the tree on the left.
2. Select the Service you want to work with.



**Security**

This sub-form allows you to control the following aspects of a service endpoint's security: Set endpoint security - authorization and authentication security mechanisms. Set endpoint ACL - manage who can manage/monitor the service endpoint, manipulate its security authentication mechanisms and control its ACL. Set endpoint method ACL specify who may invoke individual methods on the endpoint.

3. Select **Set Endpoint Method ACL** on the right, in the Security section.

Record count: 4

Operation Name	Set ACL
getServerInfo	<a href="#">Set ACL</a>
getVersion	<a href="#">Set ACL</a>
isAlive	<a href="#">Set ACL</a>
*	<a href="#">Set ACL</a>

Here you can *grant* or *revoke* permission to users.

Record count: 3

User v	State	
admin	inherited	<a href="#">revoke</a>
Joe	revoked	<a href="#">grant</a>
WASP	revoked	<a href="#">grant</a>

4. Each operation will be shown in a separate line on the table. Click on Set ACL to set authentication for each operation.

When you click on Set ACL, another window will appear, with a list of users and what rights they have. Clicking on **grant** will give them access: clicking on **revoke** will revoke access.

## Defining the Service Authentication method

Last, you need to define this service as Authorized.

1. Select **Web Services** from the tree on the left.
2. Select the Service you want to work with.

**Security**

This sub-form allows you to control the following aspects of a service endpoint's security: Set endpoint security - authorization and authentication security mechanisms. Set endpoint ACL - manage who can manage/monitor the service endpoint, manipulate its security authentication mechanisms and control its ACL. Set endpoint method ACL specify who may invoke individual methods on the endpoint.


[Set endpoint security](#) [Set endpoint ACL](#) [Set endpoint method ACL](#)



3. From the Security section, click on [Set endpoint security](#).

**Service Security Settings**

Endpoint:

Authorization Required: ☒ 

Note that there are global security settings that are applied as a default for service endpoints. The preferences you set on this form have higher precedence than these global default settings. To accept the global default security settings, click the **Use Default Security Providers** button.

Choose **Accepting Security Providers** for this service endpoint:


Accepting Security Providers:

Record count: 6

Name v	Accepting	
HttpBasic	<input type="checkbox"/>	N/A
HttpDigest	<input checked="" type="checkbox"/>	<a href="#">Properties</a>
Kerberos	<input type="checkbox"/>	<a href="#">Properties</a>
Siteminder	<input checked="" type="checkbox"/>	N/A
SSL	<input type="checkbox"/>	N/A
WS-Security	<input type="checkbox"/>	<a href="#">Properties</a>

Choose **Initiating Security Provider** for this service endpoint:

**Initiating Security Provider Settings**

Initiating Security Provider:  

4. You will then be on the Authentication and Authorization page. Select the [Authorization required](#) checkbox. Check whatever security providers you want to accept. Then click [Save Changes](#).

That is all that is needed on the provider's side. If you are accessing this Web service, you will need to provide the authentication that was specified here. How to do this is explained in Chapter 34, "How do I Securely Access a Web Service?" on page 829.



## Chapter 36: Database

---

### How do I Define a Connection to a Database?

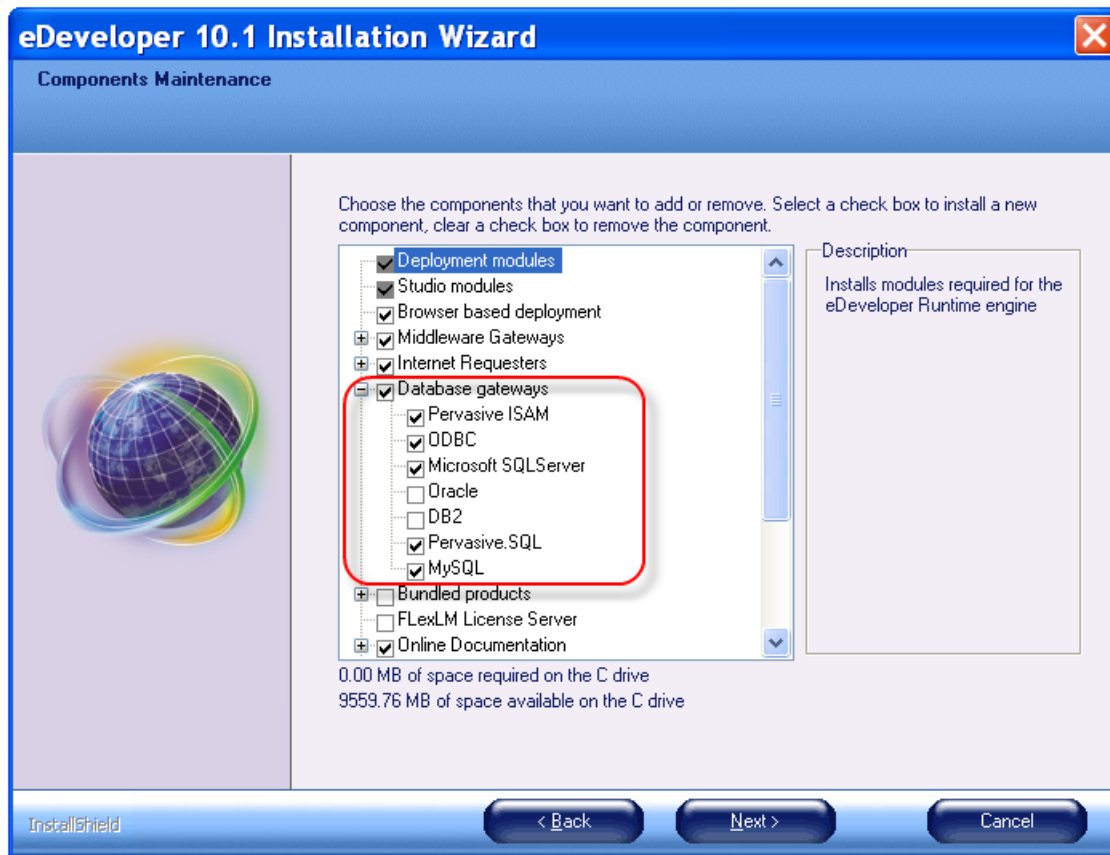
eDeveloper is very good at connecting to multiple databases, including Oracle, SQLServer, DB2, ODBC, MySQL, AS400, DB2/400 and Pervasive. You can even link to multiple databases located on multiple servers, simultaneously. Before you use the databases though, you need to define them to eDeveloper.

This is done in several steps:

- A. Checking the Database Gateway
- B. Defining the Database
- C. Checking the Connection

Each of these steps is explained below. In addition, the connection is defined slightly differently depending on the database involved. These differences are also explained below.

## A. Checking the Database Gateway



1. When you install eDeveloper, you need to select the Database gateways that you might use in the future, so the appropriate DLLs are installed. If you didn't install them when you first installed eDeveloper, you can use **Start->Control Panel->Add or Remove Programs**, selecting eDeveloper, and pressing the Repair/Modify button, or, use the installation disk. Don't uncheck any existing Database gateways, but add checkmarks for the new ones you want to install.

```

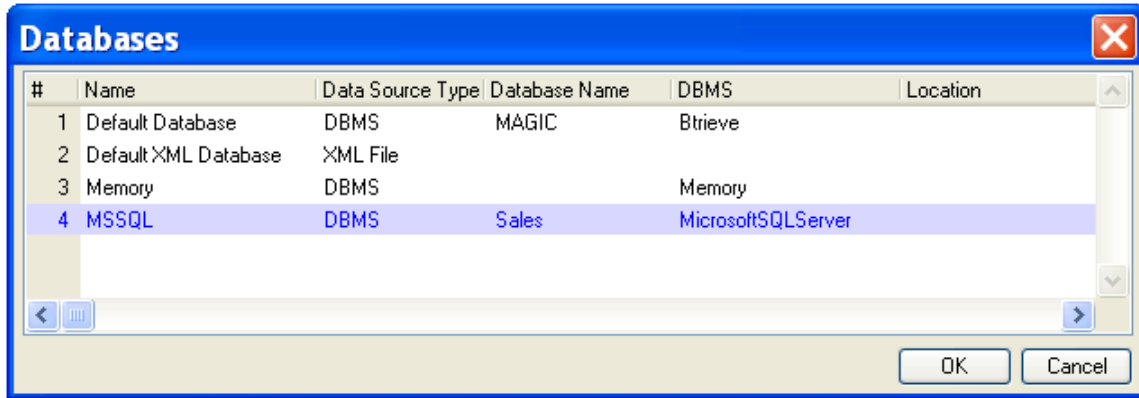
196 [MAGIC_GATEWAYS]
197 ;MGCOMM01=mgwsock.dll
198 MGDB00=C:\Program Files\MSE\eDeveloper 10.1\Gateways\MGBtrieve.dll
199 MGDB01=C:\Program Files\MSE\eDeveloper 10.1\Gateways\MGPervasiveSQL.dll
200 MGDB03=C:\Program Files\MSE\eDeveloper 10.1\Gateways\MGMySQL.dll
201 ;MGDB06=mgdb2400.DLL
202 ;MGDB13=mgOracle.dll
203 ;MGDB16=mgcac32.dll
204 ;MGDB18=mgdb2.DLL
205 MGDB19=C:\Program Files\MSE\eDeveloper 10.1\Gateways\mgodbc.dll
206 MGDB20=C:\Program Files\MSE\eDeveloper 10.1\Gateways\mgmssql.dll
207 MGDB21=C:\Program Files\MSE\eDeveloper 10.1\Gateways\mgmemory.dll
208

```

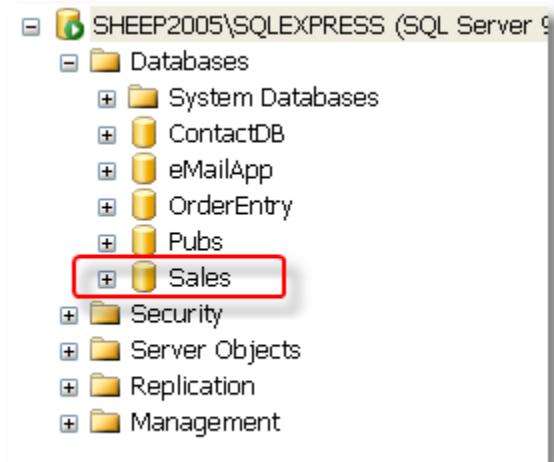
- Now, in the **Magic.ini** file you are using, you need to make sure the appropriate Databases are un-commented and that the path is correct.
- After you edit the INI, start eDeveloper.

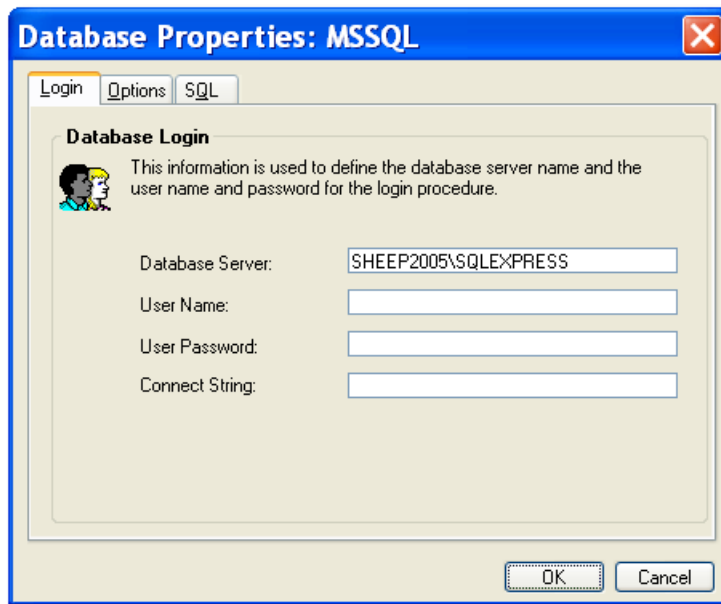
Now you are ready to define the database.

## B. Defining the Database



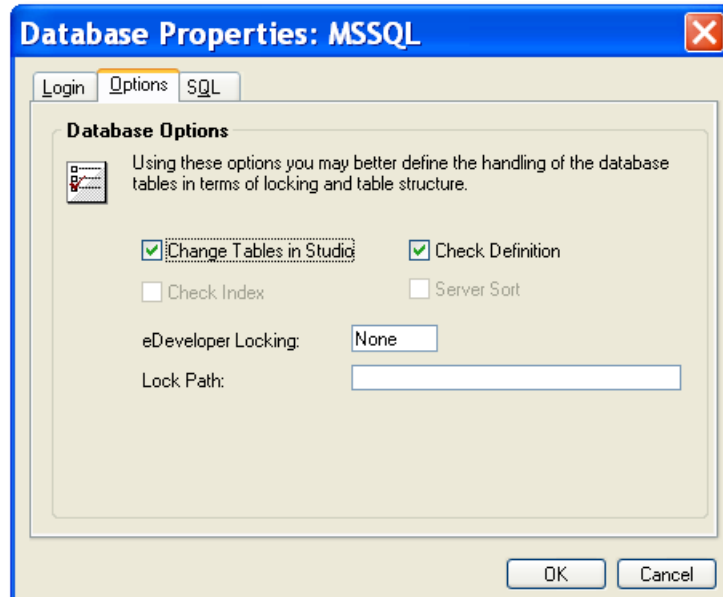
- In the **Name** column, give the database any name you like. This name is used for readability only.
- Select DBMS for the **Data Source Type**.
- For the **Database Name**, type in the actual Database name as it is defined in the database manager. The database needs to have been previously defined. In this example, Sales is defined in SQLExpress.
- Zoom from the column marked "DBMS". The list that pops up will show the Magic Databases that were found in the Magic.Ini, as described in the previous section. In our example, we are using Microsoft SQLServer. Now, press **Alt+Enter** to access the Databases's properties.

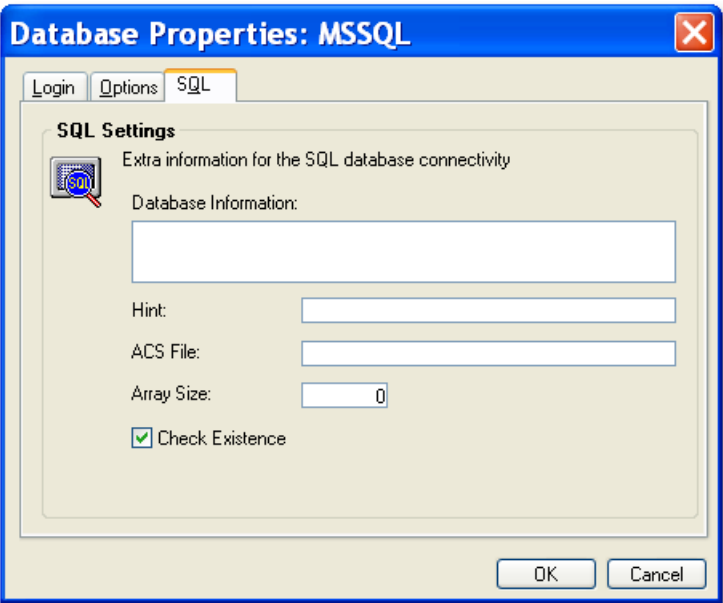




5. In the Login tab, set up the Database Server name. User Name and User Password need to be entered if the database requires them. In our example we are using Windows authentication, so no user name or password is required.

6. In the Options tab, select Change Tables in Studio if you want to maintain the table in eDeveloper. If the tables are created by another application, you will not want a programmer to change it accidentally, so you would un-check this box.





7. In the SQL tab, you can add SQL commands that will be used to connect to the tables. If you want eDeveloper to create the table, check the **Check Existence** box.

C. Checking the Connection

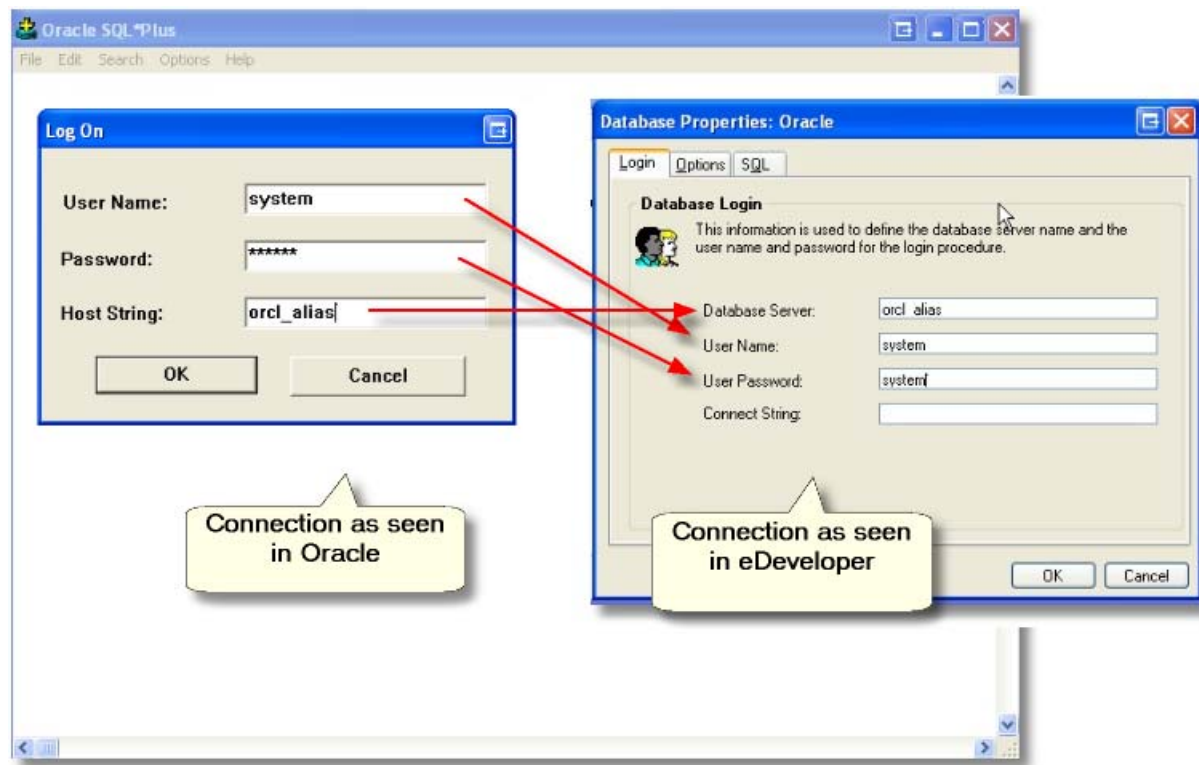
You can make sure the Database was set up correctly by using the Get Definition option in the Data Repository. You can see how to do this in Chapter 18, “How do I Access an Existing Database Table?” on page 464.

DBMS Differences

Oracle

Databases					
#	Name	Data Source Type	Database Name	DBMS	Location
1	Btrieve	DBMS		Btrieve	
2	db2	DBMS	sample	DB2	
3	Default Database	DBMS		Btrieve	
4	Memory	DBMS		Memory	
5	MySQL	DBMS	mysql	MySQL	
6	Oracle	DBMS		ORACLE	
7	SQLServer	DBMS	pubs	MicrosoftSQLServer	
8	Oracle	DBMS		ORACLE	

To define an Oracle Database, set the **DBMS** column to Oracle. The Oracle alias already points to the database, so you do not need to specify anything in the **Database Name** column.



In the Database properties, the Database Server should get the Host String, while the User name and password are the same.



## DB2

```
db2 -> connect to sample user db2admin using db2admin

Database Connection Information

Database server      = DB2/NT 9.1.0
SQL authorization ID = DB2ADMIN
Local database alias = SAMPLE

db2 ->
```

#	Name	Data Source Type	Database Name	DBMS	Location
1	Btrieve	DBMS		Btrieve	
2	db2F	DBMS	sample	DB2	
3	Default Database	DBMS		Btrieve	
4	Memory	DBMS		Memory	
5	MySQL	DBMS	mysql	MySQL	
6	Oracle	DBMS		ORACLE	
7	OracleF	DBMS		ORACLE	
8	SQLServer	DBMS	pubs	MicrosoftSQLServer	
9	SQLServerF	DBMS	ibolt	MicrosoftSQLServer	
10	DB2	DBMS	sample	DB2	

**Database Properties: DB2ForHeidi**

Login Options SQL

**Database Login**

This information is used to define the database server name and the user name and password for the login procedure.

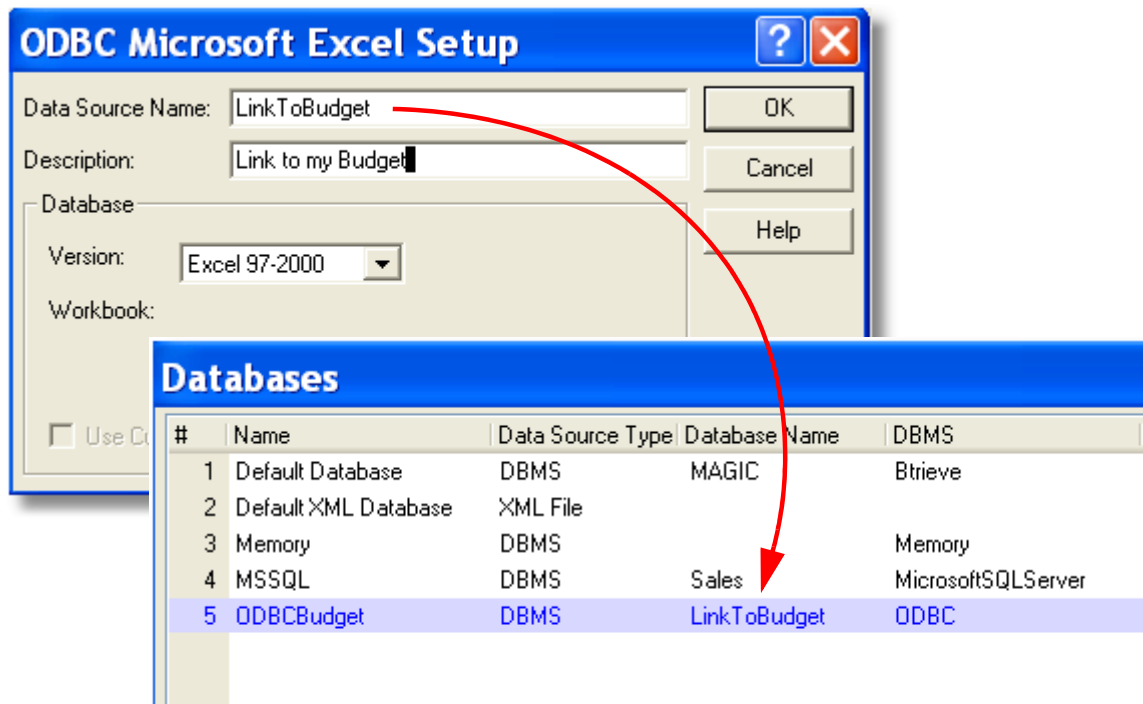
Database Server:

User Name:

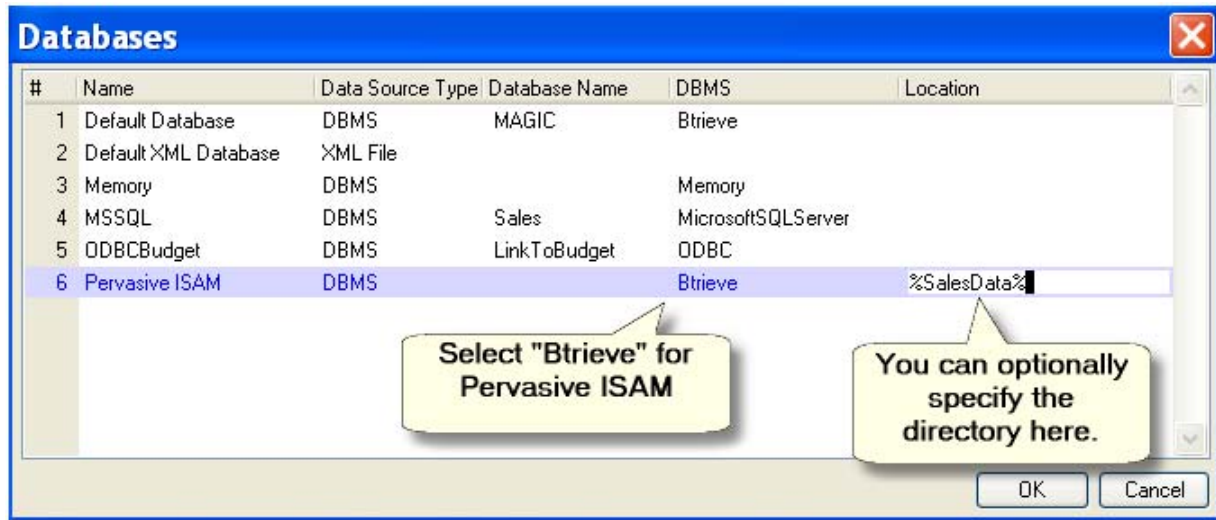
User Password:

Connect String:

To define an DB2 Database, set the **DBMS** column to DB2. The DB2 alias should be typed in the **Database Name** column.

*ODBC*

For an ODBC database, set the Database name in eDeveloper to the Data Source Name in ODBC. Also, for some databases, such as ODBC\_MSSQQL and ODBC\_MySQL, you will also need to set the user name and password.

*Pervasive ISAM*

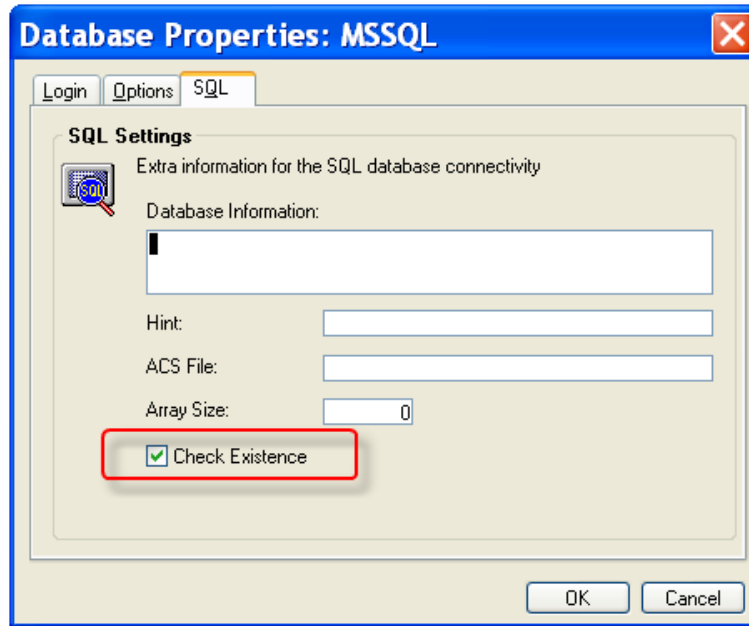
Database

For a Pervasive ISAM database, you don't need to set up a database within Pervasive. The ISAM files are created like any other operating system file.

Set the **DBMS** name to Btrieve. You can specify a path in the **Location** column if you want. You can also specify the location in Data Repository, in the **Data source name** column.

## How do I Create a Database Table From eDeveloper?

When you create a database table in eDeveloper, as explained in Chapter 18, “How do I Create a Database Table Using eDeveloper?” on page 457, eDeveloper can create the table in the DBMS automatically, and reconfigure the table if it changes.



If you want eDeveloper to do this, you need to make sure **Check Existence** is set to Yes. To do this:

1. Close your current application.
2. Select **Options->Settings->Databases**.
3. Select the database you want to check.
4. Press **Alt+Enter** to access the Database properties.
5. Select the SQL tab.
6. Check **Check Existence**.

Now, when you create a new table in Data sources, or change an existing Data source, the database table will be changed in the DBMS.

---

## How do I Access an Existing Database Table or View?

If a Database table already exists in the DBMS, you can bring the definition into eDeveloper, to automatically create the Data source. This is explained in Chapter 18, “How do I Access an Existing Database Table?” on page 464.

## How do I View SQL Statements Sent by eDeveloper to the Database?

Although eDeveloper will format SQL statements for you behind the scenes, you can view the actual SQL call in a couple of ways:

- You can turn on debugging in eDeveloper to watch each SQL call.
- You can view the SQL using logging tools within the DBMS you are using.

We won't cover the second one here, since it depends on the DBMS. Viewing the SQL statement in eDeveloper is very easy, as you can see below.

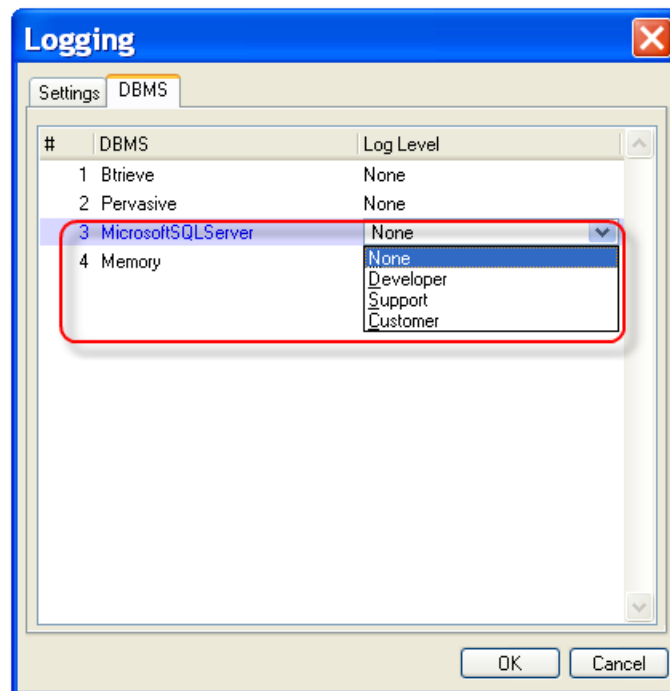
### Logging SQL Statements in eDeveloper

1. First, you need to turn on logging. You do this by going to **Options->Settings->Logging**.

Set the logging level you want for this particular DBMS. None is no logging; Customer has some, Developer has more than that, and Support has the most.

- **None** - No log file will be generated
- **Customer** - Log only the SQL commands generated
- **Support** - Additional information for the developer
- **Developer** - A full log to be generated for use by the MSE Technical Support department

**Customer** logging is sufficient to view the SQL statement sent by eDeveloper, but



you can get more information using Developer or Support. This example is an activity log with logging set to **Support**.

```

Read Transaction 0
,57843 ms7_crshr_open(): >>>> ctxID = -1.000000, dbd_hdl = 0
,57843 ms7_crshr_open(): db = pubs, table = Suppliers, crshr_hdl = 0, dir = A, dir_rev
,57843 ms7_crshr_open(): checking if dummy fetch available
,57843 ms7_connect_extra_session(): >>>> database = pubs, pConnection->sess_extra
,57843 SET IMPLICIT TRANSACTIONS ON
,57843 MS7LogNumberFld(): db_crshr->buf = :000000000000000000000000038527588:
,57843 MS7LogNumberFld(): db_crshr->buf = :000000000000000000000000038527668:
,57843 STMT: SELECT Supplier_Code,Supplier_Name FROM pubs.dbo.Suppliers WHERE Sup
,57843 STMT: SELECT Supplier_Code,Supplier_Name FROM pubs.dbo.Suppliers WHERE Sup
,57843 Using CURSOR
,57843 STMT EXECUTE: SELECT Supplier_Code,Supplier_Name FROM pubs.dbo.Suppliers WH
,57843 ms7_crshr_open(): <<<<< ctxID = -1.000000, retcode = 0
Load records

```

2. Turn on Debug mode if needed, by clicking on Debug->Debug Mode.
3. Open up the **Activity Monitor** (View->Activity Monitor)

Now, when you run your program in the Debugger, you will see the SQL statements in the Activity monitor.

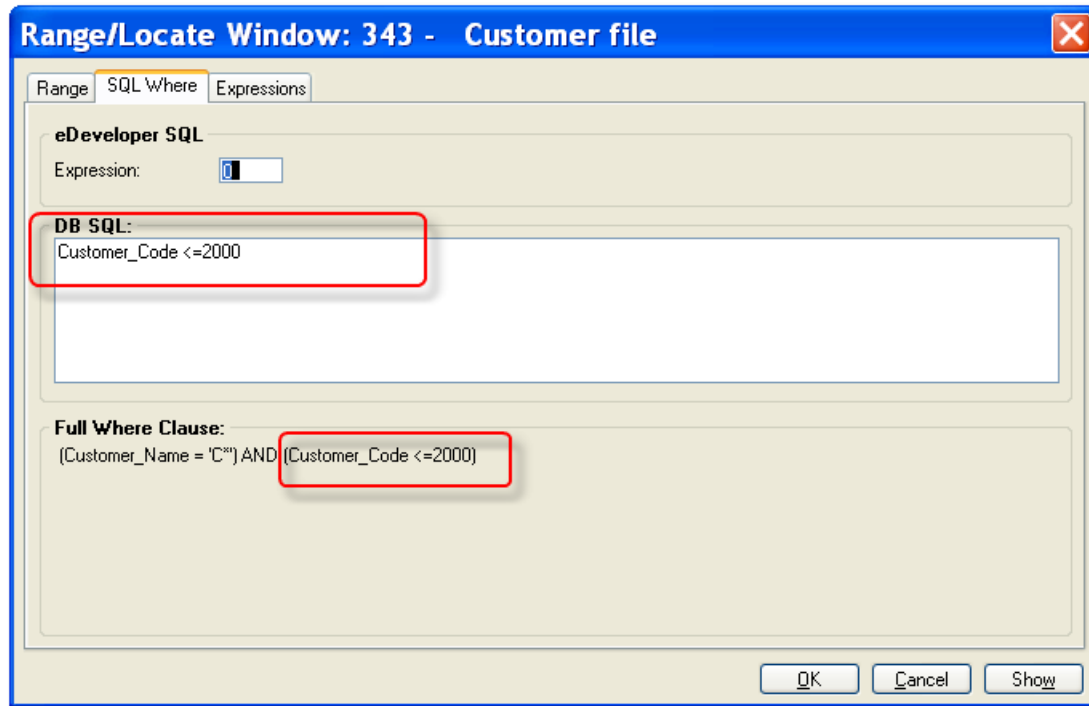
**See also:** Chapter 29, “How do I Debug my Application Using the Debugger?” on page 711.

## How do I Send My Own SQL Statements to the Database?

When you use eDeveloper with an SQL database, eDeveloper generates the SQL code needed. However, there are times when you may want to override the default code generation. For instance, you may want to call a stored procedure or tightly control how a group of records is retrieved.

This is easily done from within an eDeveloper program. There are several different ways you can send SQL statements, each of which is covered below.

## Manually entering a WHERE Clause



If you only want to send a WHERE clause, you can do this in the **Range/Locate** dialog. You can type in your Where statement in the **DB SQL** field, and you will see the result concatenated onto eDeveloper's Where clause in the **Full Where Clause** field.

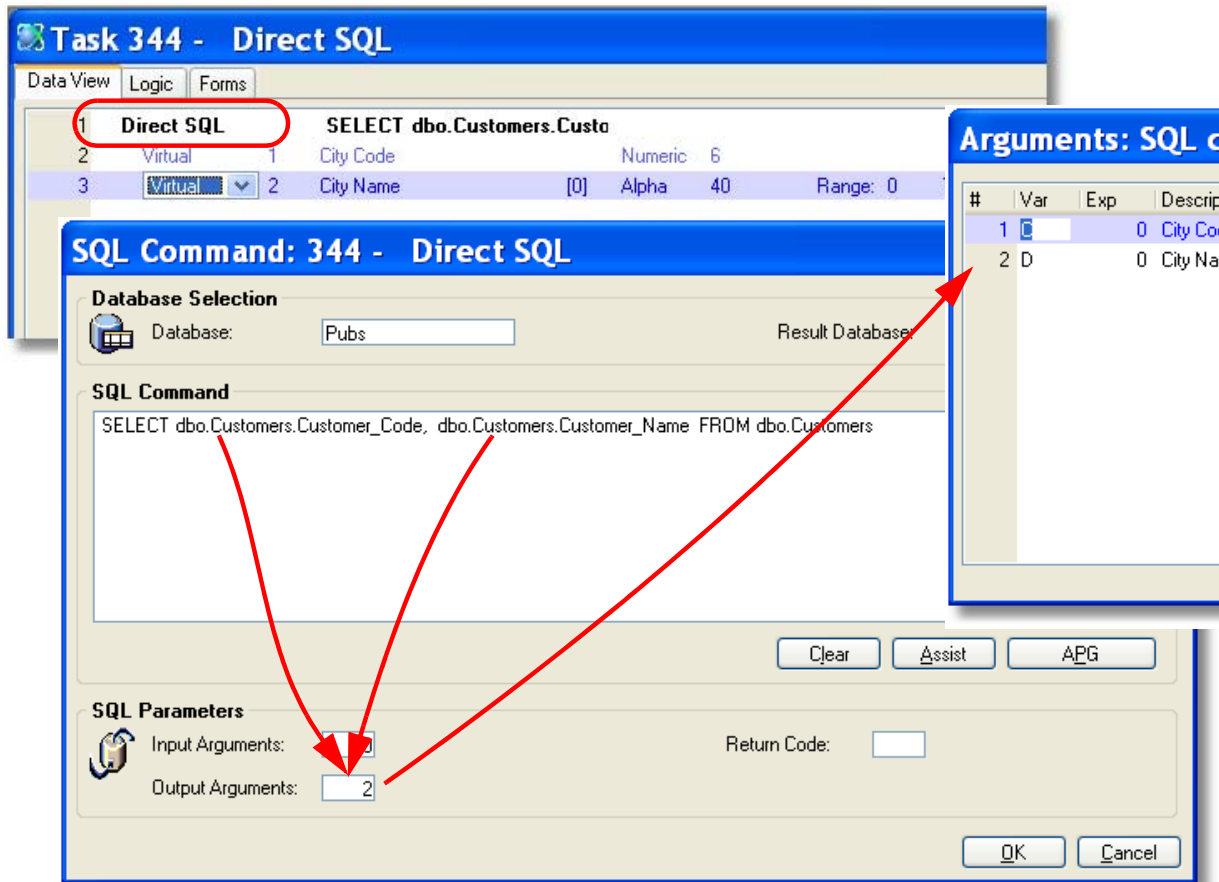
1. Go to **Task->Range/Locate**.
2. Click on the **SQL Where** tab header.
3. Enter the Where clause you want, either by typing it in the **DB SQL** field, or by zooming on the Exp field and entering an Expression.
4. The full where clause will be displayed below.

**Note:** This option is only valid when you are using Physical transactions (**Task Properties->Data->Transaction Mode = Physical**).



## Manually entering another SQL Statement

If you want to enter something more complex than the WHERE statement, you can do this using a Direct SQL statement.



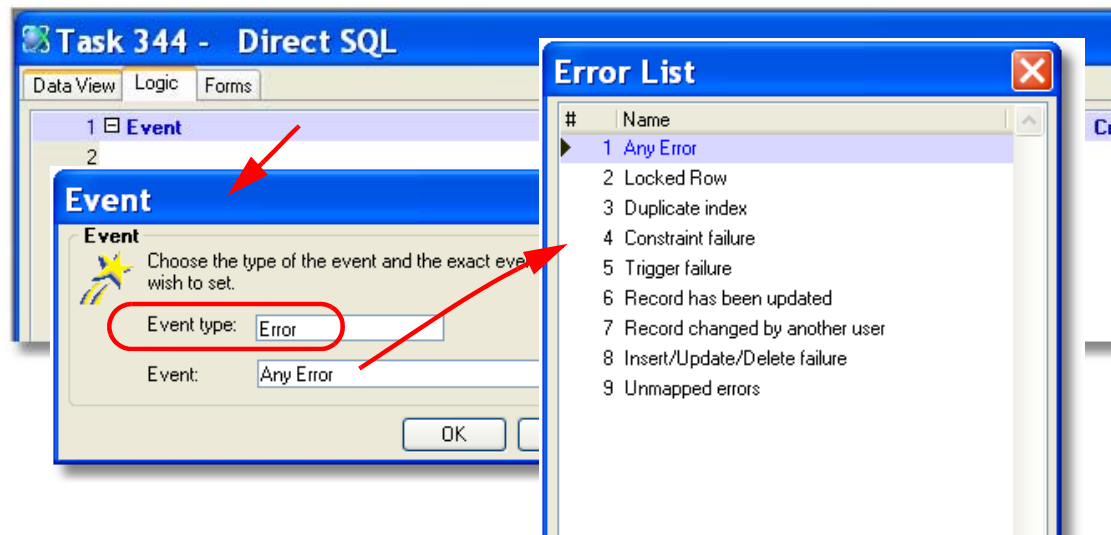
1. Go to the **Dataview** tab.
2. From line 1, select **Direct SQL**, then tab.
3. The SQL Command dialog will appear. Here you can type in an SQL command. Clicking on the **Assist** button will bring up a dialog that will help you format the SQL command.
4. Zoom from the **Output Arguments** field to map the fields from the SQL command onto virtuals in the task.

The Virtuals can be used in your program just as you would use a Real from a Main data source.

## How do I Handle a Database Error or Exception?

When a program causes the underlying database to generate an error, you can choose to handle that error within your program rather than using the default eDeveloper error handling. This is done by using a particular type of Event handler, where the Event Type is “Error”.

### Creating an Error handler



1. Go to the **Logic** tab.
2. Press **Ctrl+H** to create a header line.
3. Type **E** to select Event. The cursor will jump to the next field.
4. Zoom to bring up the **Event** dialog.
5. Select the **Error** Event type.
6. Zoom from the **Event** field to select the error you want to trap.
7. Set the **Directive** property to control how you want the eDeveloper engine to respond to the error. The engine directive tells the engine what to do after the error handler has been executed. Choosing “As strategy” will tell the engine to follow the error behavior strategy of the task (**Ctrl+P, Data tab**). Other options such as “Ignore” “Rollback & Restart” give an explicit instruction to the engine. The supported engine directives for each error. The different error strategies are described in the documentation.
8. Set the **Message** property to Yes if you want the actual DBMS Message to be automatically displayed. This overrides the **Display Full Messages** setting in **Settings->Environment->Preferences**.

As you can see there are several different specific error types you can handle. You can also handle the type “Any Error” which will be triggered when any database error is generated.

Now you have a handler that will respond to the selected error. You can add operations to handle the error.

If you add no operations, and **Directive=Ignore**, then the error will just be blocked. You may choose to do this in situations where you know there will be errors that don’t matter, such as when you are loading

records where some might be duplicates. The duplicates will generate an error and will not be loaded, but you will not have to bother the user with an error message.

One of the things you can do in the handler is to display an error message to the user, or to put an error message into a log. eDeveloper has a series of functions, such as **ErrDatabaseName**, **ErrDbmsCode**, **ErrTableName**, and **ErrDbmsMessage**, that can be used to display or save information about the error. However, if you set the DBMS Message property to Yes, then the message buffer will be cleared after the message is displayed to the user, and the DbErr and ErrDbmsMessage functions will return an empty string. So, if you want to use the DBMS error information in your program and also display it to the user, set the DBMS Message property to No, and display the message manually using a Verify operation.

## How do I Limit the End User's Access to and Manipulation of the Data?

When you automatically generate a program using the Program Generator, by default it allows the user to access any field and to change any record. However, you can control exactly how much access a user has to table data in eDeveloper by changing various settings in the task. Here is a list of some of the items you can change to control access.

1. **Task Properties->General->Initial Mode:** This sets the initial mode of the program. If it is set to Query, then the user will not be able to enter data into any field.
2. **Task Properties->Options:** Here you have a series of fields that you can set to Yes, No, or an Expression to be evaluated at runtime. If Modify is set to No, for instance, the user will not be able to put the task into Modify mode. If Delete is set to No, the user will not be able to delete a record.
3. **Main Data source properties or Link properties->Access = Read:** This property allows you to set the access mode to the table. If Access is Read, then the table will be opened in read mode, and data will not be saved regardless of how the rest of the task is configured. If the record is inadvertently updated because other settings allow modification, an error message will be generated but the record will not be updated.
4. Define only the fields that the task needs inside your task, that way they will not be available at all.
5. Set **Settings->Environment->System->Allow Update in Query** to No. When a task is in Query mode, the user cannot type in data. However, if Allow Update in Query is Yes, then the data can still be updated from within the program, if a field is updated as a passed argument or by an Update operation, for instance.

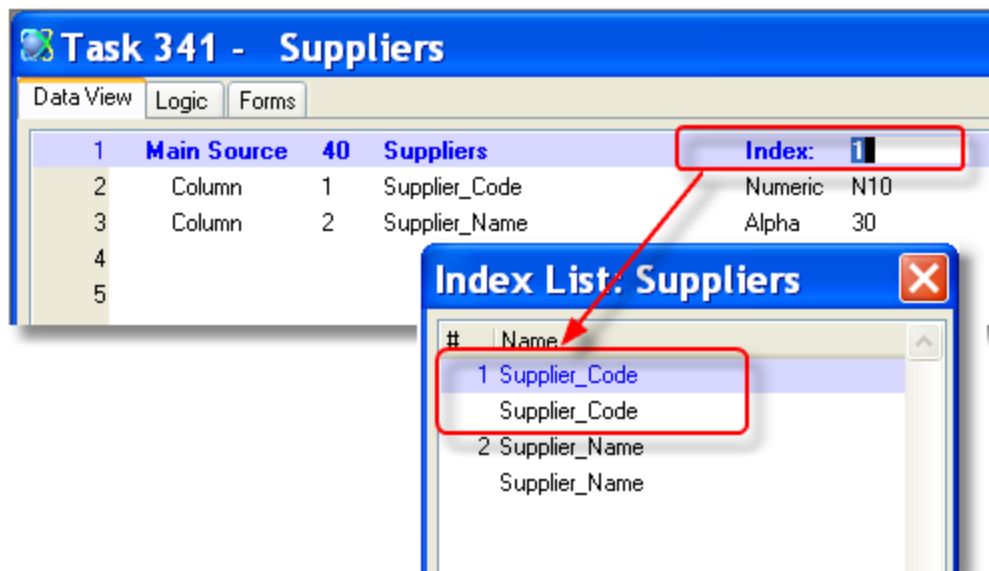
## How do I Determine the Order of the Records Retrieved from the Database?

When you display a list of records, the order in which those records are retrieved is usually very important. The record order is important from a user interface perspective, in that the user will want to see the records in a way that makes the most sense to the task. The record order is also important from a functional perspective, in that you will want the records to be retrieved in a way that is the most efficient for the filtering being done.

In eDeveloper you can have very fine control over the record ordering. There are two main ways this happens, each of which will be covered individually:

- Setting the table index
- Using the task sort

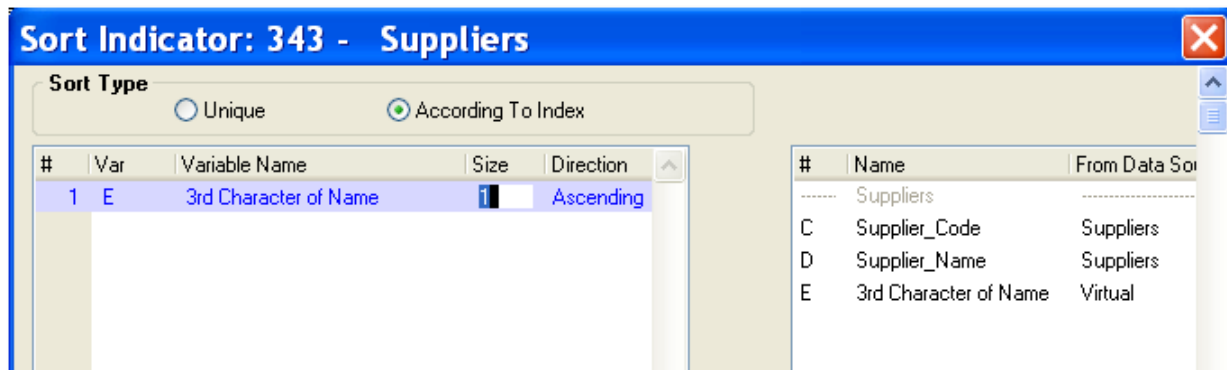
### Setting the Table Index



When you use a Data source in a task, either as a Main source or a Linked source, you will specify the index to use in the Index property. In this example, the Suppliers will be fetched by Supplier\_Code.

For the Main source, it is most efficient to use the Index that works best for the Range you are using. For a Linked source, it is most efficient to use the Index that works with the Locate used in the Link. In our example, using Supplier\_Code for the Index would be best if we are looking for, say, Suppliers where Supplier\_Code is from A001 to A003.

## Using the Task Sort



The Task Sort allows you to re-order the records after the initial record selection is done. The Task sort is extremely flexible. For instance, you can create a virtual that is initialized to some value, and use that virtual in the Sort.

To set up a Task sort:

1. Go to **Task->Sort** (**Ctrl+T**).
2. In the left hand column, press F4 to open up a line.
3. Select the variable you want to participate in the sort, from the list on the right.
4. If the field is long, you can choose to sort on only the first few characters in the Size column.
5. If you want to reverse the sort order, select Descending in the Direction columns.
6. Continue selecting variables as needed.

Now, when the task runs, the records will be sorted before the user sees them. In our example, they will be sorted by the third character of the Supplier name.

**See also:** Chapter 5, “How do I Dynamically Change the Display Order of Records in a Program?” on page 91  
 Chapter , “How do I Retrieve Records from a Database Table in a Predefined Order?” on page 475

## How do I Access a Specific SQL Type?

The screenshot shows two side-by-side windows from the eDeveloper application. The left window, titled 'Table - dbo.CustomerData', has a 'Summary' tab selected. It displays a table with columns: Column Name, Data Type, and Allow Nulls. The 'TotalSales' row is highlighted with a red box. The right window, titled 'Columns', has a 'Columns' tab selected. It displays a table with columns: #, Name, Model Attribute, and Picture. The 'TotalSales' row (numbered 2) is also highlighted with a red box.

Column Name	Data Type	Allow Nulls
Unique_ID	uniqueidentifier	<input checked="" type="checkbox"/>
TotalSales	money	<input checked="" type="checkbox"/>
NextContact	datetime	<input type="checkbox"/>
Photo	image	<input type="checkbox"/>
Status	nchar(1)	<input checked="" type="checkbox"/>
Delete_Flag	char(1)	<input type="checkbox"/>
Customer_Code	char(8)	<input type="checkbox"/>
Customer_Name	char(40)	<input type="checkbox"/>
Ship_Address_1	char(45)	<input type="checkbox"/>
Ship_Street_Address_2	char(25)	<input type="checkbox"/>
Ship_City	char(20)	<input type="checkbox"/>

#	Name	Model Attribute	Picture
1	Unique_ID	0 Alpha	38
2	TotalSales	0 Numeric	10.3
3	NextContact	0 Date	##/##/####
4	NextContact_time	0 Time	HH:MM:SS
5	Photo	0 Blob	
6	Status	0 Unicode	1
7	Delete_Flag	0 Alpha	1
8	Customer_Code	0 Alpha	8
9	Customer_Name	0 Alpha	40
10	Ship_Address_1	0 Alpha	45
11	Ship_Street_Address:	0 Alpha	25

Each column in eDeveloper represents corresponding data in the database. Every database server has its own data types, however, eDeveloper has a translation for those types inside the database gateway.

You can see a listing of the translation in the eDeveloper Help files, under

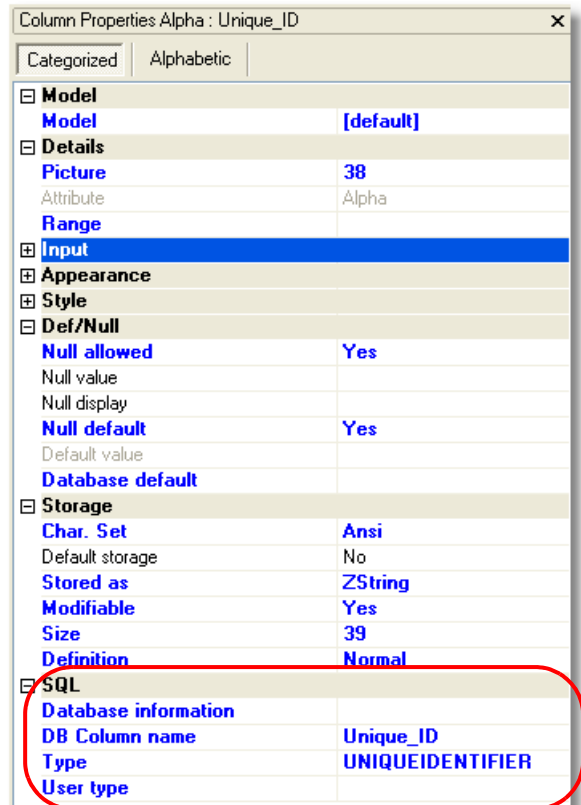
**Data Management > SQL Considerations > Configuration and Performance > Transactions**

However, you can also just look at the translation that eDeveloper does when a Get Definition is done on an existing table, or when a table designed in eDeveloper gets created in the database.

In the example above, we see two tables. On the left is the MSSQL representation of the columns in a table, CustomerData. On the right we see the table as it was brought in to eDeveloper with a **Options->Get Definition** (F9).

If you look into the properties of each column, you can see more detail about how each column is defined, including how nulls are handled. The original database-defined type is also retained in the SQL->Type property.

Some columns are handled specially. DATETIME columns are split by eDeveloper into two fields, Date and Time, so that the two parts can be handled individually while in eDeveloper, but they will be stored as a proper DATETIME column in the database.



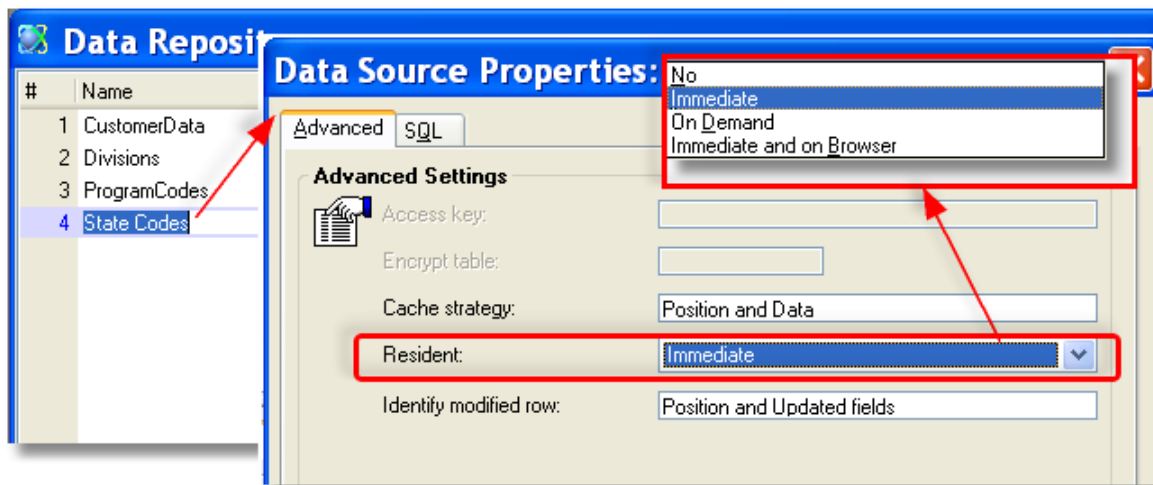


## How do I Minimize Database Access for Read-only Data?

Database access tends to slow down programs, because each request for data has to go through the database server. Some kinds of access are slower than others; data that is locked takes longer than data that is read-only.

However, if you have data that is accessed a lot, and doesn't change very often, such as lookup tables, you can load these tables just once when the application opens (or when the table is first accessed) and continue to access the same copy throughout the application.

This is accomplished by defining the table as *resident*.



To change the Resident setting:

1. Position the cursor on the table you want to change, in the Data repository.
2. Press **Alt+Enter** to access the Data source properties. The Advanced tab will be chosen by default.
3. Choose the option you want on the Resident field.

You have several choices here:

- **No** is the default setting. The table will be opened every time it is opened by a task.
- **Immediate**: The table will be opened once, when the application starts. Any tasks that use the table will get the same data that was in the table when it was first opened.
- **On Demand**: The table will be opened once, when it is first used by a task. After that, other tasks will get the same copy of the data.
- **Immediate and on Browser**: This refers only to Browser client tasks. It will cause the table to be downloaded to the client's browser.

## Updating a Resident Table

#	Name	Data source name	Database
1	CustomerData	CustomerData	MSSQL
2	Divisions	Divisions	MSSQL
3	ProgramCodes	ProgramCodes	MSSQL
4	State Codes RESIDENT	State_Codes	MSSQL
5	State Codes not resident	State_Codes	MSSQL

Once a table is defined as Resident, no program in the application can update it.

If you want to update that table within the same application, you need to make a *copy* of the table within the Data repository, and set the Resident flag to *No* for the copy. In this example, we have two copies of the State\_Codes table. One is Resident, the other isn't.

Now, we can create a program that updates table #5, and those changes will be stored in the State\_Codes table. However, this will not change the copy of the "State\_Codes" table that the other programs are using, which is the Resident copy, table #4.

Data View	Logic	Forms
1	<input checked="" type="checkbox"/> Task	Prefix
2	Evaluate	Expression
3		
		1 DbReload ('4'DSOURCE,")

So, to refresh the copy of the State\_Codes table that is being used, we use an eDeveloper function called **DBReload()**. DBReload takes on parameter, the DSource number (and, optionally, name). After the DBReload() is executed, the copy in memory will be refreshed.

## How do I Reduce Database Access?

eDeveloper handles much of the database access automatically, so you do not have to code statements to tell eDeveloper when to fetch records. However, you do have control over the factors that affect how much access is being done. Using the database access properties in eDeveloper in a way that fits your application is one of the biggest influences on the response speed of your programs.

**Hint:** *You can use the Debugger and the native SQL profiler tools to determine just how many times the database is being accessed: when tables are being opened and how. It is a good idea to be familiar with what is going on behind the scenes, to make sure your programs are efficiently using resources.*

### Read-only Tables

Some tables are used primarily for reference, and are not updated often, such as code tables. There is a special access property you can use for such tables, the Resident property. This is described in Chapter 36, “How do I Minimize Database Access for Read-only Data?” on page 877.

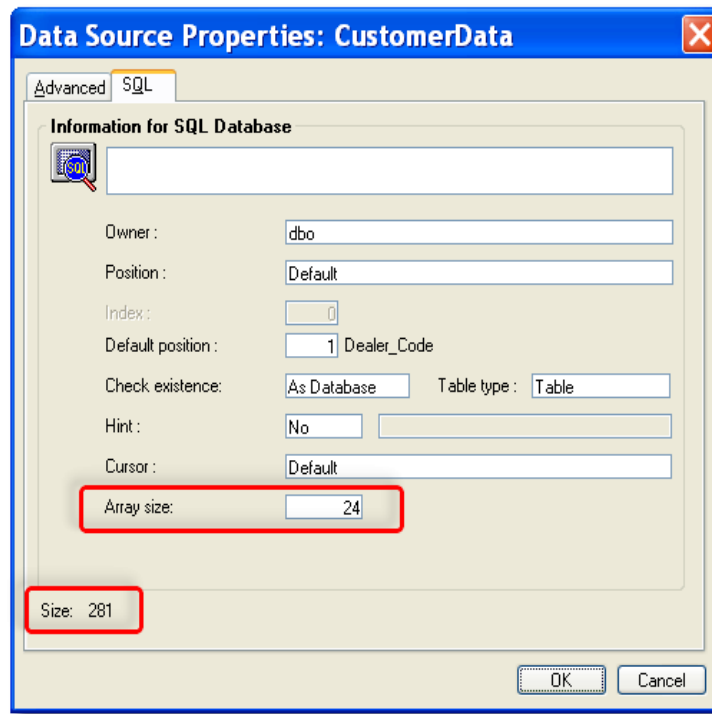
### Controlling Record Fetching

You also have a great deal of control over how many records are fetched from the database, and when. There are three main factors that affect this:

- Array Size
- Cache
- Preload View
- Use of ranges and Direct SQL

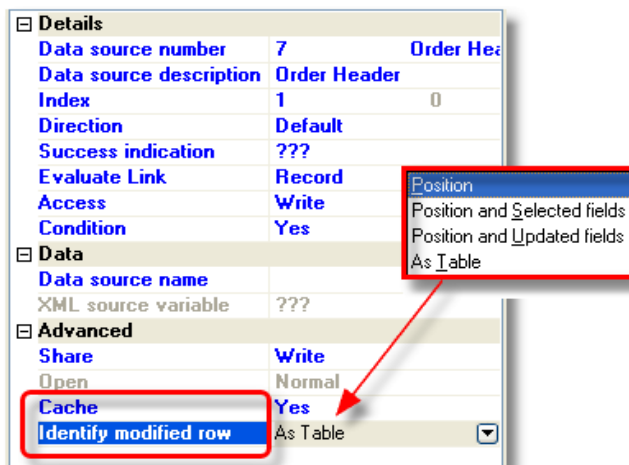
#### *Array Size*

When eDeveloper fetches records, it does not fetch them one at a time, but rather as a block of records. You can control how many records are fetched at one time by setting the **Array Size** property for the Data source.



If the Array size is set to zero, then the default setting for array size for the Database is used. If the array size on the database is also zero, then the eDeveloper default is used.

The eDeveloper default is 1200/record size. So if your record is 200 bytes long, 6 records will be fetched at a time. For SQL files, the actual record length will vary at runtime, depending on how many columns are fetched back.



## Cache

You can use the Cache property to reduce data-base access also.

Cache can be used in online tasks for all tables. For batch tasks it can be used only on linked tables.

When there is a cache on the main table (position and data) eDev will not re-read the record from the database when positioned on it. When there is cache on a linked table, eDeveloper will read it once from the database and reuse it if needed for a link from another record while in the same task. One can also keep the cache on a linked table even when

leaving the task, by defining the task as resident and opening the linked table in one of the ancestor tasks, the cache will be kept until leaving the task opening the table.

### *Preload View*

The **Task Properties->Data->Preload View** property has an effect on the number of fetches done before the window opens. If this is set to **Yes**, then the records are all fetched before the window is displayed. This allows scroll bars to be displayed accurately. However, if there is a lot of data that the user is not likely to scroll through, it will result in many more record fetches than would otherwise be necessary.

### *Use of Ranges and Direct SQL*

You should also try to get the database server to do as much your filtering as you can, to reduce the number of records fetched.

For instance, if you want to print all the customer records where the *Amount Overdue* is greater than \$12,000, you could do this by using a **Condition** on the **Form Output** operation. However, if you do this, the engine has to fetch every single customer record back, to test the condition. This could require hundreds of fetches. On the other hand, if you use a **Range** to filter the data, eDeveloper will format one SQL statement, and the server will only return the records where *Amount Overdue* is greater than \$12,000, which hopefully will only be one fetch, depending on how many customers owe you money.

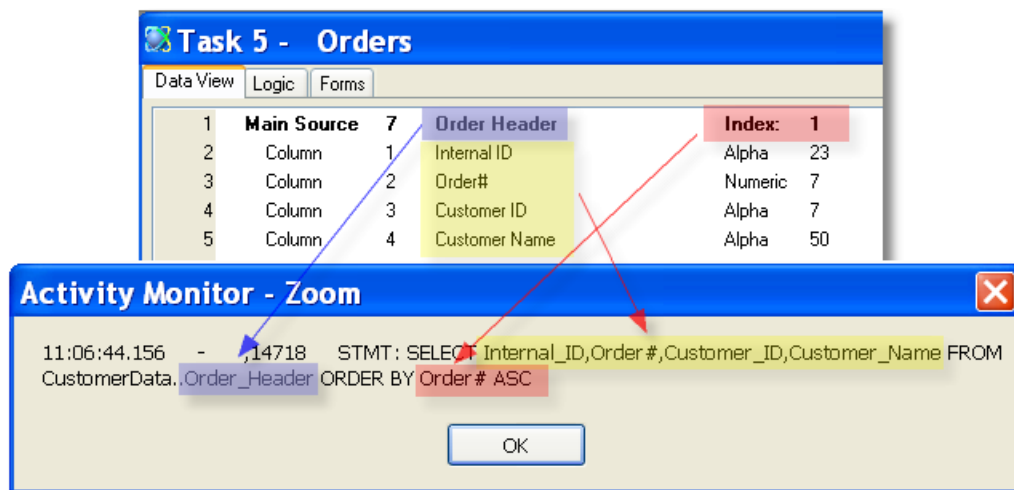
There are also instances where a Direct SQL statement can be the most efficient way to work with data. For more information about this, see Chapter 36, "How do I Send My Own SQL Statements to the Database?" on page 867.

## How do I Affect the Select Statement Sent to the Database?

When you use the eDeveloper studio to write a task, you will use the Studio interface to quickly write database queries. You don't need to know how the underlying SQL, ISAM, XML, or memory table query actually works, because the database gateways will handle that.

However, you can affect how an underlying SQL query is structured by how you structure your eDeveloper program.

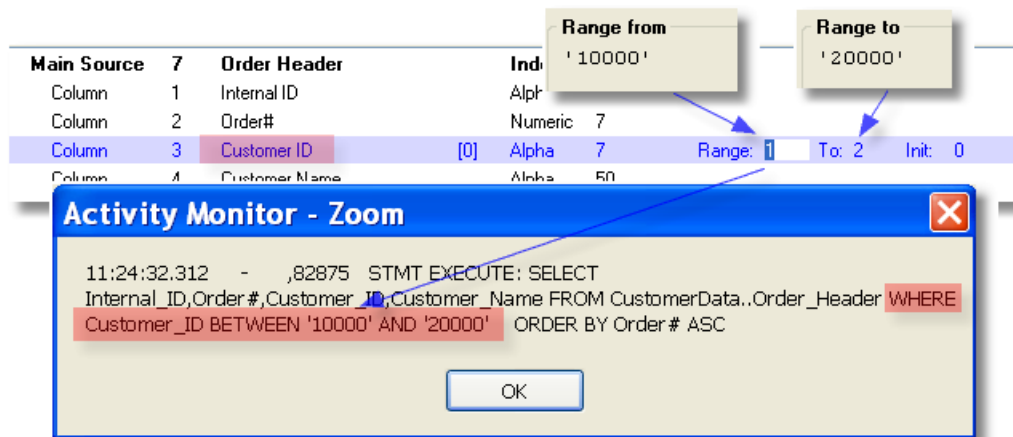
**Note:** You can view the SQL Query that is generated using the Debugger. See Chapter 36, "How do I View SQL Statements Sent by eDeveloper to the Database?" on page 866.



Here are the items that affect the SQL Query:

**Which columns you choose:** The columns you choose will be included in the Query, in the order you selected them in the Data View.

**The eDeveloper index chosen (and whether or not it is unique):** The Index will be translated into an ORDER BY clause.



**The task Ranges:** The Ranges will be included as WHERE clauses.

**A Task Sort:** Any task Sorts that you enter will be translated into an ORDER BY clause.

And, of course, you can also add **Direct SQL** statements to the Query. See Chapter 36, “How do I Send My Own SQL Statements to the Database?” on page 867.

## How Can I Determine eDeveloper Behavior When Several Users are Modifying the Same Row?

First of all, eDeveloper will behave differently regarding modified rows depending on the kind of transaction handling you are doing.

*Physical* transactions are handled by the database engine, while *Deferred* transactions are handled first by eDeveloper and then committed to the database in a separate step.

### Identifying Modified Rows using Physical Transactions

If you are using Physical Transactions, then how the row is identified depends on the underlying DBMS settings.

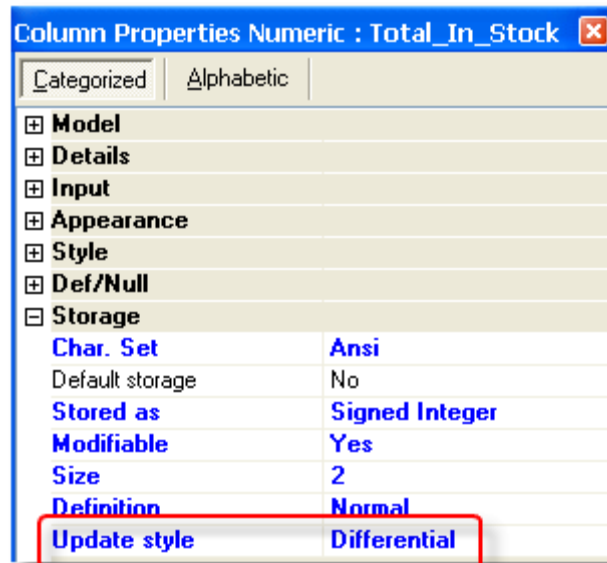
### Identifying a Modified Row, Using Deferred Transactions

First, eDeveloper has to identify that in fact, in fact, the row is being modified. The way this is handled depends on the **Identify Modified Row** property in the Data Source. There are three different settings, and they have three very different results.

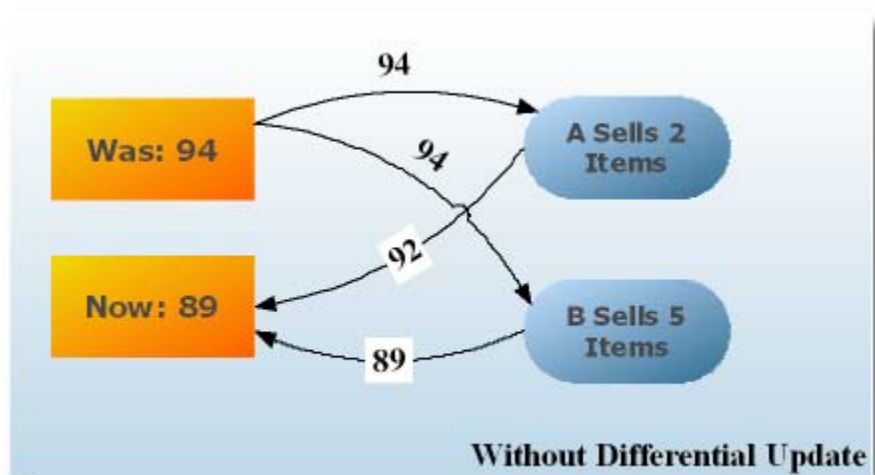
Let's take a look at what happens if two users, User A and User B, are updating the same row in an SQL table. Both fetch the same row for update. Both then update the row, but User A writes the record before User B does.

1. **Position** means that eDeveloper will only look at the unique ID of the row that is modified. So User A's record is written, then User B's record overwrites the changes. Only User B's changes are saved, and no error is raised.
2. **Position and Updated Fields** means that eDeveloper will look at the unique ID of the row, and also what field was updated. So if User A and User B actually updated *different* fields, then both sets of changes would be saved and there would be no error. But if they both updated the *same* field, then an error would be raised for User B and his changes would not be saved.
3. **Position and Selected Fields** means that eDeveloper will look at the Unique ID, and also all the fields that were selected in the tasks involved, whether or not those fields were updated. So if User A and User B had the same fields selected in their tasks, User B would get an error and his changes wouldn't be saved.
4. **As Table** means that the **Identify modified row** property will be inherited from the Data source definition.



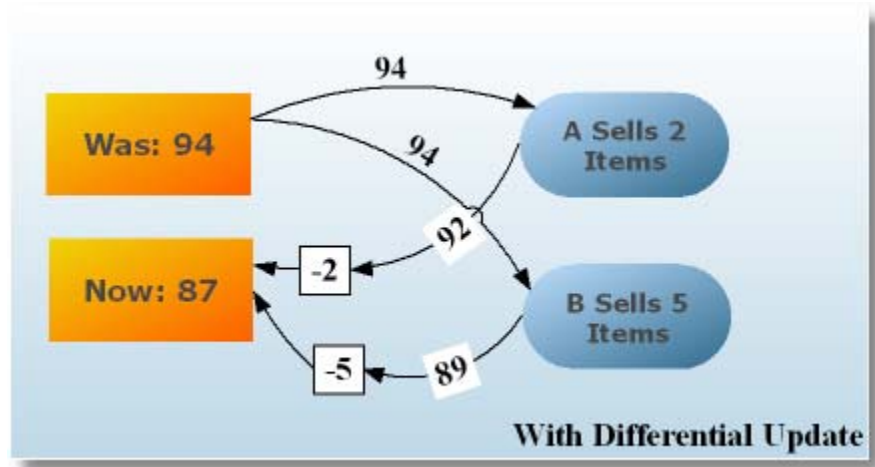
*The Effect of Update Style*

You can also use the **Update Style** Property to minimize the problems of users updating the same numeric data.



Suppose you have two shopping cart programs running. Program A reads that there are 94 items in stock: so does Program B. Then Program A sells 2 items, and writes 92 to the field. Program B sells 5 items, and writes 89 to the field. The field is now 89, which is incorrect.

However, if the column was defined in the Data Source Repository with **Update Style->Differential**, then it will be updated differently.



Here you can see that the programs work just as they did before. But instead of moving 92 or 89 into the field, the field is decremented by the difference between the value read and the value to write. So instead of writing "92", Program A causes the field to be decremented by 2. And program B, instead of writing "89", subtracts 5. There was no change to the coding of Program A or Program B; the change was to the Data source column property.

### ***Difference between Differential Update and Incremental Update***

Note that **Differential Update** is quite different from **Incremental Update**. Whereas Differential Update is a property of a Data source column, Incremental Update is a property of the Update Operation. When Incremental Update is used in a task, the actual update operation is coded differently. Normally you would have to code explicit logic to handle adding, modifying, or deleting totals when working with sub-records, similar to that shown below.

Logic		Forms	
Event	Add Item		Scope SubTree
Update	Variable	H	Total Cost
Update	Variable	F	Total Items
Update	Variable	G	Total SubRecords
With:	2	Total Cost+Item_Cost	
With:	3	Total Items+Item_Qty	
With:	4	Total SubRecords+1	
Event	Delete Item		Scope SubTree
Update	Variable	H	Total Cost
Update	Variable	F	Total Items
Update	Variable	G	Total SubRecords
With:	5	Total Cost-Item_Cost	
With:	6	Total Items-Item_Qty	
With:	7	Total SubRecords-1	
Event	Modify Item		Scope SubTree
Update	Variable	H	Total Cost
Update	Variable	F	Total Items
With:	8	Total Cost-VarPrev['I'	
With:	9	Total Items-VarPrev['J'	

However, with Incremental update, this is all handled automatically in Record Suffix. The programmer just indicates the amount that needs to be added or deleted:

**Task 11 - Incremental Update 1**

Record	Suffix	Variable	With	Item_Cost	Item_Qty
1	Update	H	2	Item_Cost	
2	Update	F	3		Item_Qty
3	Update	G	4		1
4					
5					
6					

**Properties of : Update Ope...**

Category	Value
Variable	H
With	2
Incremental	Yes
Force update	No
Condition	Yes 0

In this example, when a record is created, *Item\_Qty* will be added to *Total Items*. When a record is deleted, *Item\_Qty* will be deleted from *Total Items*. But if *Item\_Qty* is changed, *Total Items* will be changed by the amount that *Item\_Qty* was changed. Incremental update is a sort of “smart total” for summing child records.

## How do I Implement a One-to-Many Relationship in eDeveloper when there is a Database Constraint?

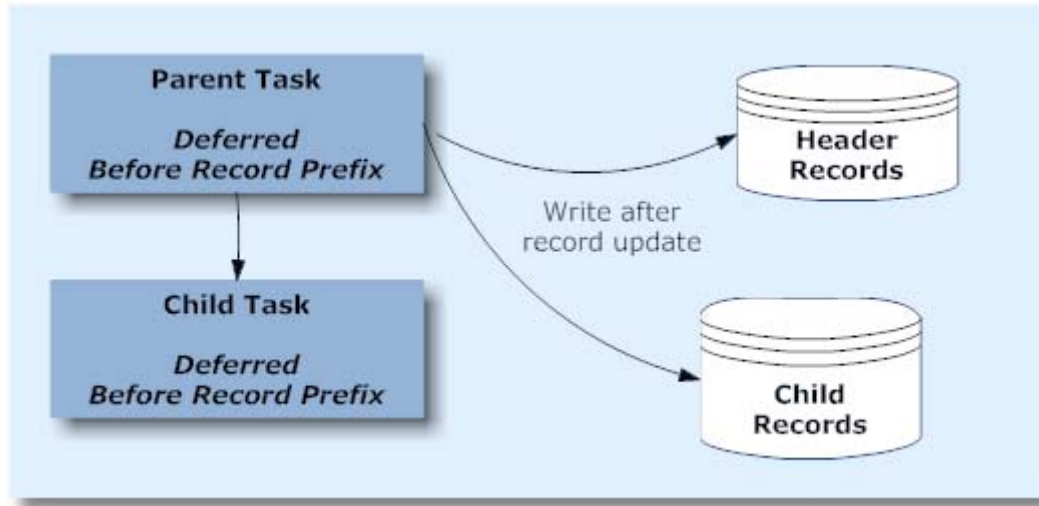
Often you will have a form where the parent task displays and updates one record, while a subtask displays and updates child records. In this case, the parent record might not actually be written to the database table when the child task takes control.

To avoid this, you want to commit the parent record before the child task takes control. You can do this easily in eDeveloper by doing the following:

- Set the Parent transaction mode to **Deferred** or **Within Active** (where the task that opens the transaction is Deferred).
- Set the Child task transaction mode to either **Deferred** or **Within Active**.
- On the Call operation, set the Sync Data property to **Yes**.

If you are using a Subform rather than a Call operation, you don't have Sync Data property available to set. However, if the parent and child tasks are set as explained above, the Subform task is automatically called as if the Sync Data property were set to **Yes**.

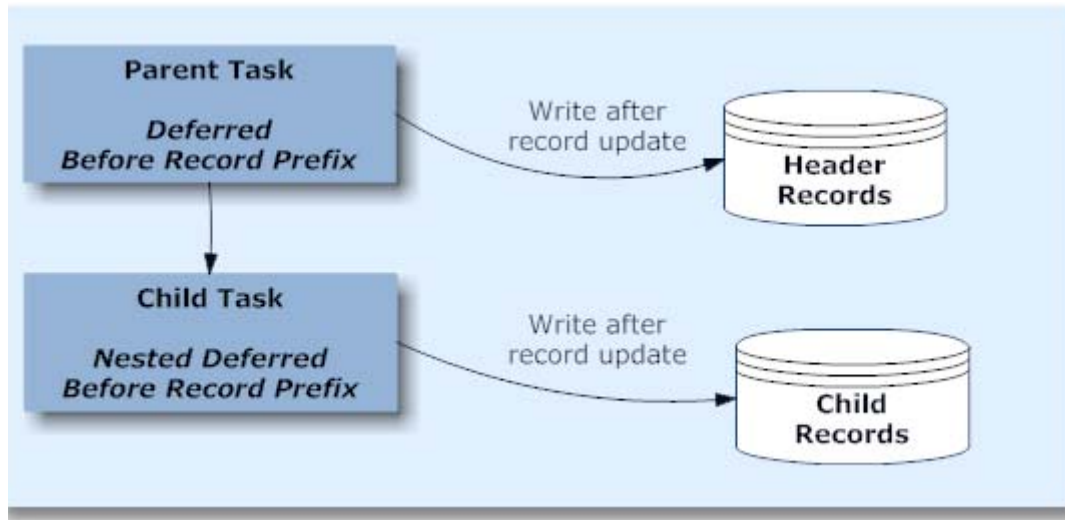
## How do I Implement a Nested Transaction?



When you are using Deferred transactions, you might have a situation where one task has a transaction open, then calls another task which also opens a transaction. For instance, suppose you have a parent task that opens a transaction in Deferred, Before Record Prefix, and calls a child task which does the same.

However, all the actual commits in this case will happen at the same time, in the parent task. If you do a rollback at the child level, the child changes will be rolled back, and so will the changes done in the parent. The parent and child tasks share in the same transaction.

If you want the child task's changes to be independent of those made in the parent, you will need to use a different kind of transaction, called **Nested Deferred** transactions, as shown below.



Here, eDeveloper caches the updates that were made by both the parent and the child task. The child task's transaction gets committed before the child task exits. If a rollback is done in the child task, only the child task's transactions are rolled back. The parent task's transactions are handled separately, in the parent task.

This doesn't implement a nested transaction in the native DBMS, but it has the same effect.

**Note:** The transaction type "Within Active" is used so the same subtask can be used by a task using either Physical or Deferred transactions, but it doesn't implement nesting. A child task set to "Within Active" would work as in the first example, as if it were set to "Deferred".

## How do I Force Writing the Current Record to the Database?

Usually, in an online task, eDeveloper writes the current record when the user leaves that record --- by moving to the next record or exiting the task. However, there are times that you will want to save the record before the user leaves the task. A common scenario is when you want to add a “print” button to the screen, to print the current record. If the print routine includes the current record, then you need to be sure that the current record is committed before the routine runs.

This is easily done by using the **Record Flush** Internal event.

Event	Print Order	Scope	SubTree
Raise Event	Record Flush	Wait:	No
Call	Program	16	Print Order
			[1 Arguments]

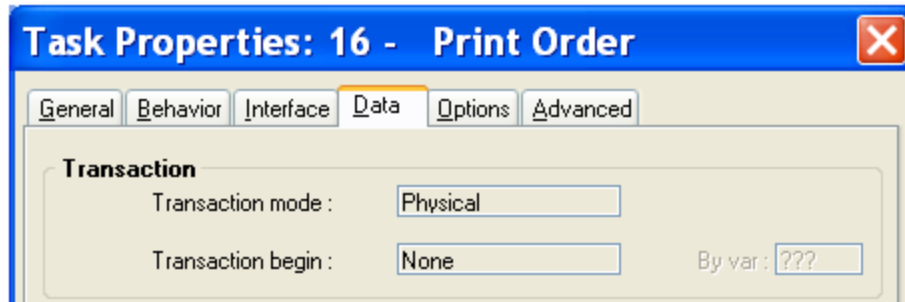
Here we have a Print Order handler, which invokes the Record Flush internal event before calling a program to print the order. This ensures that the record is written before the Print Order is called.

When using Record Flush, you need to make sure that the task is in an idle state: that is, when the task is not interacting with the user. Using **Wait=No** is a good idea also (this is the default for this operation anyway).

**Hint:** *Instead of explicitly calling Record Flush, you can also use a User Event with a Force Exit of “Post Record Update”. This forces the current record to be written before the event is executed.*

## How do I Refrain From Opening a Transaction?

By default, most tasks will open a transaction. However, if you don't want a transaction to be opened, do the following:



For on Online task:

1. Go to Task Properties (**Ctrl+P**).
2. Set Transaction mode to **Physical**.
3. Set Transaction begin to **None**.

For a Browser task:

1. Go to Task Properties (**Ctrl+P**).
2. Set Transaction mode to **None**.



## How do I Explicitly Roll Back a Transaction?

The main idea behind having transactions is that they can be rolled back ... that is, the data not committed ... when you decide that should happen. eDeveloper will automatically roll back a transaction under some circumstances, such as if a database error occurred. But you can also manually roll back a transaction based on any event or data condition, by using the ***Rollback()*** function. You can get more detailed information from the eDeveloper help, but the function basically works as follows:

**Rollback ( *Message?* , *Generation* )**

where:

***Message?***: If 'TRUE'Log, there will be a message box asking for user confirmation.

***Generation*** : 1 will roll back the current transaction.

: 2 will roll back the parent transaction

: 0 will roll back the entire transaction to the beginning (however many levels there are).

## How do I Affect the Database Optimizer Behavior?

Normally when working with an SQL database, the database optimizer decides how to parse the statement and what index to use. Under most circumstances, the optimizer makes good decisions and the search is as efficient as possible.

However, built into the database optimizer is the ability to override the optimizer's behavior. One does this by sending *hints* in the SQL statement. The exact coding of the hint will vary depending on what SQL engine you are using.

eDeveloper provides three different levels where you can code SQL hints.

- **Database:** You can code the hint at the database level, and the hint will be included in every Select statement for that database.
- **Table:** You can code the hint at the table level, and the hint will be included in every Select statement that is sent for that table.
- **Index:** You can code the hint at the index level, and the hint will be sent for every Select statement that is sent for that table and index.

At each of these levels, you will find the Hint property on the Property (Alt+Enter) SQL tab .

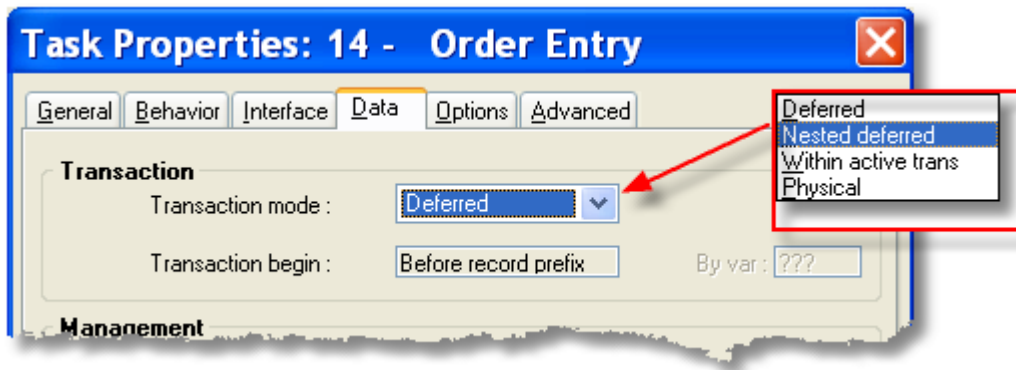
In addition, if you are using Direct SQL in a task, you can code a hint there.

## How do I Initiate a Database Transaction?

eDeveloper has built-in transaction processing, which is initiated automatically by default. Using it is mainly a matter of understanding how it works so you can use it most effectively.

Here we will look at the basics of how to set up a database transaction in eDeveloper. There are more details included in the F1 Help and elsewhere in this chapter. Also, if you are unclear about how transaction processing is working in your program, check the eDeveloper log files, or the native database log files, to see exactly how transactions are being handled.

### How to define the transaction type



Within each task, the transaction mode is set in the Task Properties (Ctrl+P) on the Data tab. For on online or batch task, you will have the four choices shown above. For Browser tasks, you have the choice of “None” instead of “Physical”.

However, there are basically three different basic types of transactions.

1. **Physical transactions** rely on the underlying DBMS to do all the transaction handling.
2. **Deferred transactions** are handled by eDeveloper. eDeveloper caches the data manipulation statements, and does the rollback if needed. Once the data is committed, eDeveloper sends the data manipulation statements to the server in one batch.
3. **No transactions:** Or, you can opt for **no transactions** at all. (See Chapter 36, “How do I Refrain From Opening a Transaction?” on page 892 for how to do this).

The other two transaction types are extensions of Physical or Deferred.

- **Nested deferred** is a special type of Deferred transaction, which is explained in Chapter 36, “How do I Implement a Nested Transaction?” on page 889.
- **Within active trans** will either initiate a Physical transaction or a Deferred transaction, whichever was used by the parent task.

So, your first decision when setting up an eDeveloper task is whether you want Physical, Deferred, or No transactions. Deferred transactions give you somewhat more flexibility, and there are some eDeveloper

features (such as nested transactions) that will only work with Deferred transactions. Also, if you are working with a Browser task, Physical transactions aren't available.

### *The transaction tree*

When deciding which kind of transaction processing to use, you need to keep in mind the tree structure of your programs. How a transaction works in a child task depends on the transaction settings of the parent task.

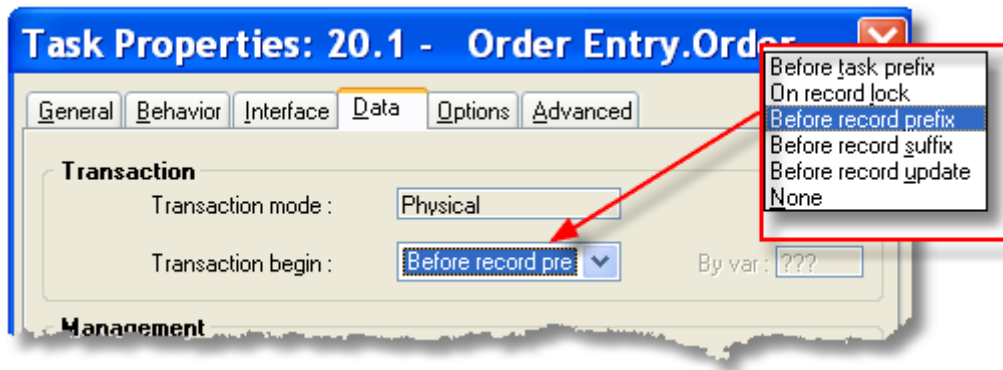
Suppose we have an Order Entry screen, showing one header record and three child records. We change the header record, then we change all three child records. While parked on the last child record, we do a rollback at the child level. Then we repeat the experiment, doing the rollback at the parent level. Here are the results for each of the settings. (All transactions are set to *Before Record Prefix*).

Parent	Child	Result	Parent Rollback	Child Rollback
<b>Deferred</b>	<b>Deferred</b>	The child changes are considered part of the parent changes. All of them are committed or rolled back as a unit.	Rolls back parent change and all child changes.	Rolls back parent change and all child changes.
<b>Deferred</b>	<i>Within Active</i>	<i>Same as Deferred-Deferred</i>		
<b>Deferred</b>	<b>Nested Deferred</b>	Each child change and each parent change is considered independent. Each record is committed when the user leaves that record.	Rolls back the parent changes (which are uncommitted since we are still on that record).	Rolls back only the last (uncommitted) child change.
<b>Deferred</b>	Physical	Each child change and each parent change is considered independent. Each record is committed when the user leaves that record.	Rolls back the parent changes (which are uncommitted since we are still on that record).	Rolls back only the last (uncommitted) child change.
Physical	Physical	The child changes are considered part of the parent changes. All of them are committed or rolled back as a unit.	Rolls back parent change and all child changes.	Rolls back parent change and all child changes.
Physical	<i>Within Active</i>	<i>Same as Physical-Physical</i>		
Physical	<b>Deferred</b>	<b>ERROR</b>		

What happens is that usually, if a transaction is opened in the parent task, the child task shares in the same transaction rather than opening it's own. The exceptions to this are if a Deferred transaction parent opens a Physical transaction child, or if the child is Nested deferred.

Also, if Physical transaction parent calls a child task set to use Deferred, this will generate an error. If you are not sure of the transaction mode of the parent program (as when a task might be called from several different programs), the safest bet is to set the Child task to use Within active. Then it will work with parents that use either Physical or Deferred transactions.

## How to define where a transaction starts and ends



The **Transaction begin** property determines when a transaction will be opened. For Physical transactions you have the six choices shown above; for Deferred transactions the only choices are Before task prefix and Before record prefix.

The transaction will end at the same level at which it began. If the transaction began at the task level, then it will end when the task ends. If the transaction began at the record level, then it will end when the user is done with that record.

However, as shown in the previous section, the current task may be sharing in the parent's transaction, in which case the transaction is open for the life of the child task regardless of the Transaction begin setting.

## How to define the tables involved

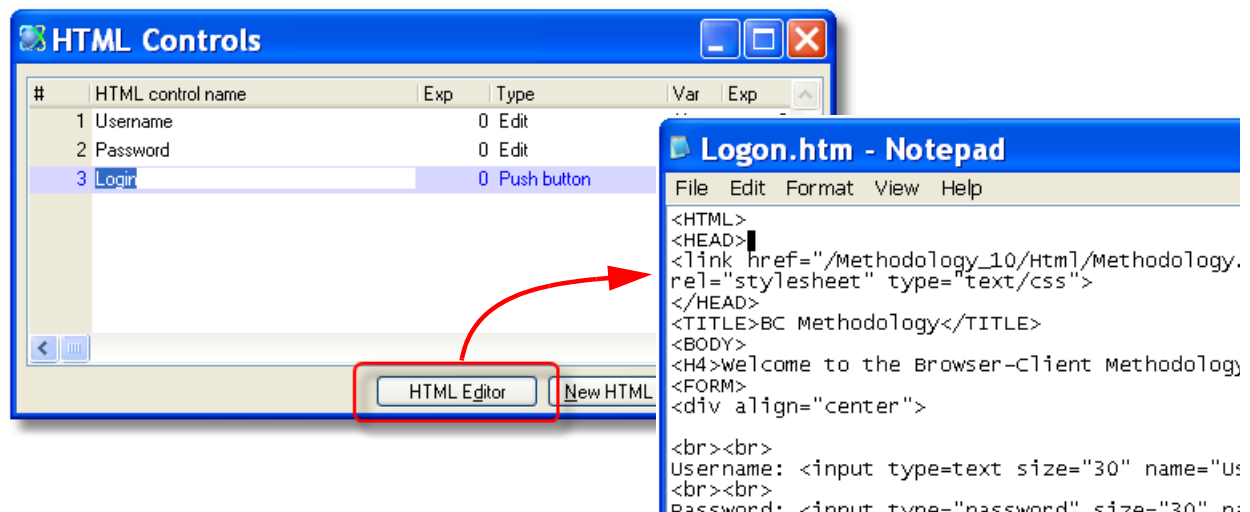
Data View						
Logic						
Forms						
1	Main Source	9	Order Header	Index:	1	
2	Column	1	Internal ID	[26]	Alpha	30
3	Column	2	Order_Num	[25]	Alpha	10
4	Column	3	Order Date	[3]	Date	##/##/####
5						
6	Declare	10	Order Lines			
7						
8	Column	4	Order Placed by	[10]	Alpha	12
9	Column	5	Customer ID	[23]	Alpha	7

When you are setting up the parent task of the task tree, you should make sure all the tables that will participate in the transaction are declared in the parent task. Declaring the table in the parent will open the table at the highest level, which in addition to making the transactions work properly, will also make the data access faster.



## Chapter 37: Browser

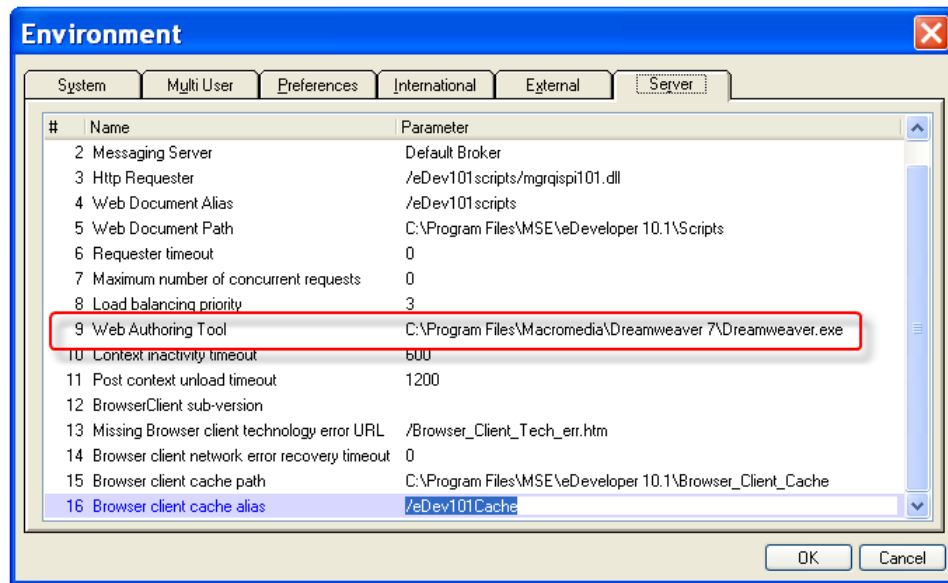
### How do I Set My Preferred HTML Editor?



From the Form editor, you can edit the HTML directly, either by zooming on the form from a Merge program, or by clicking on the HTML Editor in a browser-client program. By default, Notepad is the editor that appears.

However, you can choose which editor you want to use. Using an editor designed specifically for HTML, such as Front Page or Dreamweaver, can be useful.

## Setting the Preferred HTML Editor



1. Go to **Options->Settings->Environment**
2. Click on the **Server** tab.
3. Go to the **Web Authoring tool** line.
4. Enter the name of the authoring tool you would like to use. You can zoom to select the tool from a file list.

Now when you zoom to edit the form, your chosen editor will open.



## How do I Implement JavaScript Functions Within the Application?

```
</HEAD>
<script language="javascript">
function resizeFrame()
{
    document.all.BottomRightFrame.style.top="0px";
    document.all.BottomRightFrame.style.left="0px";
    document.all.BottomRightFrame.style.height="100%";
    document.all.BottomRightFrame.style.width="100%";
}
</script>
<body scroll=no>
<input type="text" size="1" name="Parking" style="position:absolute;left:-100
<iframe name="BottomRightFrame" src="about:blank" frameborder="0" frames
```

Task 17 - Layering Dispatcher				
Data View Logic Forms				
1	Task	Prefix		
2	Raise Event	Resize Frame		
3	Raise Event	Call "Layering1"		
4				
5	Event	Resize Frame		
6	Evaluate	Expression	3	CallJS ('resizeFrame()')
7				

When you are writing a browser-client program, sometimes you may want to call a JavaScript that is embedded in your HTML page. This is easily done using the **CallJS()** function. You can use this function with an Evaluate operation at any point in your program, and the JavaScript will be executed at that point.

The JavaScript to be executed needs to be located on the HTML page you are working with, either directly in the HTML or in a linked JavaScript file.

## How do I Implement a One-to-Many Relationship?

## Simple 1:N

New Order			
<b>Customer:</b>	1	Barry	
<b>Order Date:</b>	<input type="text" value="12/12/2008"/>		
<b>Ship Date:</b>	<input type="text" value="12/12/2008"/>		
<b>Status</b>	<input type="text" value="Open"/>		
<b>Total:</b>	333.00	66.81 (US\$)	

Row#	Product	Price	Qty	Total
1	<input type="text" value="Select"/> <input type="text" value="1"/> Cadbury - Nuggets	45.00	<input type="text" value="2"/>	90.00
2	<input type="text" value="Select"/> <input type="text" value="3"/> Cote dor 1	243.00	<input type="text" value="1"/>	243.00
3	<input type="text" value="Select"/> <input type="text" value="4"/> Cote dor 2	99.00	<input type="text" value="2"/>	

◀ ◀ ◀ ▶ ▶ ▶
📄 ✖ ↺

[Close](#)

Setting up multiple tables within one HTML screen can be tricky if you are doing it manually, but it is easy using the browser-client. You can set up one or more subforms, and use a separate eDeveloper task to handle each subform independently. If the data is connected, as it is in our example, then when the data in one form changes the subform data will also change automatically.

There are three steps to setting up a one-to-many task:

- 1.** Create the header task
- 2.** Create the subtask
- 3.** Tie the header and subtask together with a subform control

Let's go through these one at a time.

## 1. Create the Header Task

1. Create the header task as you would any other Browser-client task. In our example, we are displaying one Order header record.
2. On the Order header form, create a table entry in the location you want your repeating data. Be sure to give the table an id, as this will be used in the next step.

## 2. Create the subtask

```

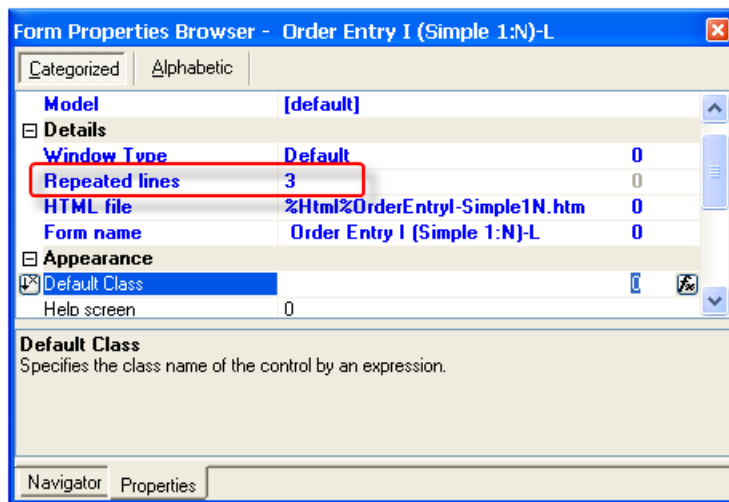
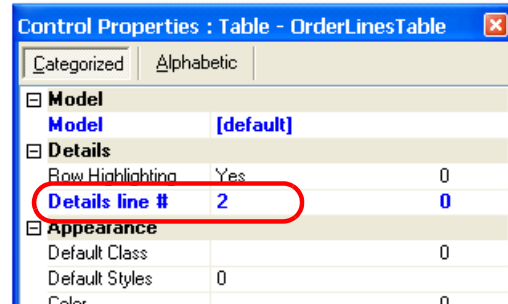
35 <BR>
36 <BR>
37 <table id="OrderLinesTable" border=0 cellspacing=1 cellpadding=0 align=center>
38 <tr>
39 <th class="title-grey-narrow">Row#</th>
40 <th class="title-grey-narrow" width=260>Product</th>
41 <th class="title-grey-narrow" width=100>Price</th>
42 <th class="title-grey-narrow">Qty</th>
43 <th class="title-grey-narrow" width=100>Total</th>
44 </tr>
45 <tr>
46 <td class="leftTD"><input type="text" size="9" name="Row#">
47 <td class="TD" nowrap><input type="text" size="3" name="SelectProduct" href="#">
48 <td class="TD" nowrap><input type="text" size="3" name="Product">
49 <td class="TD" nowrap><input type="text" size="3" name="ProductPrice">
50 <td class="TD" nowrap><input type="text" size="10" name="Qty">
51 <td class="TD" nowrap><input type="text" size="10" name="TotalLine" style="width: 100px;">
52 </tr>
53 </table>

```

#	HTML control n...	Exp	Type	Var	Exp
1	OrderLinesTable	0	Table		0
2	Row#	0	Edit	BI	0
3	SelectProduct	0	Hyper text	???	11
4	Product	0	Edit	BJ	0
5	ProductName	0	Hyper text	BL	0
6	ProductPrice	0	Hyper text	BM	0
7	Qty	0	Edit	BN	0
8	TotalLine	0	Hyper text	???	7

1. Create your subtask as you would any Browser-Client task, selecting the fields you want to display on the form.
2. Select the same HTML page that you used in your parent task.

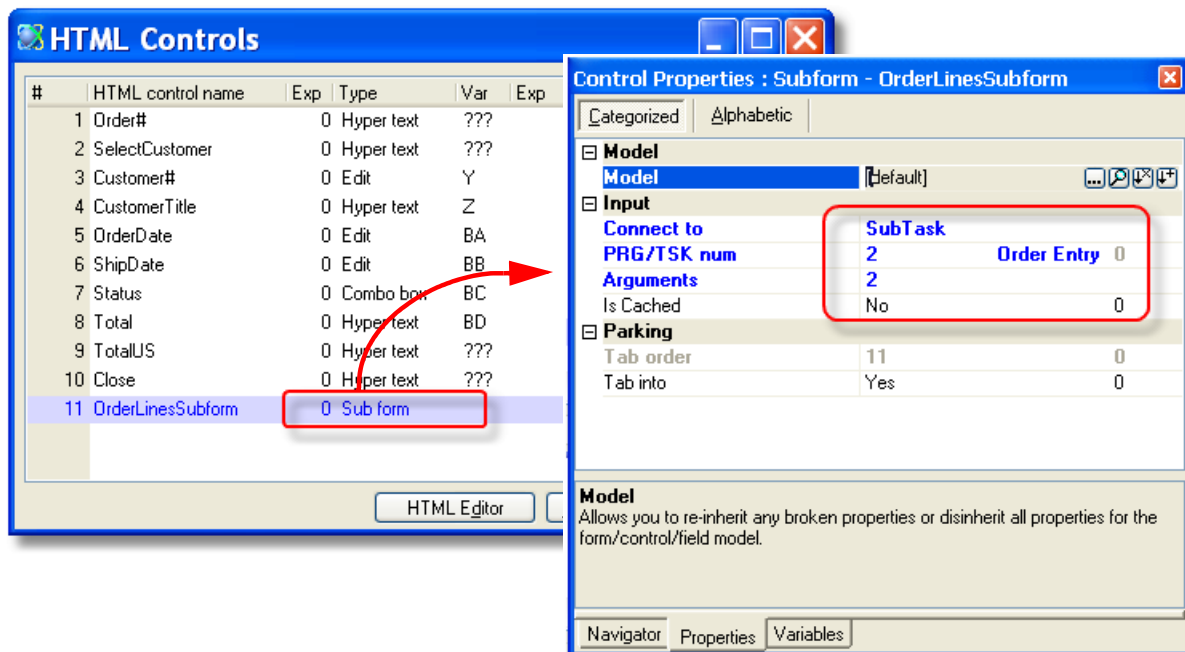
3. Create one HTML control of type *Table*. This control will have the same name as the id= in the HTML table control.
4. In the *Control Properties* for the table, set the *Detail line #* property to indicate which line you want to have repeat. In this case, the first line of the HTML table is the header line, so the second line is the one we are repeating.



5. In the *Form Properties*, set the *Repeated lines* property to show how many times you want the line to repeat.

6. Add controls for each of the data items you want to put on the table.
7. Set up arguments to control the Range and Locate of the task, as you would for any task. In our example, we pass in the start mode and order number, so the order lines only show for the current order and the mode matches that of the parent.

### 3. Tie the header and subtask together with a subform control

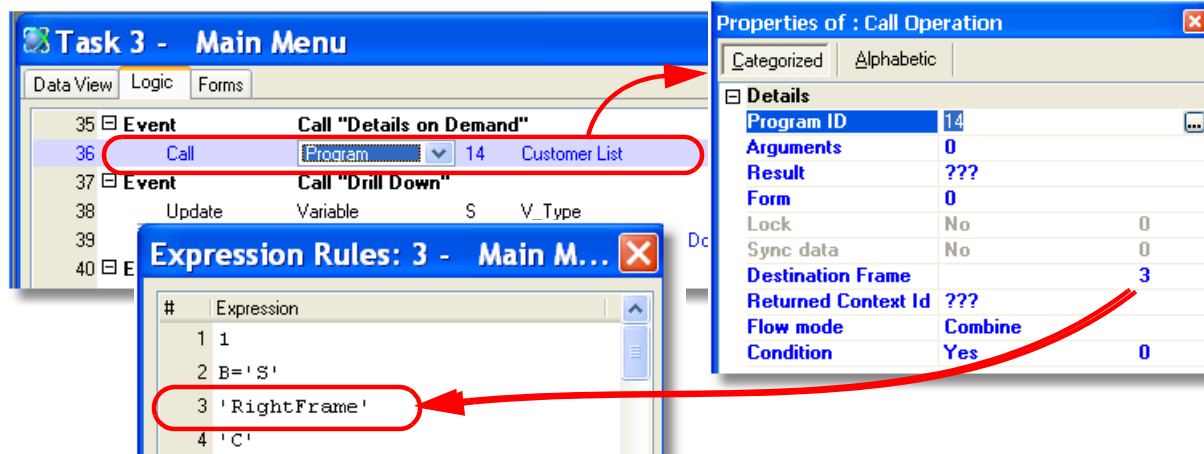


1. Last, you will add a Subform HTML control to your parent task. Press **F4** to add a line to the HTML Controls, then select Subform in the Type column.
2. In the HTML Control properties, set the **Connect to** to **Subtask** if you are calling a subtask, or **Program** if you are calling a program.
3. In the **PRG/TSK num** field, zoom to select the subtask or program you are calling.
4. In the **Arguments** field, zoom to set the arguments you are passing to the called task. In our example, we are passing the mode and order number, so that the subform displays the order lines for this order, in the same mode as the parent task. This methodology is the same as that used in using subforms in online tasks, as explained in Chapter 8, "Subforms" on page 197.

Now when the main task is called, it will automatically display the subtask form as well.

## How do I Prevent a New Window From Being Opened When Calling a Task or Program?

When you call a new task in browser-client, it will by default be opened in a new browser window. However, you can avoid this by opening the new task in the same HTML IFrame as a previous task, overriding whatever was displayed before.



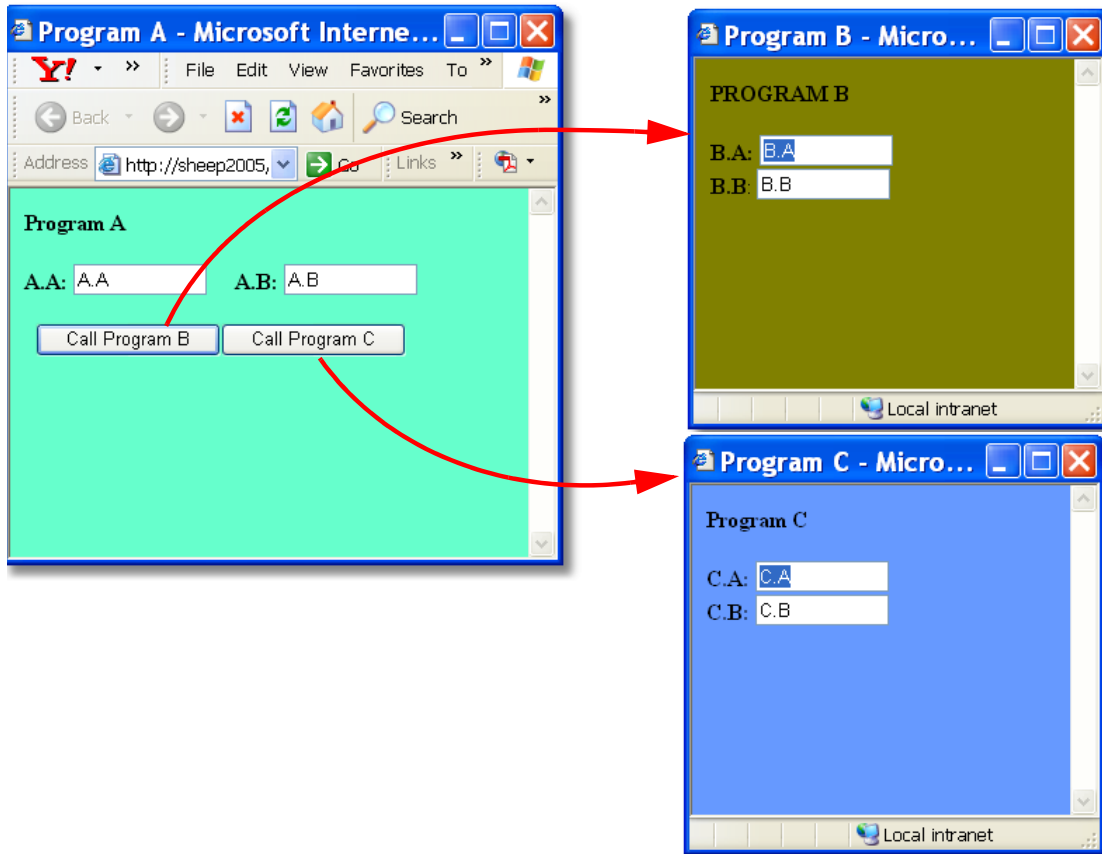
### Calling a program to be displayed in a specific IFrame

1. Create your Call operation
2. Go to the Call Operation properties
3. In the Destination Frame property, type in the name of the HTML IFrame, or create an Expression that evaluates to the name of the IFrame at runtime.

Now, at runtime, the called program will be displayed in the specified IFrame.

**See also:** *The Browser-Client Application Development Methodology* PDF, which is installed in the Demo\_BC\_Methodology directory. The chapter *Windows and Frames* explains popups and frames.

## How do I Display the Interface of different Programs in the Same Window?

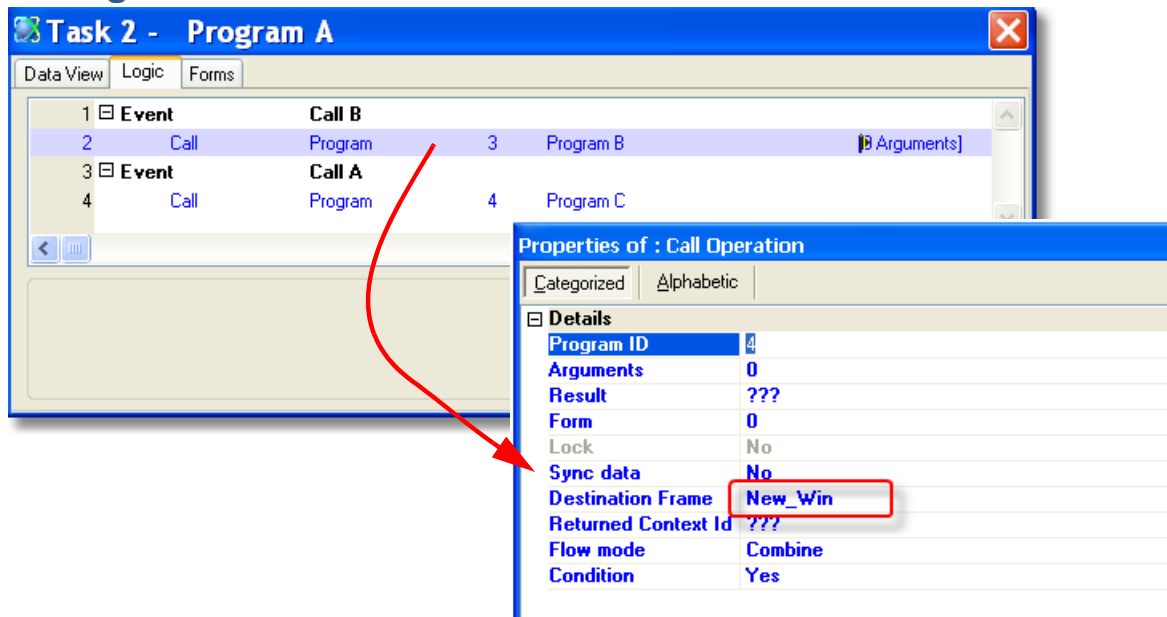


Rather than have a lot of windows open, you may want to re-use one window to display multiple different programs. This is slightly different than using IFrames, as IFrames are partitions of one window. IFrame usage is explained in Chapter 37, “How do I Prevent a New Window From Being Opened When Calling a Task or Program?” on page 906.

In our example, when a user presses Call Program B, then Program B is opened in a window. When the user presses Call Program C, then Program C appears in the same window (the windows are shown separately in our illustration, but for the user they overlay each other as you would expect).

This type of behavior is easily done in eDeveloper.

## Re-using a Browser Window



To call a different programs using the same window, just use the same Destination Frame.

In our example, we have two events, each of which is raised by a different push button. Each event calls a different program, and each program has its own HTML to control the display. But in the Call Operation Properties, both have a *Destination Frame* of “New\_Win”.

“New\_Win” is not an IFrame, nor is it entered anywhere in the HTML. It is just a random name ... any string of characters would work. By specifying the same name in two program calls though, you are telling eDeveloper to use the same window for both calls. When the Destination frame is a new string of characters, a new window is opened: when the string is the same as the previous frame, the frame or window is re-used.

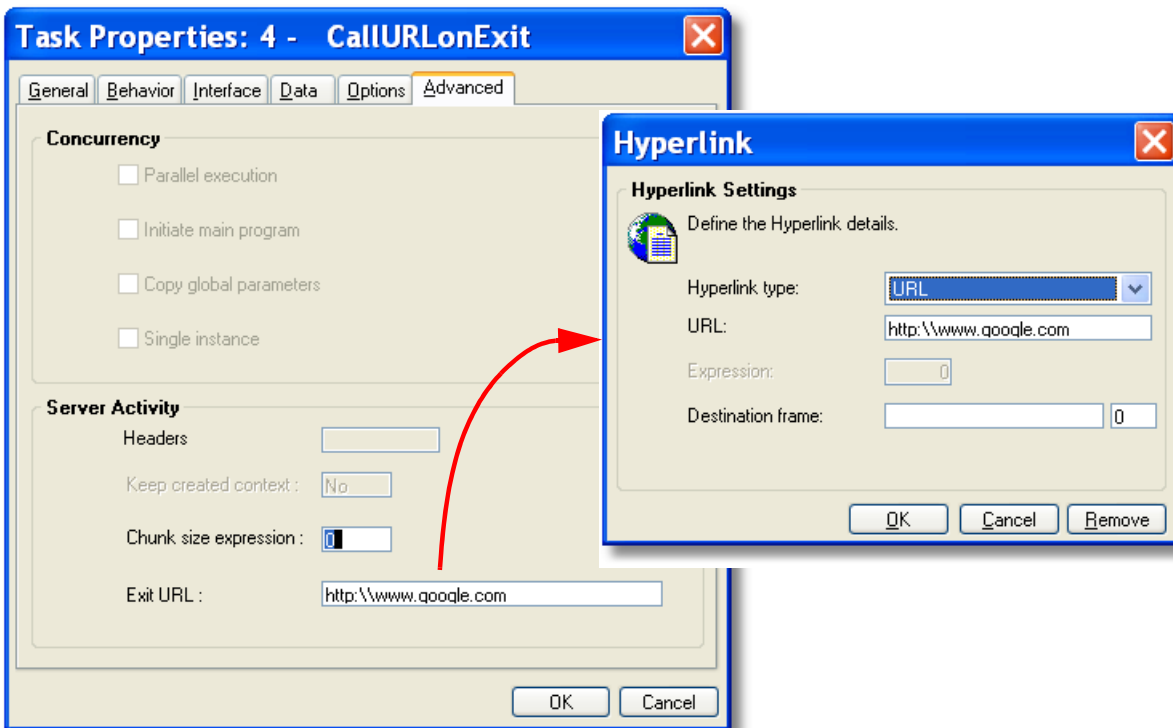


## How do I Direct the Browser to a Given URL When Closing the Top Level Program?

When you are using eDeveloper Client-Server, most browser windows are interactive, and the window is waiting for a result from the user. However, when the user is exiting, you will often want to show some window that is just some plain HTML, and not part of the Browser-Client loop. Or, you may want to invoke a different eDeveloper application or program to handle the task when the user exits. This is easily done using **Task Property->Exit URL**.

**Note:** The Exit URL hyperlink doesn't work when the user exits the window by clicking on the X. That sort of exit isn't under the programmer's control (to prevent creating unclosable windows). eDeveloper can, however, react to the internal Exit event.

### Displaying a URL on Exit

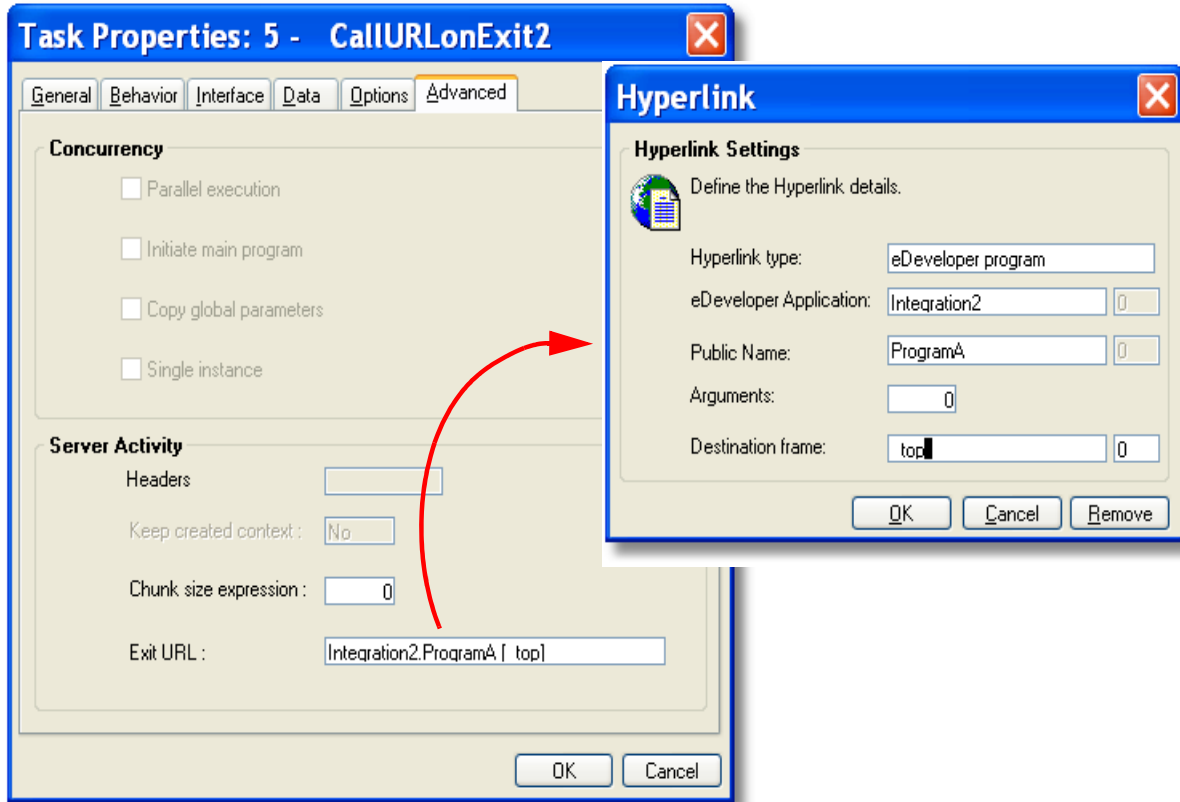


To call a URL when the program ends:

- Zoom on **Task Properties->Advanced->Exit URL**. A Hyperlink dialog will appear.
- For Hyperlink type, select **URL**.
- Enter the URL you want to call.
- Enter the Destination frame, if you want the URL to appear in a different window or frame. Otherwise it will appear in the current window or frame.

Now, when the program exits, control will go to the URL specified.

## Exiting to an eDeveloper Program



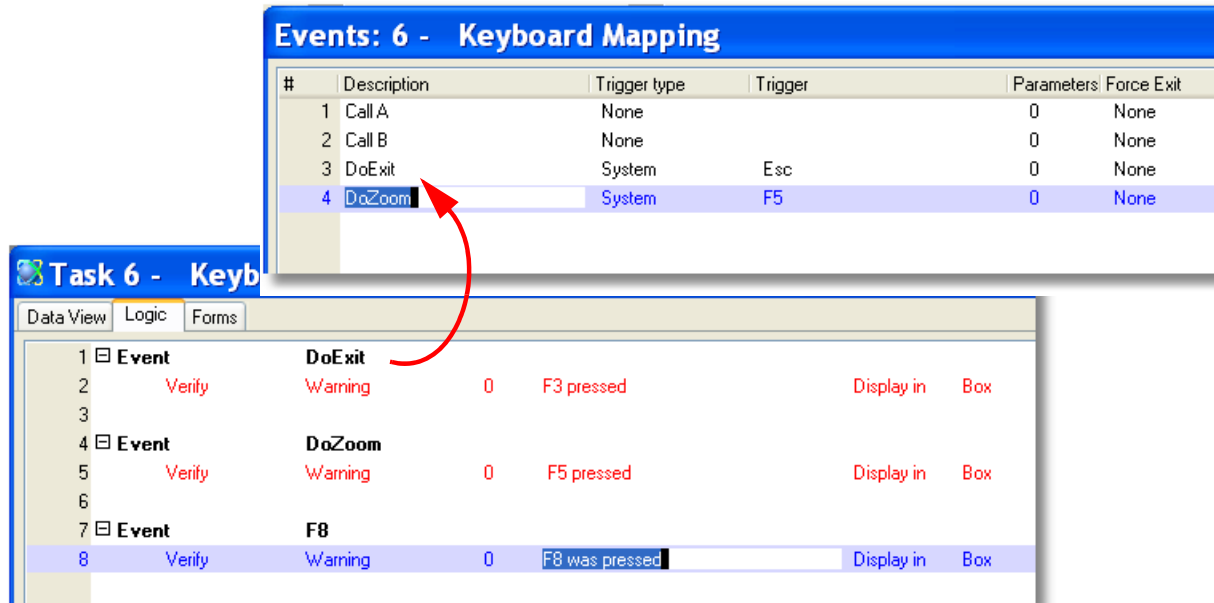
To call another eDeveloper program when the program ends:

- Zoom on **Task Properties->Advanced->Exit URL**. A Hyperlink dialog will appear.
- For Hyperlink type, select **eDeveloper program**.
- Enter the eDeveloper Application name. The program you are calling can be in the current application, or a different one.
- Enter the Public Name of the program you want to call, and any argument string you want to pass.
- Enter the Destination frame, if you want the URL to appear in a different window or frame. Otherwise it will appear in the current window or frame

Now, when the program exits, the eDeveloper program will be executed.

## How do I Map Keyboard Strokes to eDeveloper Internal Events?

When you are working in a Browser, the usual default eDeveloper client-server keyboard mappings don't apply. For instance, in most Browsers, F5 is mapped to "View Refresh", not Zoom.



However, you can set the keyboard mapping to do what you like by using Events. You can set up these events just as you would for an online program.

### Using a keystroke in a Logic Unit

You can enter the keystroke directly on your Event logic unit as follows:

- Press **Ctrl+H** to create a Logic unit header.
- Type **E** to select **Event**. An Event dialog will appear.
- Select System for the **Event type**. Then tab. The Key definition dialog will appear.
- Press the key or key combination you want to use for this event.

Now, when the user presses that key combination, the Event logic unit will be triggered.

### Using a keystroke in a User Event

You can also use the keystroke to trigger a user event. This can be useful when a particular user event can be triggered multiple ways. For instance, you might want a certain program to be called by a pushbutton or by a standard function key or by being triggered from another program.

- Press **Ctrl+U** to go to the user events.

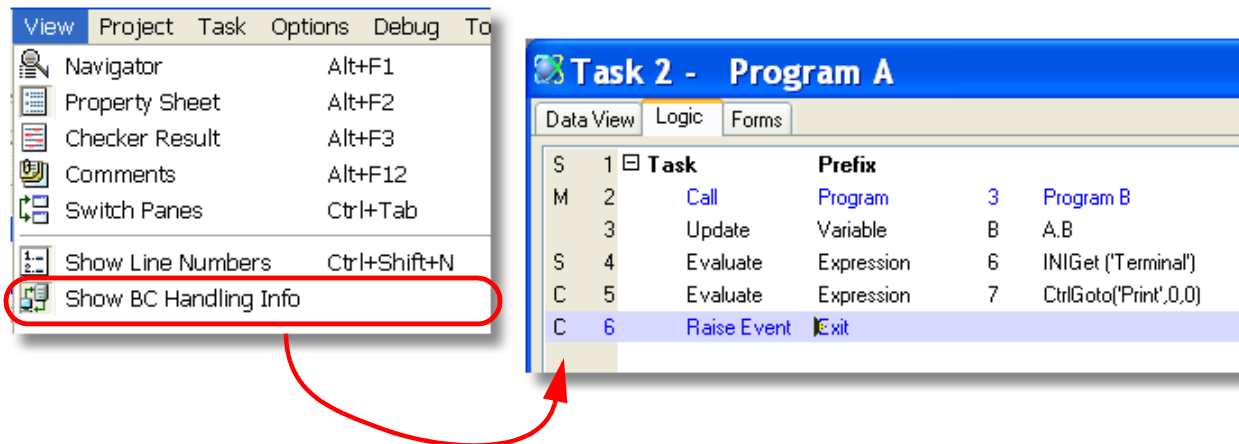
- Press F4 to open up a line.
- Give the user event a name. Then tab to the next field.
- Select System for the *Trigger type*. Then tab.
- Zoom from the *Trigger* field. A key definition dialog will appear.
- Press the key or key combination you want to use for this user event.

Now, when user presses that key or key combination, the user event will be triggered.

## How do I Distinguish Between Server Side and Client Side Operations and Functions?

When you create a Browser-client program, some of the operations will be executed on the Client using the eDeveloper applet, while other operations are executed on the Server. Since this may affect how fast the application runs, it's good to know where each operation is executing.

You can see which functions execute on the server and which on the client by looking in the Help facility, under the "Server-Side Functions" and "Client-Side Functions" entries. However, you can see where each operation is executed by using the **Show BC Handling Info** option.

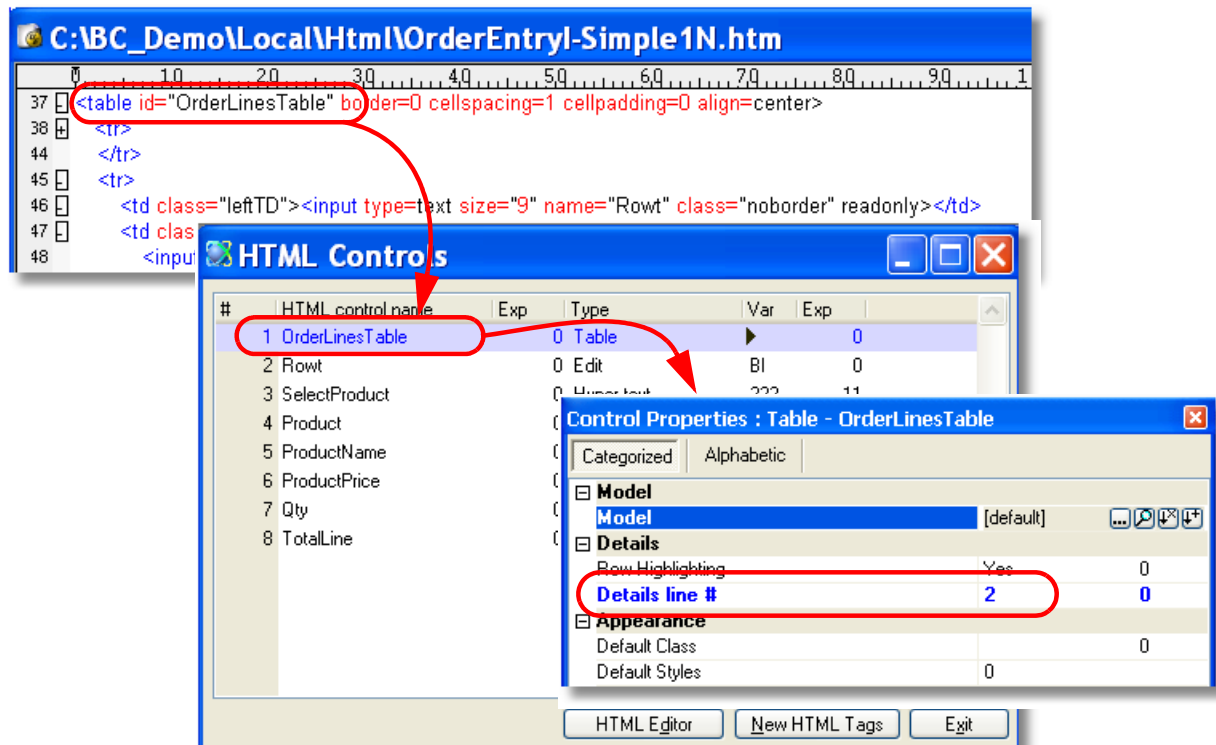


Set **View->Show BC Handling Info** if it is not already set.

When the option is set, you will see **S**, **M**, or **C** in the margin of the entries in the Logic tab.

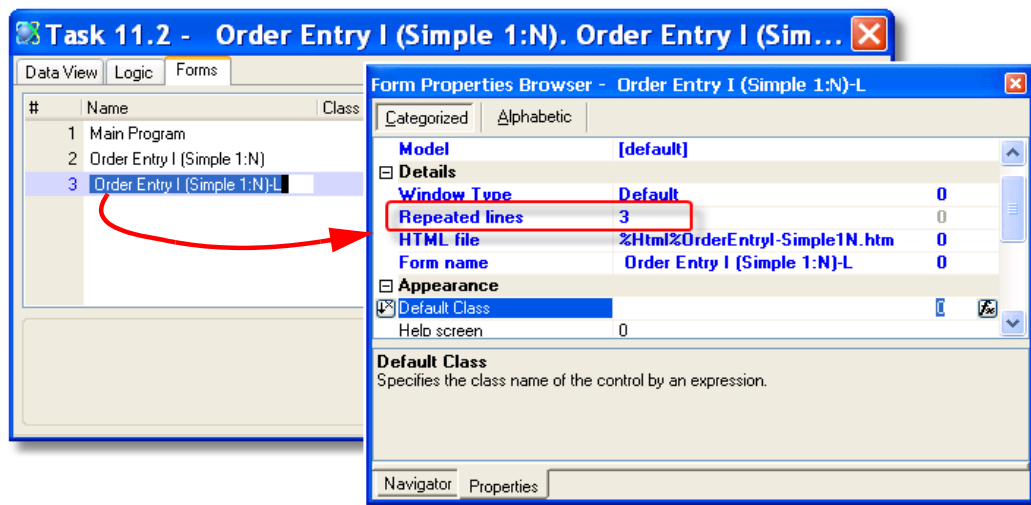
- **S**: indicates that the operation will execute on the server
- **C**: indicates that the operation will execute on the client
- **M**: indicates mixed-mode handling
- **blank**: indicates the operation could be executed on either the client or the server.

## How do I Set the Number of Repeatable Records in a Table?



A table in a Browser-Client program is an HTML table. The HTML table is referenced in the eDeveloper HTML Controls via the `id=` HTML attribute.

Within the **Control Properties** for the table control, the **Details line#** property indicates which line to repeat. In this example, there are two `<td>` rows in the table. The first row contains the table header. The second line contains the row to repeat. So we set **Details line#** to 2



Now, within the Form that contains the table, set the Repeated lines property to the number of times that line should repeat. In this instance, the line will repeat 3 times.

Row#		Product	Price	Qty	Total
1	Select	1 Cadbury - Nuggets	45.00	2	90.00
2	Select	3 Cote dor 1	243.00	1	243.00
3	Select	4 Cote dor 2	99.00	2	

Here is the result. The 2nd HTML row repeats three times.

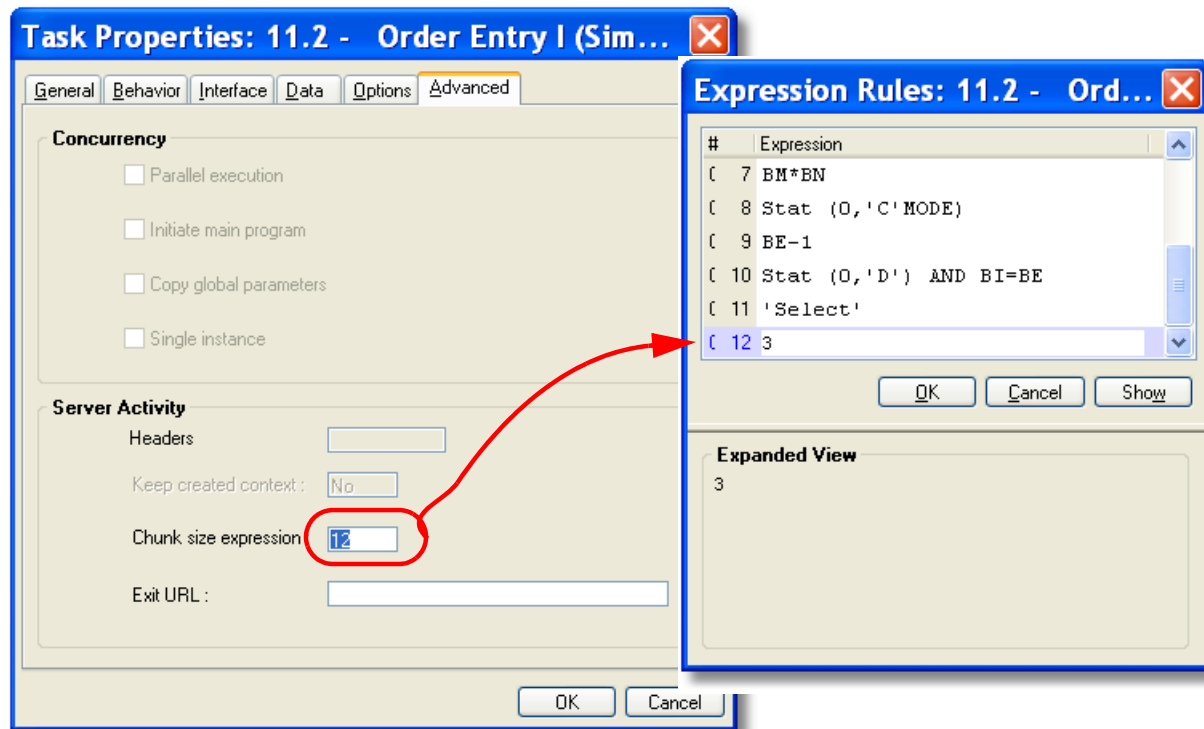
## How do I Set the Number of Records in a Record Set that are Passed to the Browser?

One of the reasons that the browser client gives good response to the user is that a “chunk” of data is sent to the client, not just the records that are currently visible. This means that when the user scrolls down a table, there is immediate response.

For smaller tables, it isn't a problem if all the records are sent at once. The records are encrypted and compressed, and they transmit very quickly. However, if the table is very large, then sending all the records can take some time. The ideal size of the “chunk” of records kept in the client cache depends on several factors, including the size of each record, and how much the user is expected to scroll through the data.

If no chunk size is set, then it defaults to 30 for a task with a Main source, or 1 for a task with no Main source.

### Setting the Chunk size



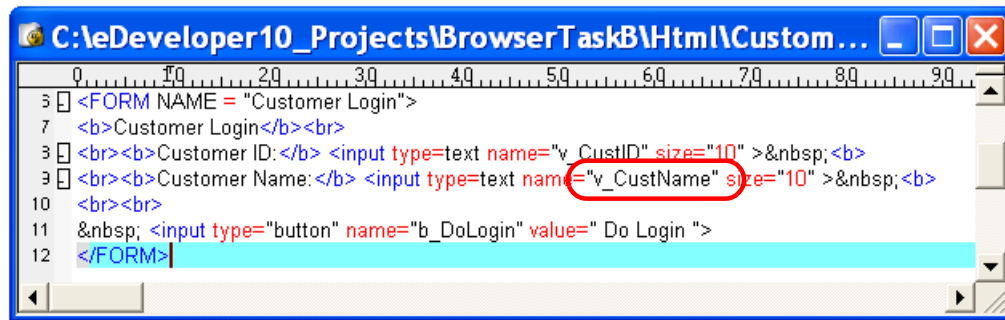
- Go to **Task Properties->Advanced**
- Zoom from the **Chunk size expression** field to the Expression Rules.
- Press **F4** to open up a line. Type in the number of records you want fetched in each chunk.
- Press **OK** to close the **Expression Rules** and bring back the expression number.
- Press **OK** to close the **Task Properties**.



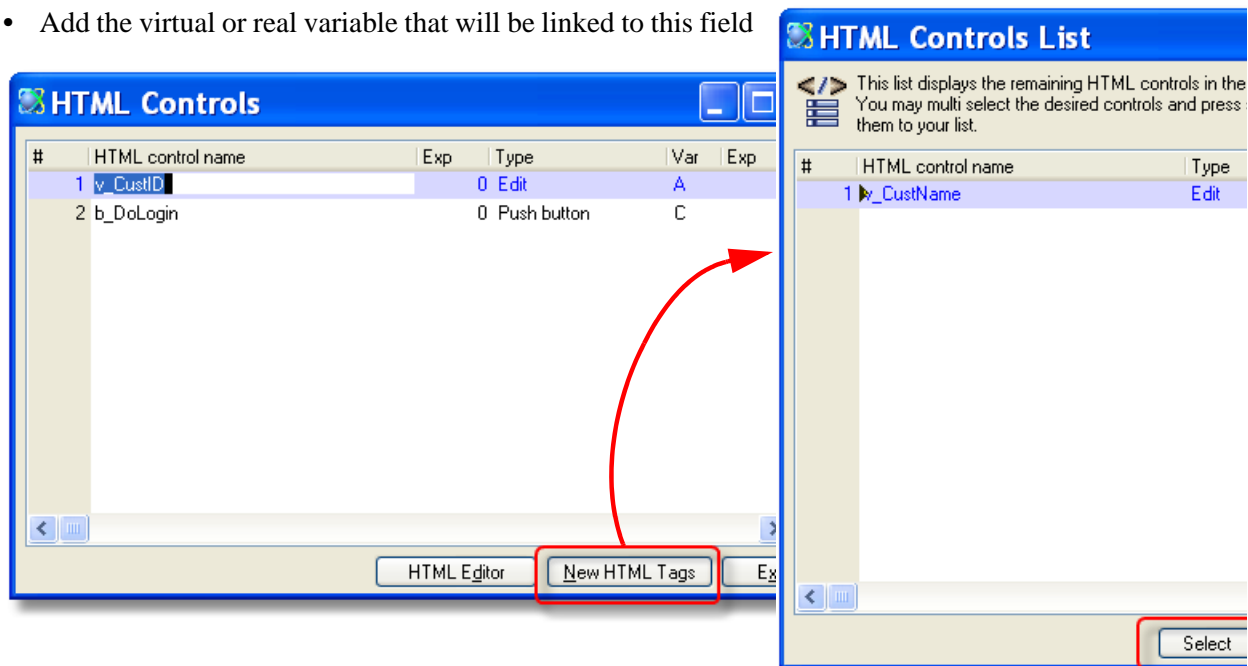
## How do I Add Controls to an Existing Form?

After you have created a Browser-client program, you may need to add more controls to the HTML page. When you do this, you will need to then modify your eDeveloper program to recognize and work with these new controls. Here is how to do it.

### Adding a new control



- Add the new control to the HTML. Set the **name=** attribute to what will be used to link to the element in eDeveloper. In our example, the name is “v\_CustName”.
- Add the virtual or real variable that will be linked to this field



- Go to the Form in your eDeveloper program.
- Zoom from the Form name. This will bring up the existing HTML controls.

- Move the cursor to the line above where you want the new entry to be in the HTML controls list. Then press the **New HTML Tags** button. This will bring up a list of all the tags that are not currently referenced in the HTML controls list. Highlight the one you want, then click Select. The control will be brought back to the HTML controls list. You can use **Ctrl+Click** to select more than one control if you need to.
- Zoom from the **Var** column to select the variable that will attach to this control. Alternatively, you can zoom from the **Exp** column to use an expression.

Now, the new control will function as part of the browser task.

## How do I Implement Styles and Classes?

Within HTML, there are several different ways to implement styles and classes. For instance, you can:

- Add color and font attributes to each field tag.
- Use classes to access Styles to apply to multiple tags.
- Use Style sheets to apply Styles to multiple HTML files.
- Use Scripts to change fonts and colors interactively

If you are using eDeveloper, these HTML options still exist, and in addition you can access the coded HTML features from within eDeveloper. Additionally, you can use styling within eDeveloper independently of any HTML code.

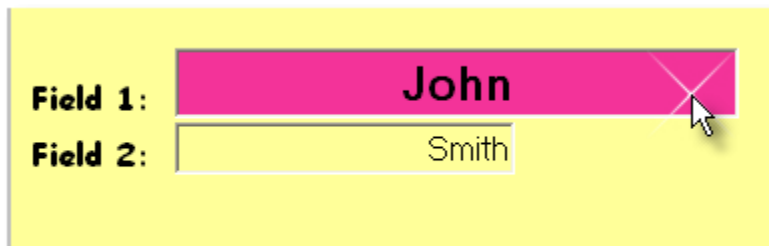
Which option you want to use might depend on how much flexibility you need, and on the general design of the system. As a general principle you want the system to be as easy to maintain as possible, so keeping most formatting at a high level -- using eDeveloper Models and/or HTML Style sheets -- is the best approach. But some individual fields won't fit into any one Style, so you will want to format them at the field level.

Let's see some of the ways to implement styles and classes in eDeveloper.

### The Style Sheet

```
.formreg { border:thin dashed silver; font-family:"Comic Sans MS",Arial; background-color:#FFF99; }  
.bighdr { text-align:center; color:black; background-color:red font-family:'Century Schoolbook',serif; font-size:18pt; }  
  
.regtext { text-align:left; font-size:12pt; background-color: #FFFFFF; }  
.overtext { text-align:center; font-size:18pt; font-weight:bold; background-color: #F33399; }  
.outtext { text-align:right; font-size:12pt; background-color:#FFF99;}
```

Normally, the style sheet will be in a separate CSS file, so it can be used with multiple web pages. In our example, all the programs use the same simple style sheet, shown above.



The styles are used to create a simple web effect such that when the cursor hovers over an input field, the field gets big and turns pink, as shown here, where the cursor is hovering over Field 1.

Now we'll see three different ways to code this effect.

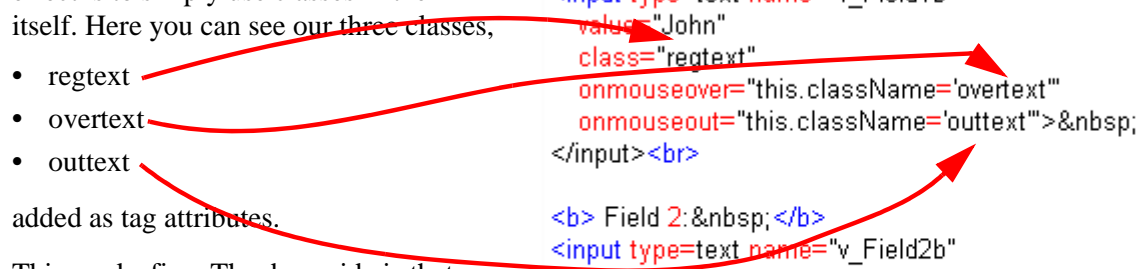
## Using HTML Classes

The first way we can use styles to code this effect is to simply use classes in the HTML itself. Here you can see our three classes,

- regtext
- overtext
- outtext

added as tag attributes.

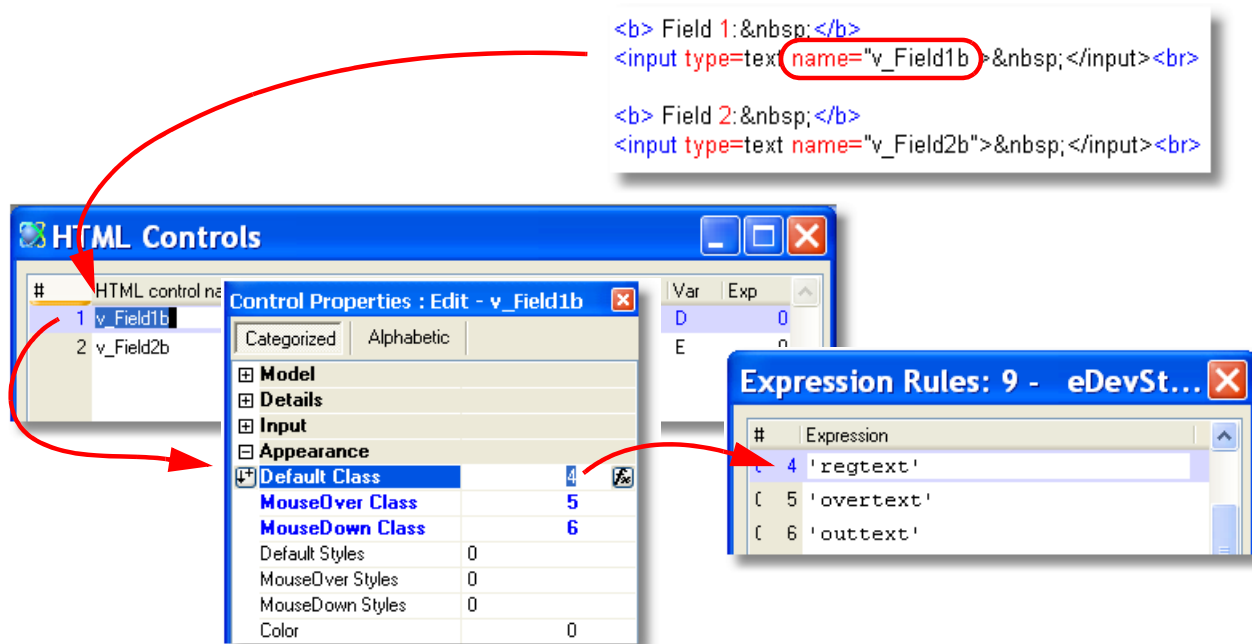
This works fine. The downside is that we need to repeat the typing for every tag that needs it, and we can't change the value from inside eDeveloper at runtime.



```
<b> Field 1:&nbsp;</b>
<input type=text name="v_Field1b"
value="John"
class="regtext"
onmouseover="this.className='overtext'"
onmouseout="this.className='outtext'">&nbsp;</input><br>

<b> Field 2:&nbsp;</b>
<input type=text name="v_Field2b"
value="Smith"
class="regtext"
onmouseover="this.className='overtext'"
onmouseout="this.className='outtext'">&nbsp;</input><br>
```

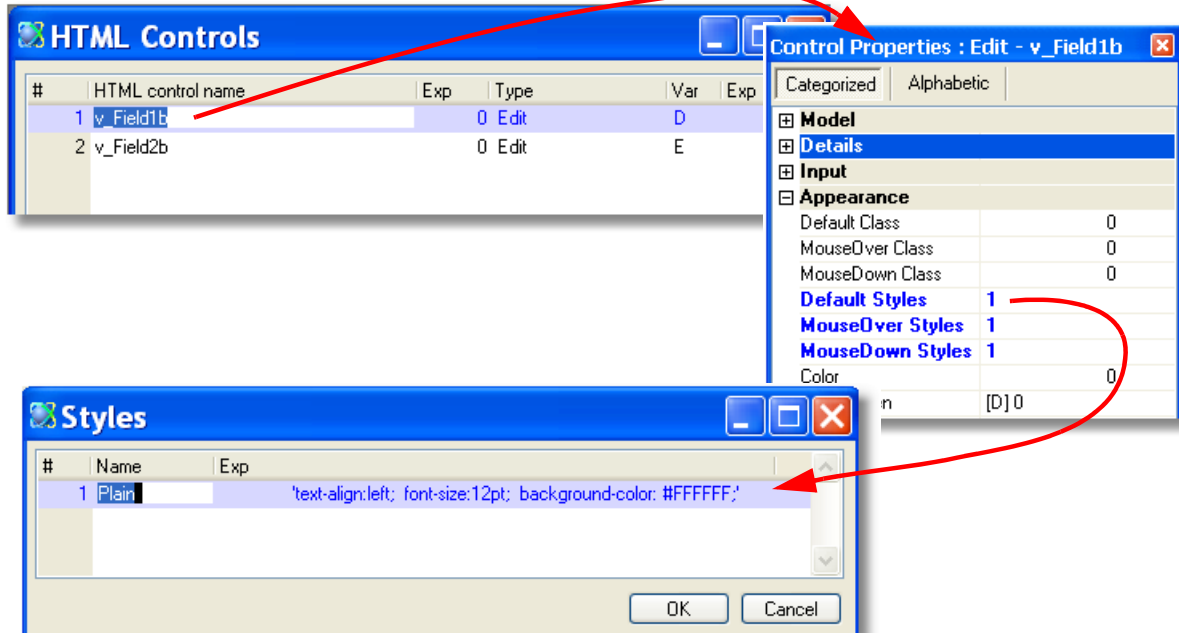
## Using Classes in eDeveloper



This example works exactly like the last example, except instead of coding the class names in the HTML, we have added them to the Control Properties for the field. Each control property points to an expression, which is the name of the class.

The expression can point to a global virtual which evaluates to the style name also, so the exact style name doesn't need to be entered in different programs.

## Coding the Style in eDeveloper



Here, we have a program that runs just like the previous two. However, it doesn't use a style sheet at all. Instead, the style is coded inside eDeveloper only. Zooming on the Default Styles, MouseOver Styles, or MouseDown Styles properties brings up a dialog where you can code the actual styles as Expressions. The style entered here is the same string that was used in the CSS style sheet shown earlier. It could also be held in a global variable. The Style Name to the left is for documentation only.

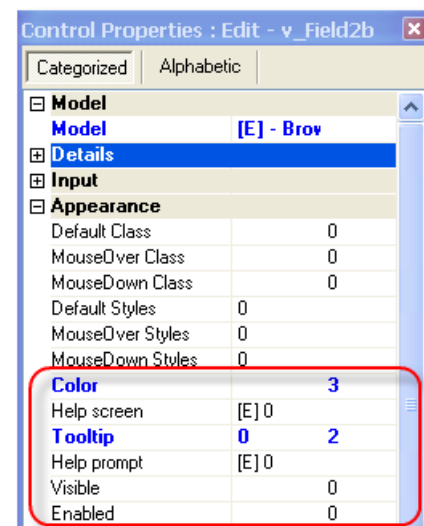
## Using other Control Properties

Another option is to use the other eDeveloper properties, rather than using the HTML styles. These properties give you a lot of runtime control over the HTML, without coding anything extra into the HTML.

This set of properties works in the browser much as the same properties work in eDeveloper client server.

For instance, using the **Color** property, you can use an expression to set the color of the field based on values of other fields. This option uses the eDeveloper colors found in **Options->Settings->Colors**, rather than using the HTML color settings.

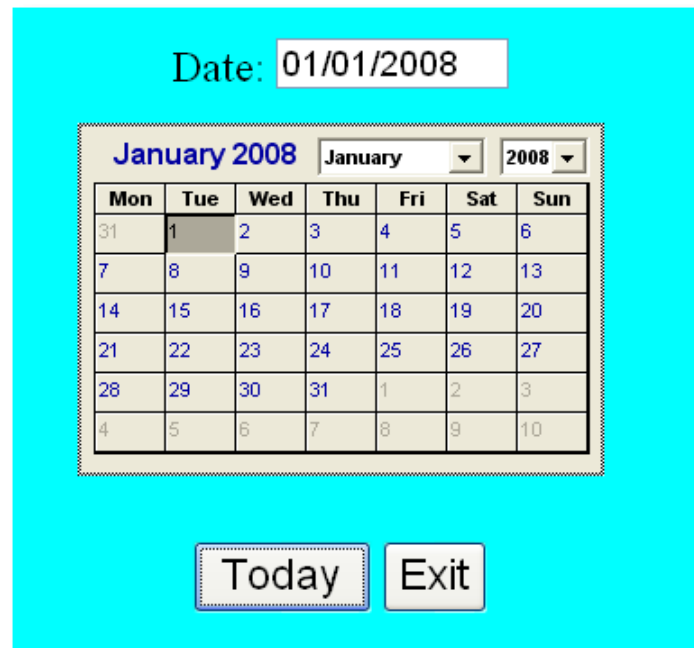
The **Help screen** property allows you to point to a URL Help entry, so that when the user presses **F1**, the user can see a new window



containing the help screen for this field. **Tooltip** allows you to point to a Tooltip help entry, or simply enter an expression to display a tooltip.

You can use the **Visible** property to make the field appear only in certain circumstances, and **Enabled** to allow or disallow data entry.

## How do I Implement ActiveX Controls on a Page?



You can implement ActiveX controls on an HTML page rather easily. There are three steps to working with an ActiveX control:

1. First, you need to put the ActiveX control on the HTML page.
2. Second, you need to use VBScript or JavaScript to work with the control.
3. Third, you use the CallJS commands in eDeveloper to work with the scripts, to handle the control

Let's see how this works.



## Putting an ActiveX control on an HTML Page

```
<form name = "BC External Event">
<br><font size="5">Date: <input type=text size="10" name="Date" style="font-size: 14pt"
onclick=document.Calendar1.Day=8><br>
</font>
<p>
<object classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02" id="Calendar1" width="288" height="192">
  <param name="_Version" value="524288">
  <param name="_ExtentX" value="7620">
  <param name="_ExtentY" value="5080">
  <param name="_StockProps" value="1">
  <param name="BackColor" value="-2147483633">
  <param name="DayLength" value="1">
  <param name="MonthLength" value="2">
  <param name="DayFontColor" value="0">
  <param name="FirstDay" value="1">
  <param name="GridCellEffect" value="1">
  <param name="GridFontColor" value="10485760">
  <param name="GridLinesColor" value="-2147483632">
  <param name="ShowDateSelectors" value="-1">
  <param name="ShowDays" value="-1">
  <param name="ShowHorizontalGrid" value="-1">
  <param name="ShowTitle" value="-1">
  <param name="ShowVerticalGrid" value="-1">
  <param name="TitleFontColor" value="10485760">
  <param name="ValuesIsNull" value="1">
</object>
```

First, you have to insert the ActiveX object into your HTML. This is done with an `<object>` tag. The **classid** attribute specifies the universal identifier of the object. The `<param>` tags specify various properties of the object.

Each object will be different, depending on how it was written. You need to consult the documentation that comes with the ActiveX object to know what you can do with it.

## Working with the ActiveX control with a script

```
| <SCRIPT LANGUAGE="JavaScript">  
//This JavaScript segment provides a function that manipulates the embedded Calendar object  
//This allows the Magic engine to easily interact with external modules on the page.  
  
function Calendar_update(MGDay,MGMonth,MGYear)  
{  
  document.Calendar1.Day=MGDay;  
  document.Calendar1.Month=MGMonth;  
  document.Calendar1.Year=MGYear;  
}
```

Once you have the ActiveX object declared, you can work with it using scripts. Most of the scripts will operate locally on the page, but you can call them from eDeveloper and they can, in turn, raise an event in eDeveloper.

This script, Calendar\_Update, takes three parameters, and changes the date of the calendar object.

### *Calling the script from eDeveloper*

⊟ Event	Today	Scope: SubTree		
Update	Variable	A	Date	With: 4 Date ()
Update	Variable	D	v_RC	With: 3 CallJS ('document.Calendar1.Tod

Here is the code in eDeveloper that calls the JavaScript function, using the **CallJS** function. The event itself is triggered when the user presses the “Today” button.

### *Handling events from the script in eDeveloper*

The script can also raise events that will be handled in eDeveloper. This is explained in Chapter 37, “How do I Invoke eDeveloper Logic from an External Script in an HTML Page?” on page 927.

## How do I Invoke eDeveloper Logic from an External Script in an HTML Page?

```
<SCRIPT LANGUAGE="VBScript">
//This VBScript segment handles the click event of the Calendar object.
//Upon a click on the object the MGExternalEvent function is called with the expected arguments.
//This allows external modules on the page to interact with the Magic engine.

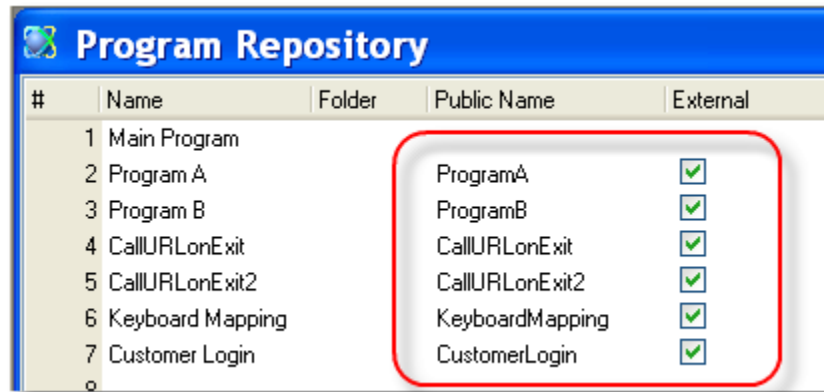
Sub Calendar1_click()
call window.MGExternalEvent(document.Calendar1.Day,document.Calendar1.Month,document.Calendar1.Year)
end sub
</SCRIPT>
```

C	4	Event	External Event	on:	Scope:	Su	
	5	Variable	Virtual	5	Day	Alpha	2
	6	Variable	Virtual	6	Month	Alpha	2
	7	Variable	Virtual	7	Year	Alpha	4
	8	Update	Variable	A	Date	With: 1	DVal (Day&Month&Year,
C	9	Raise Event	Refresh Field				Wait:
	10						

From inside your HTML, you can use a special script called MGExternalEvent. Calling this script will activate an event in eDeveloper called External Event, and pass the parameters you specified.

In this example, a VBScript script calls the MGExternalEvent script, passing 3 parameters, the Day, Month, and Year. The event handler catches those three parameters, and responds by updating a field and refreshing the screen.

## How do I Allow a Program to be Called Externally?



The screenshot shows a window titled "Program Repository" with a table containing the following data:

#	Name	Folder	Public Name	External
1	Main Program			
2	Program A		ProgramA	<input checked="" type="checkbox"/>
3	Program B		ProgramB	<input checked="" type="checkbox"/>
4	CallURLonExit		CallURLonExit	<input checked="" type="checkbox"/>
5	CallURLonExit2		CallURLonExit2	<input checked="" type="checkbox"/>
6	Keyboard Mapping		KeyboardMapping	<input checked="" type="checkbox"/>
7	Customer Login		CustomerLogin	<input checked="" type="checkbox"/>
8				

A red rectangle highlights the "Public Name" and "External" columns for rows 2 through 7.

In order to allow an eDeveloper program to be called from outside the eDeveloper application, such as from a Web browser, you need to do two things:

1. Give the program a Public Name
2. Check the External checkbox.

That's all there is to it! You can tell at a glance, looking at the Program Repository, which programs are accessible from outside the application.

## Chapter 38: The Broker

---

### How do I Configure an IIS Web Server so eDeveloper Will Receive Requests?

Before you can work with the eDeveloper broker, you need to have the Web server properly configured. There are several parts of this process:

1. The IIS Web Server, must be installed, before eDeveloper is installed.
2. The Internet Requesters for eDeveloper need to be installed.
3. After the installation, check the installation and make sure the server is running.

Let's look at each of these steps one by one.

#### 1. Installing up Microsoft IIS

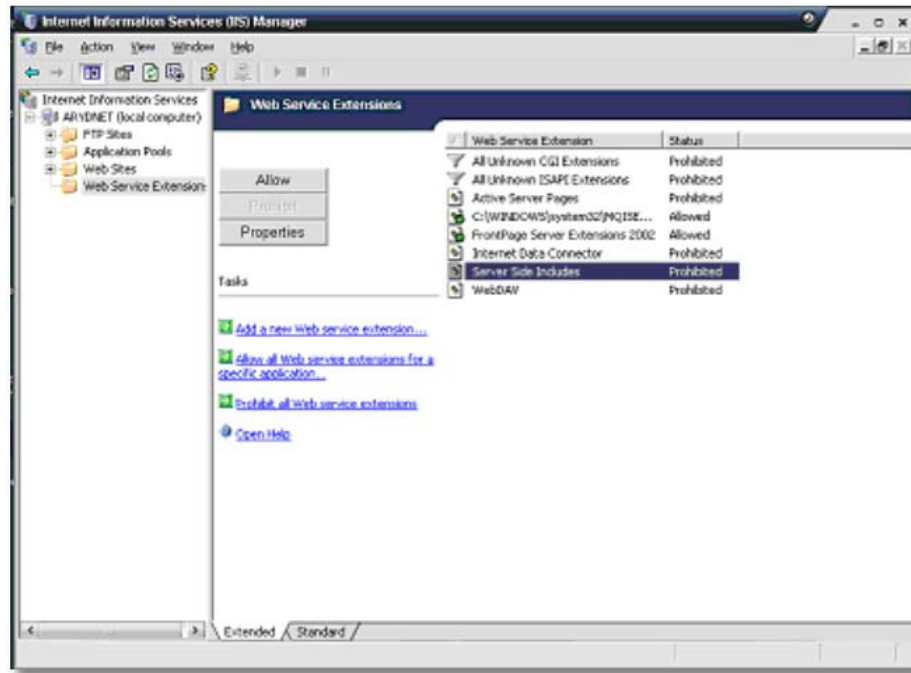
IIS stands for Internet Information Services. It is a Microsoft Component that allows your computer to act as a Web server, and most Windows computers today have IIS installed as part of the operating system. You can check by looking on **Start->Control Panel->Administration->Internet Information Services**.

If IIS is not installed, you can install it using **Start->Control Panel->Add or Remove Programs**. In the margin you'll see the option for *Add/Remove Windows Components*. Follow the dialog to install IIS.

The eDeveloper installation, automatically does the required setup if it detects that IIS is installed on the Windows server and the developer chooses to install the eDeveloper Web requesters (for example ISAPI Web requester).

The eDeveloper setup will also define the required web aliases, such as the "eDevScripts" alias, with the proper permissions (for Browser client, "eDevCache" is also required).

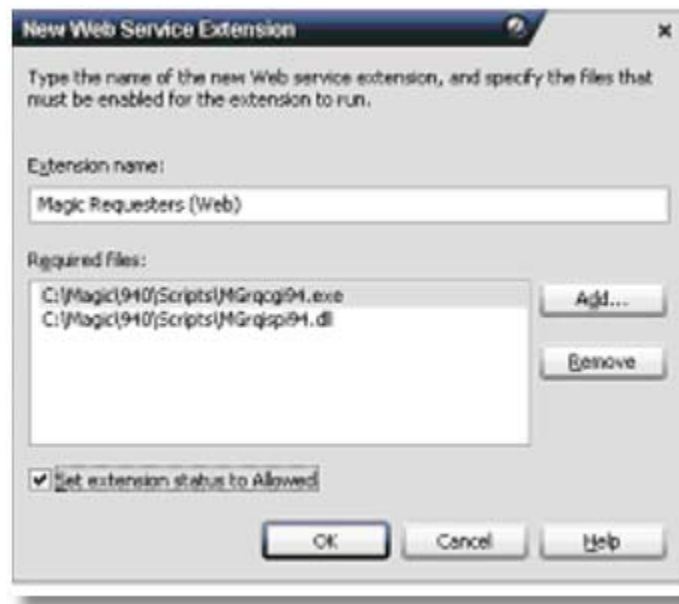
## Microsoft Security Issues



A security enhancement for IIS prevents DLL files and executables from working unless they are explicitly allowed. Implementation Steps

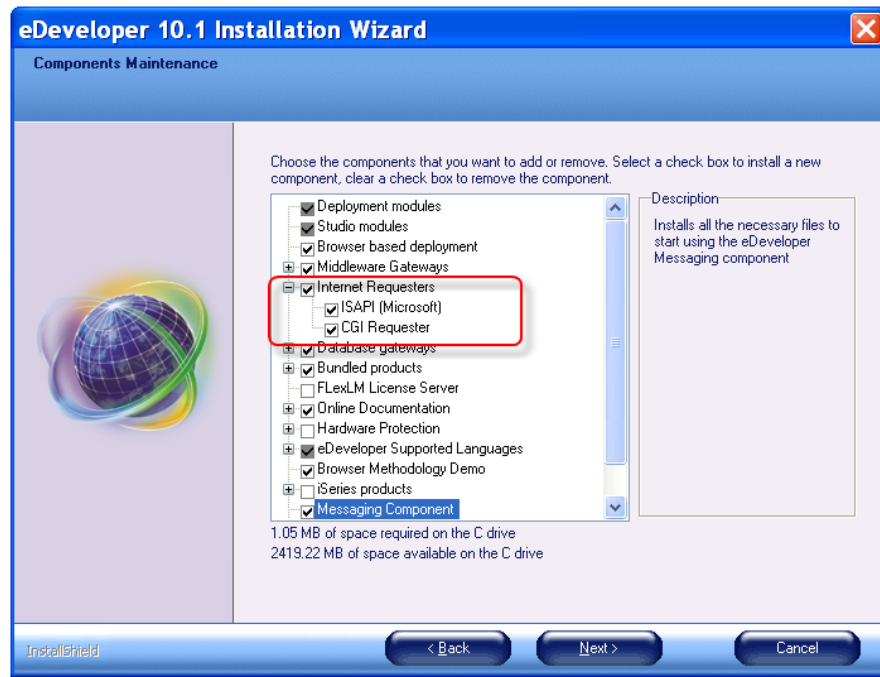
The steps below show how to configure the IIS 6.0 to work with ISAPI and CGI requesters.

1. Right-click the *My Computer* icon and select *Manage*.
2. Go to **Services and Applications->Internet information service->Web Services Extensions**.
3. Click the Add a new Web service extension task.
4. Click the Add button and add the *MGrqispi101.dll* file.



When you are done, you will see the Magic requesters listed under the Web Service Extensions, with the Status “allowed”.

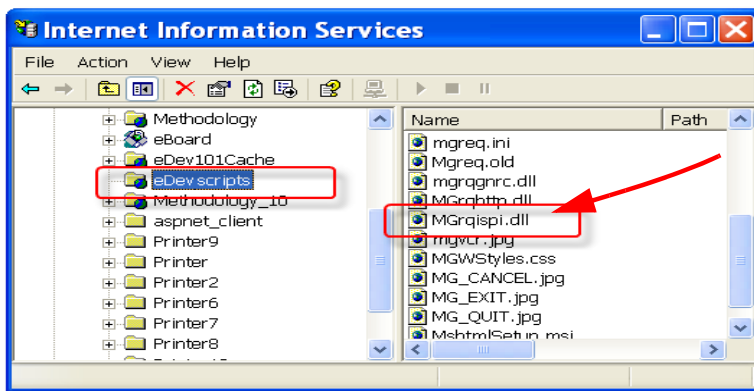
## 2. Installing eDeveloper



Now, when eDeveloper is installed, make sure that the appropriate Internet Requesters are installed. Check the ISAPI box for Microsoft IIS. CGI is for Apache.

If you have already installed eDeveloper, you can add the requestors by:

- Go to *Control Panel->Add or Remove Programs->Developer->Change/Remove*.
- Select the *Modify* installation option.
- Add a check before the ISAPI requestor, as shown above.
- Do NOT uncheck any option: that will cause that option to be un-installed.



When the requester is properly installed, you will see the MGrqispi DLL installed in the IIS directory.

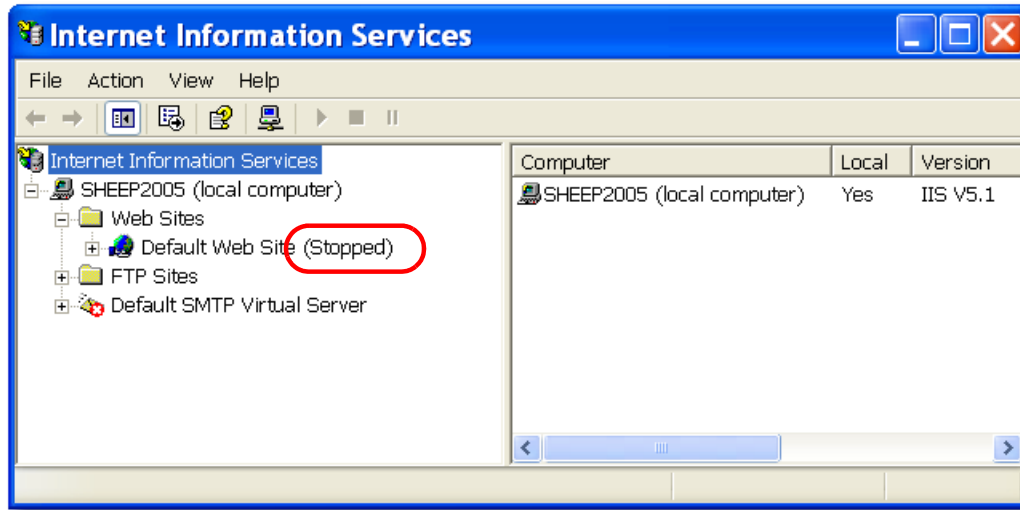
This is the DLL that is called in the Internet request under IIS.



## Check that the Broker is Running

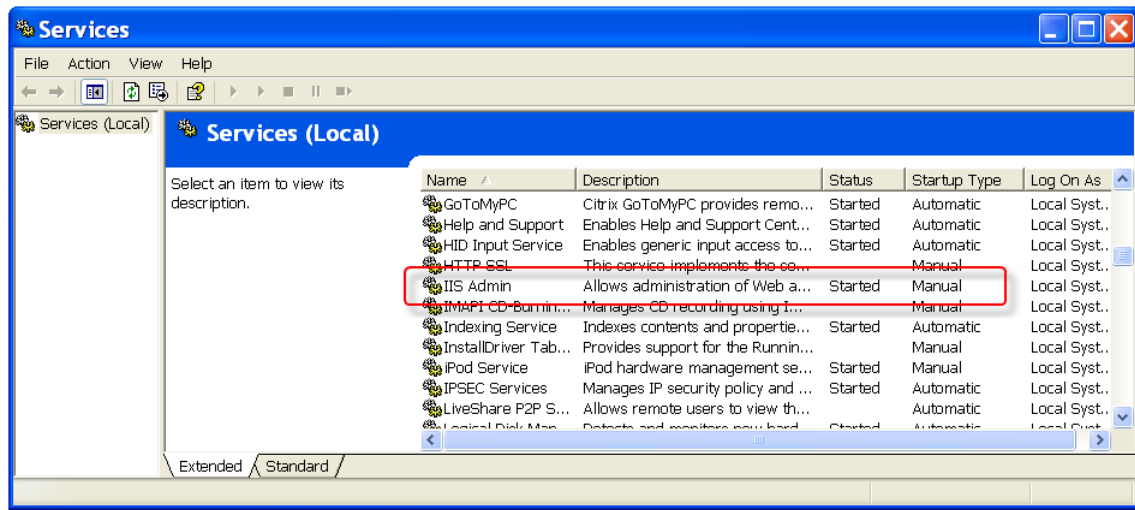
Depending on how you installed eDeveloper, the Broker will either run as a Service, or as an executable. If it runs as a Windows Service, then it can start automatically. If it is installed as an executable, you'll need to start it manually. There is an option on the **Start->eDeveloper 10->Broker** menu to start and stop the Broker.

You can ensure the Broker is running, and watch as it runs, by starting the Broker Monitor. You can read more about the Monitor in Chapter 38, "How do I Monitor Broker Activity?" on page 939.



You can check if the server is running by looking at the web site entry in IIS services. If it says *(stopped)* then it needs to be started. You can start the server from that line using the right-click menu.

## Checking that the Web Server is Running



The webserver may or may not start automatically when the computer starts, depending on how you have the Service configured. For IIS, this is configured in **Start->Control Panel->Administration Tools->Services**.

## How do I Configure an Apache Web Server so eDeveloper Will Receive Requests?

Here are the steps you need to do to set up an Apache server with eDeveloper.

1. The Apache Web server must be installed, before eDeveloper is installed.
2. The Internet Requesters for eDeveloper need to be installed.
3. Configure the Apache directories
4. After the installation, check the installation and make sure the server is running.

Let's look at each of these steps one by one.

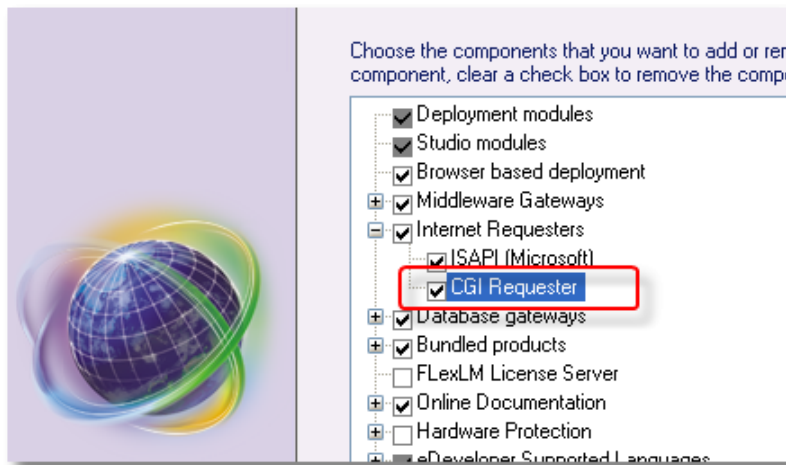
### 1. Install the Apache Server

Download and install the Apache server. You can download the version you need for your computer from <http://www.apache.org>. On the left, under "Apache Projects", choose "HTTP Server". The site has some installation tips, including how to deal with Windows security issues.

It comes packaged with its own installer; just follow the prompts. For a development machine, just choose "localhost" ("this PC") for your Network domain and Server name.

After Apache installs, you will see the little "feather" Apache icon in your system tray. You can click on this to manage and monitor the Apache service. To test the Apache installation, open your browser and go to the address <http://localhost>.

### 2. Make sure the eDeveloper requesters are installed.



Now, install eDeveloper, or modify the eDeveloper installation, so that the CGI requesters are installed.

If you have already installed eDeveloper, you can add the requesters by:

- Go to *Control Panel -> Add or Remove Programs -> Developer -> Change/Remove*.

- Select the *Modify* installation option.
- Add a check before the ISAPI requestor, as shown above.

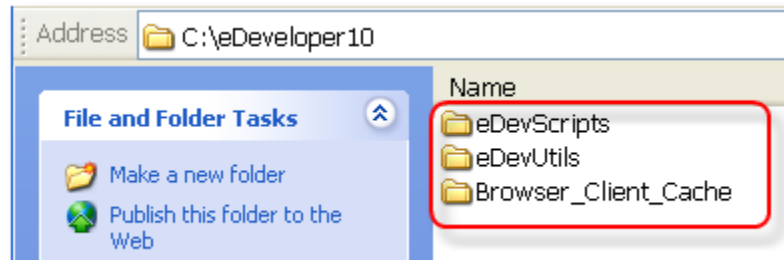
Do NOT uncheck any option: that will cause that option to be un-installed.

### 3. Configure the Apache directories

Next, you need to set up the Apache directories.

- a. First, you need to create directories for the scripts.
- b. Then, you need to point the Apache configuration file to point to those directories.
- c. Last, the Magic.ini needs to point to these directories

#### *a. Create directories for the scripts*



Next you need to create three directories, one for scripts, one for utilities, and one for cache. In our example, we have created these three in a directory called *C:\eDeveloper10*. We will be using these directories in our examples that follow.

#### **The eDevScripts directory**

Into this directory, copy the following eDeveloper files:

- mgrqgnrc101.dll
- mgrqhttp101.dll
- mgrqcgi101.exe
- MGREQ.INI

#### **The Utils Directory**

Into this directory, copy the following eDeveloper files from the eDeveloper Scripts directory:

- MGBC\*.cab
- MGBC\*.js
- \*.jpg
- \*.css

### *b. Apache configuration file changes*

The Apache configuration file is called *httpd.conf* and is found in the \conf subdirectory of wherever Apache is installed.

Here you need to create aliases for each of the three directories we created. You can copy the text below, changing the directory names to match your installation.

```
ScriptAlias /eDevScripts/ "C:/eDeveloper10/eDevScripts/"
<Directory "C:/eDeveloper10/eDevScripts">
Options None
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

```
Alias /eDevUtils/ "C:/eDeveloper10/eDevUtils/"
<Directory "C:/eDeveloper10/eDevUtils">
Options None
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

```
Alias /eDevBCCache/ "C:/eDeveloper10/Browser_Client_Cache/"
<Directory "C:/eDeveloper10/Browser_Client_Cache">
Options None
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

### *c. Magic.ini changes*

Change the following entries in the Magic.ini:

```
InternetDispatcherPath =/eDevScripts/mgrqcgi101.exe
WebDocumentAlias =/eDevUtils
CTLCacheFilesPath = C:\eDeveloper10\Browser_Client_Cache
CTLCacheFilesAlias = /eDevBCCache
```

## 4. Check that the Broker is Running

Depending on how you installed eDeveloper, the Broker will either run as a Service, or as an executable. If it runs as a Windows Service, then it can start automatically. If it is installed as an executable, you'll need to start it manually. There is an option on the **Start->eDeveloper 10->Broker** menu to start and stop the Broker.

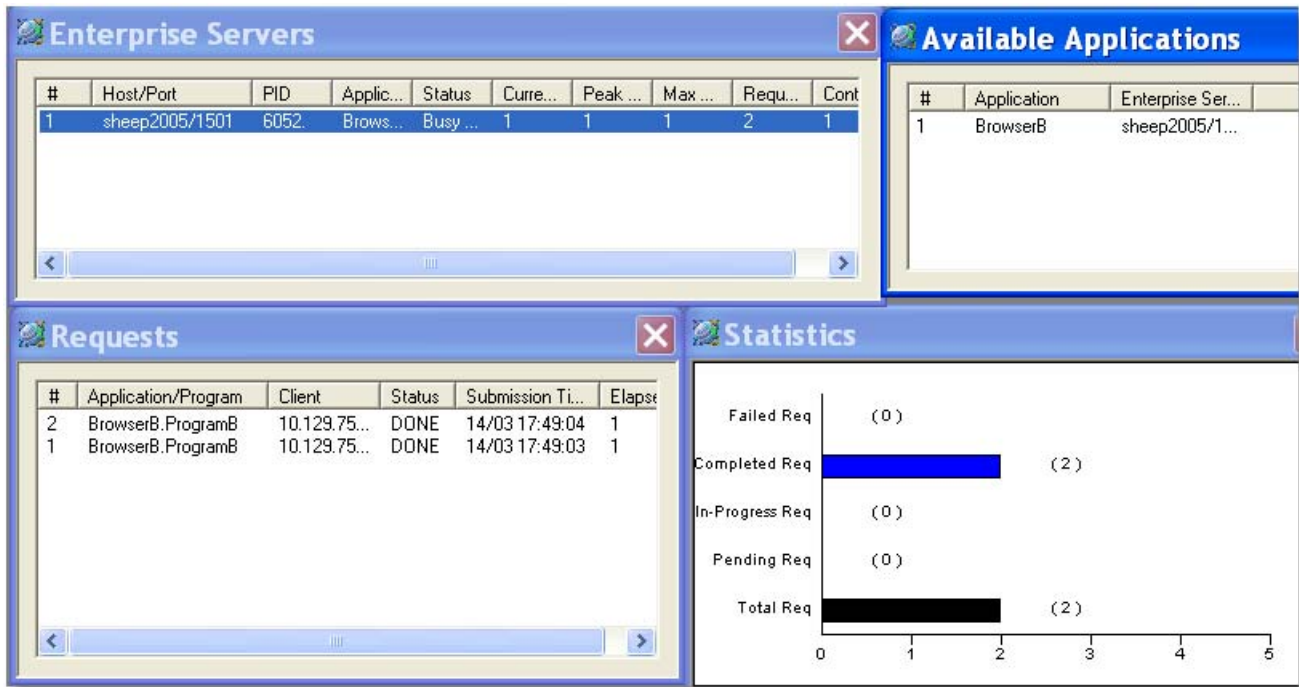
You can ensure the Broker is running, and watch as it runs, by starting the Broker Monitor. You can read more about the Monitor in Chapter 38, “How do I Monitor Broker Activity?” on page 939.



To check if Apache is running, you can use the Apache monitor, or just hover over the icon in the Windows Task bar.

## How do I Monitor Broker Activity?

The Broker



When using the Broker, you may need to get statistics on the broker load. eDeveloper has several tools for monitoring broker activity.

The Broker Monitor, shown above, provides a visual, real-time display of what the Broker is doing.



You can bring up the Broker Monitor by clicking on Broker Monitor from the right-click menu of the Broker icon in the Windows Task bar.

## eDeveloper RQ Functions

The information shown on the Monitor is also available using the built-in eDeveloper RQ functions. For instance, **RqLoad** returns the total number of requests, and the number of requests that are pending, in progress, successfully executed, and that failed execution.

In addition to returning information about the Broker, some of these functions also allow you to control certain Broker operations. For instance, **RqRtBlock** will block requests going to a particular server or service.

Many of these functions require the use of a password -- either a Supervisor password or a Query password -- which is set up in the *mgrb.ini* file (see Chapter 38, "How do I Define the Broker Password?" on page 943).

## Command Line Information

The mgrqcmdl.exe utility allows you to fetch information about the Broker externally to eDeveloper. For instance,

```
mgrqcmdl -query app
```

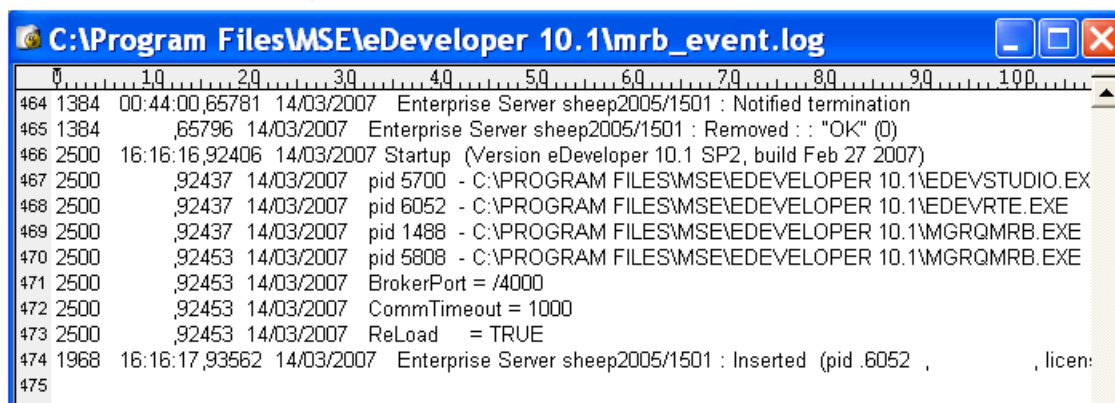
lists all the applications supported by the current Enterprise server.

To get a list of all the options, open a command prompt window and enter the command line option with no parameters:

```
"C:\Program Files\MSE\eDeveloper 10.1\mgrqcmdl"
```



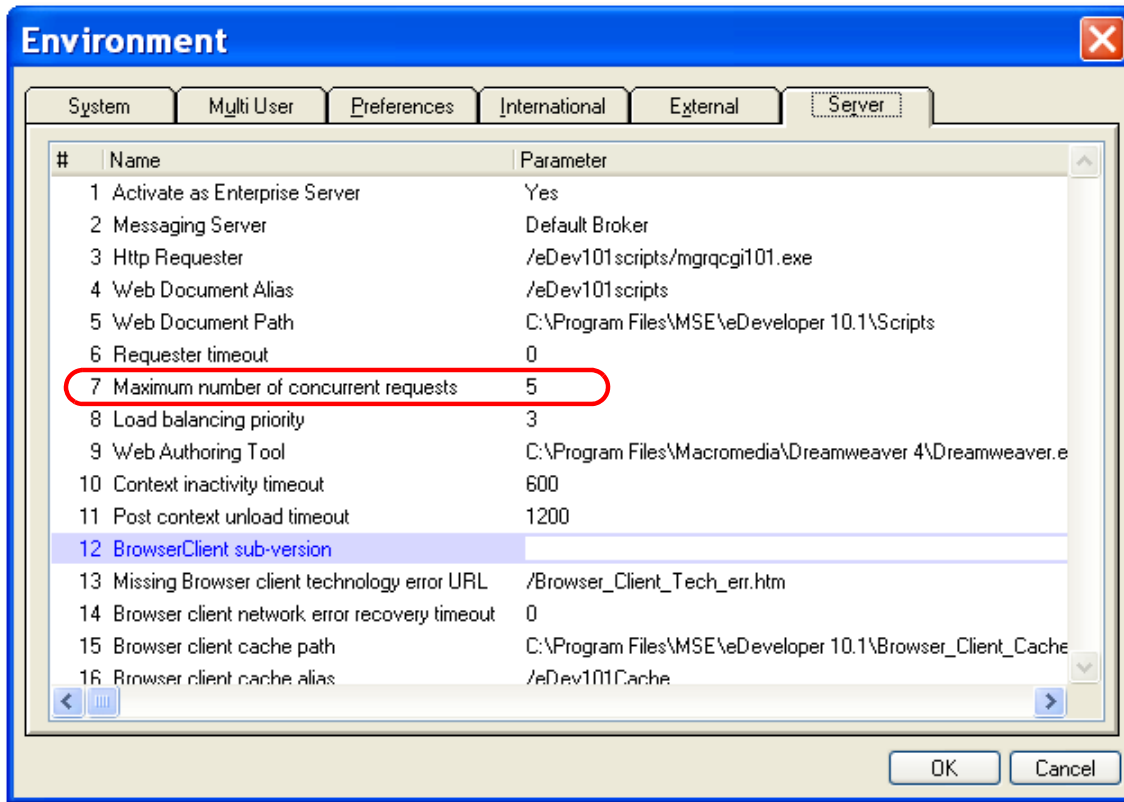
## The Broker Main Log



You can also get information from the *mrb\_event.log*. This log is by default located in the eDeveloper installation directory.



## How do I Limit the Number of Requests That eDeveloper Will Handle Simultaneously?



The number of threads that the Enterprise server can handle at one time is limited according to which license you are using. You can limit this number, however, by setting the Maximum number of concurrent requests parameter in **Settings->Environment->Server**. The number of threads that the engine will use is limited to the minimum between the thread limit (according to the license) and the value that is written here.

A zero entry means that the number of threads will be limited only by the license limit.

## How do I Configure the Broker to Automatically Load an Application?

### [APPLICATIONS\_LIST]

**Online = eDevStudio.exe /DeploymentMode=T,C:\Program Files\MSE\eDeveloper 10.1,,,0,0**

**App1 = eDevRTE.exe /DeploymentMode=B /StartApplication=C:\OrderProc\Orders.ecf,C:\Program Files\MSE\eDeveloper 10.1,,,0,0**

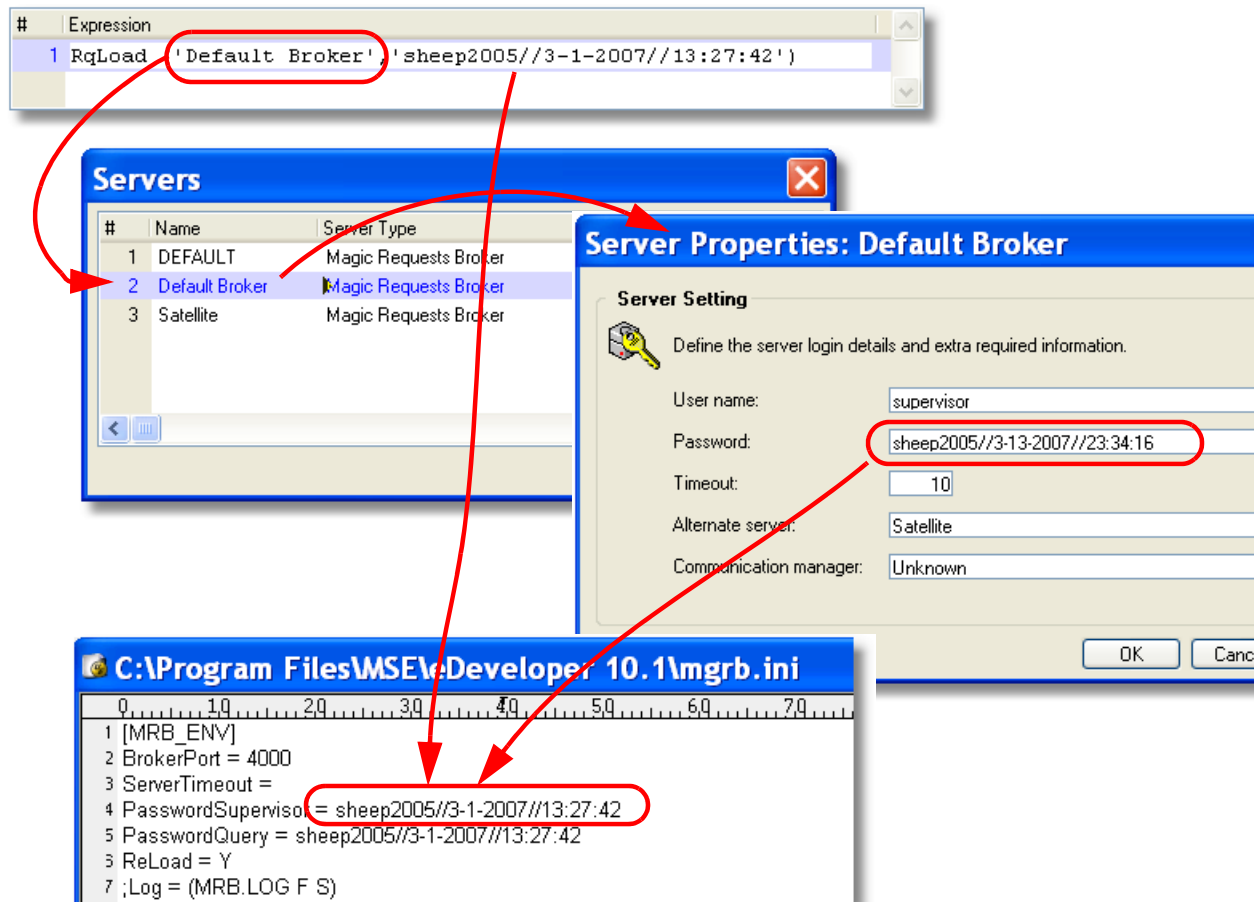
If you want an application to be automatically loaded, you can enter it in the [APPLICATIONS\_LIST] section of the mgrb.ini. The example above loads the studio and a runtime application.

The syntax is:

**EXE\_ENTRY\_NAME=<command>[<work dir>],[<username>], [<password>],[<number of times to perform upon broker initialization>],[<maximum number of engines>]**

If [<number of times to perform upon broker initialization>] is set to 1, then it will load one instance of the declared application.

## How do I Define the Broker Password?



The Broker password is defined in the mgrb.ini. There are actually two settings: the Password for the Supervisor and the Password for Query. The Supervisor password is needed to manage the Broker, while the Query password is used when making queries.

The same Supervisor password must also be entered in the Server Properties in **Options->Settings->Server Properties**.

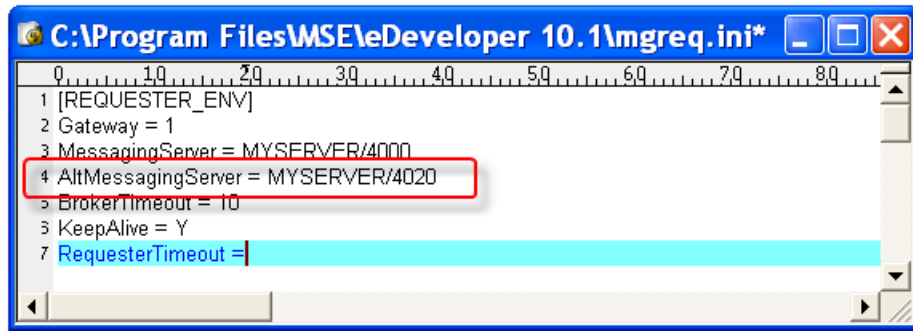
These Supervisor password set automatically during the eDeveloper installation, in the *mgrb.ini* file and in the server properties for the Default Broker.

The Query Password is not set up during the installation. It has only limited application, allowing people to query the engine status and running applications.

## How do I Define an Alternate Broker?

The eDeveloper requester architecture allows you to define an alternate Broker in addition to the main Broker. This way, the requester can use the alternate Broker when the main Broker isn't available.

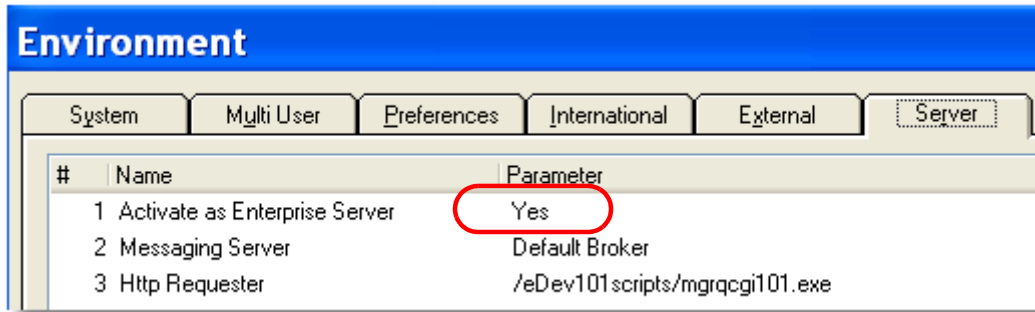
### Defining the Brokers in Mgreg.ini



The requester uses a special .ini file to configure its operation, called *mgreq.ini*.

To add an alternate broker, simply enter the host name/port number the broker is to use on the line *AltMessagingServer*.

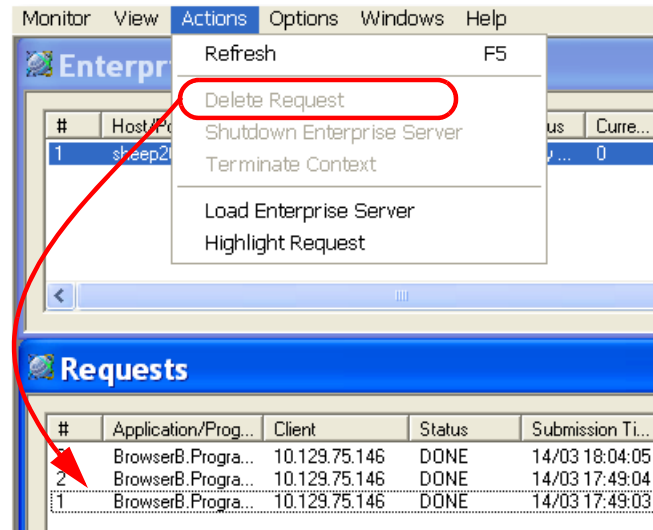
## How do I Disable the Runtime Engine From Serving Broker Requests?



To completely disable serving requests you should instruct the server not to connect to a Broker at startup by setting **Options->Settings->Environment->Server-> Activate as Enterprise Server** to *No*.

However, you can temporarily stop forwarding requests from the Broker to the Server by using the functions **RqRtBlock** and **RqRtResume**.

## How do I Remove a Request Waiting in the Queue?



If a request is still Pending in the Request queue, you can delete it from the Broker Monitor.

- Select the request in the Request panel.
- Choose **Actions->Delete Requests**.

Then, the request will be deleted.

---

## How do I Implement Load Balancing

For simple load balancing you should have at least two eDeveloper Enterprise Servers connected to the same Broker with the same application. The Broker will balance the load between the eDeveloper servers automatically.

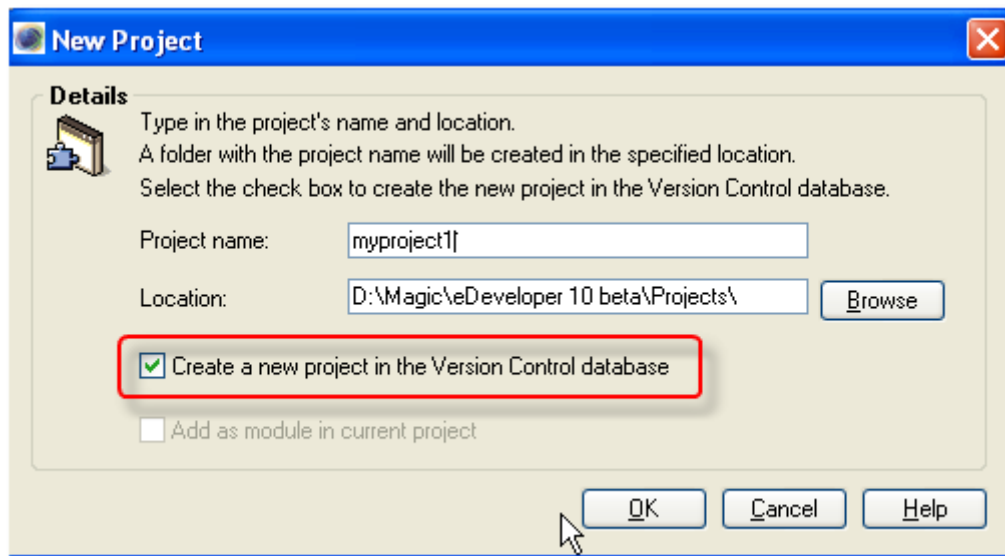
You can implement more complex load balancing by starting several different eDeveloper servers on several different machines. In addition, you can give each server a different priority level by setting the **Options->Settings->Environment->Server->Load Balancing Priority**.





# Chapter 39: Source Management

## How do I Create a Project to be Managed by Version Control?

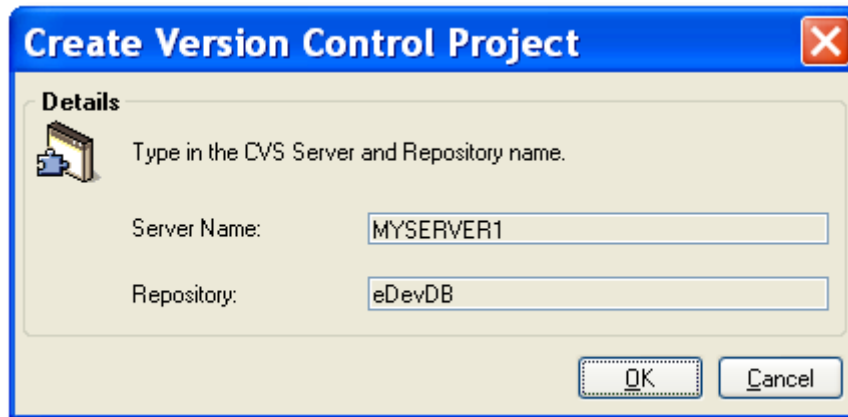


When you are creating a new project in eDeveloper, you can choose to have it managed from the outset by Version Control. Here is how to do it.

**Prerequisite:** You must have a Version Control database set up. See Chapter 39, “How do I Determine the Version Control Provider?” on page 965.

1. Start creating a new project as you normally would, by selecting *File->New Project*, or clicking on the New Project button on the welcome screen. The New Project dialog will appear.

2. Fill out the Project name and Location as you normally would, then check the *Create a new project in the Version Control database* checkbox.

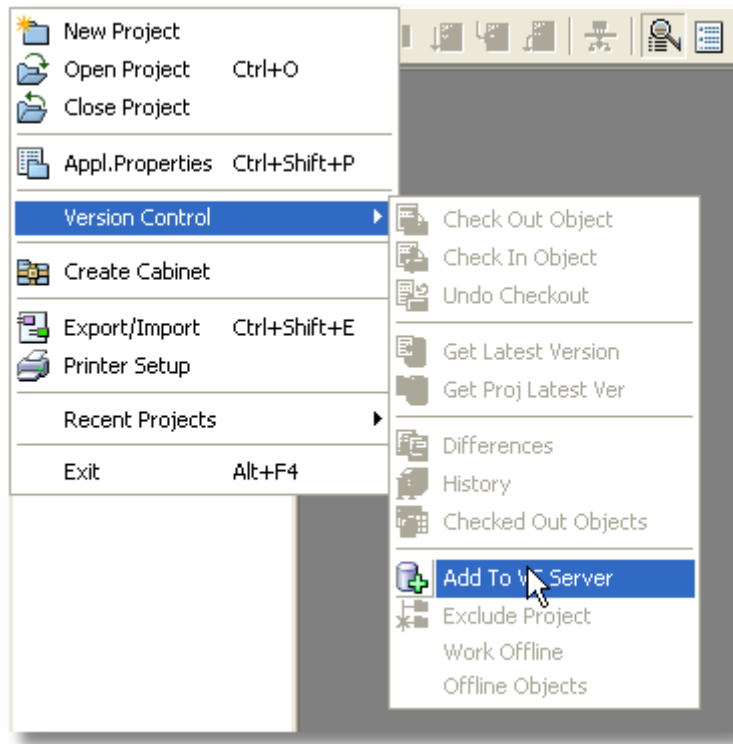


3. You will then be prompted for the server and VC database to use. If you chose to install eDeveloper with the CVS version control server, then CVS will have created a repository called (by default) eDevDB. But you might have another Repository set up, or you might be using another Version Control server.
4. Press OK. You will see a “processing” screen while eDeveloper checks the project into the database.

Now, your new project will be managed by Version Control.

For more about how this works with CVS, see Chapter 39, “How do I Work with CVS?” on page 967.

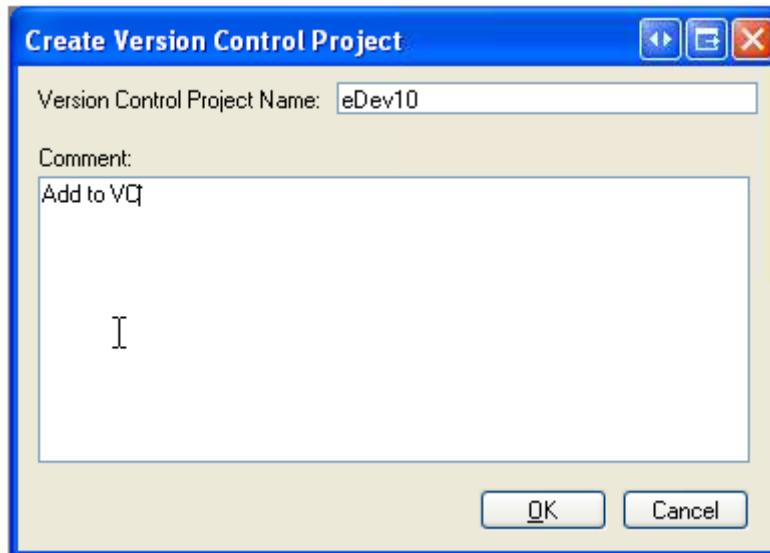
## How do I Add an Existing Project to a Version Control Database?

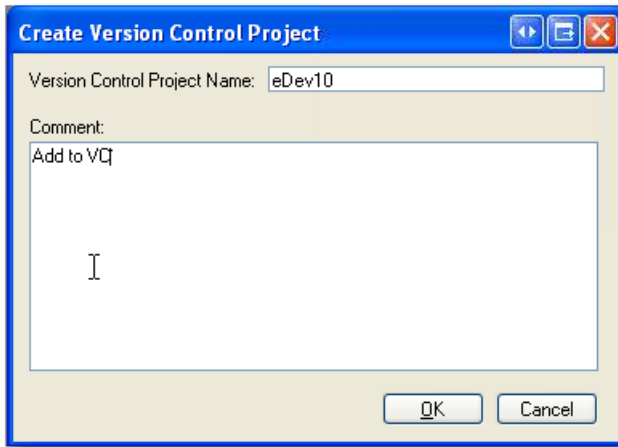


If you have a project already under development, you can check it into the VC Server if you like.

**Prerequisite:** You must have a Version Control database set up. See Chapter 39, “How do I Determine the Version Control Provider?” on page 965.

1. Open the project you want to put under Version Control.





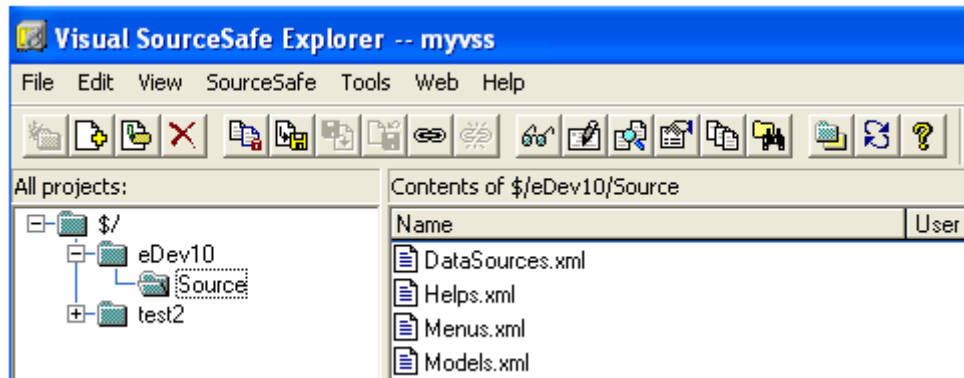
When using Source Safe, the screen looks like this. You can add a comment when you create the project.



When you are using CVS, you have a different screen that allows you to define a Server name and Repository, but does not have a comment.

2. A Create Version Control Project dialog will appear. The screen will be different depending on the Version Control product you are using, as shown above. Enter your data, then Press OK.
3. The process will run for awhile, then your project will be stored in the VC database.

## The project in the VC database

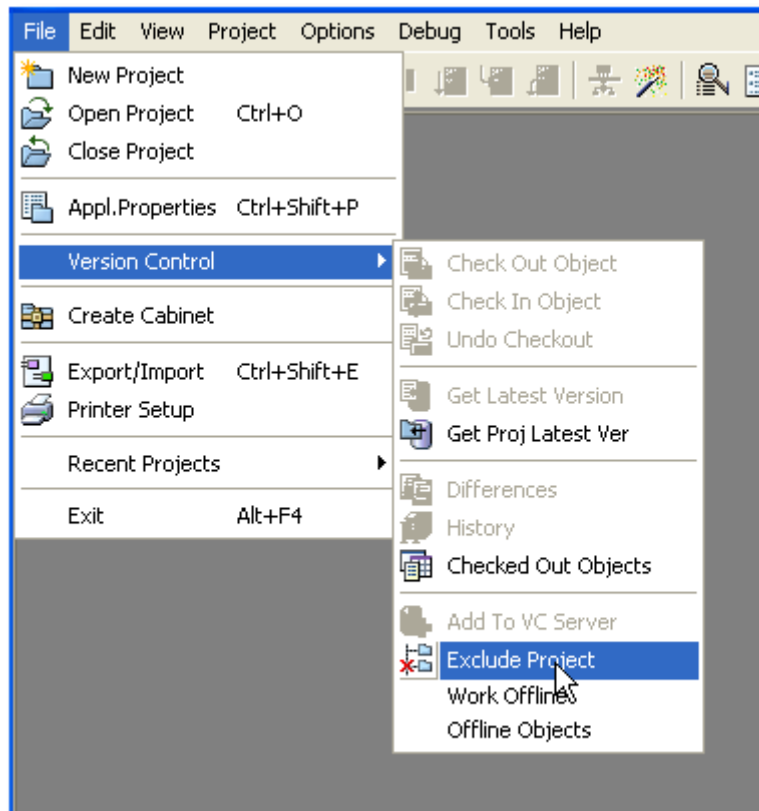


Once the project is stored in the VC database, you can view it within the VC tool. Here you can see our project checked in to Visual SourceSafe.

## Effect of being in a VC database

Once a project is in the VC database, no one can make changes to a program unless they check out the program first. If you try to open a program without checking it out, you will get a warning message and it will be opened in read-only mode.

## How do I Remove a Project from Version Control?



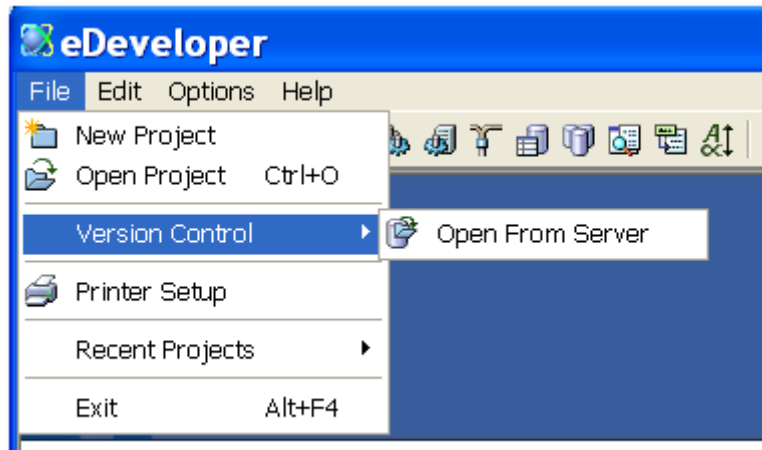
If you decide to no longer use VC Server with a project, you can remove it from the VC Server by selecting **File->Version Control->Exclude Project**.

It should be noted that when you exclude a project here, a copy of it still remains on the VC Server. If you decide to start using Version Control again, you should delete either the copy on the VC Server or the client copy.

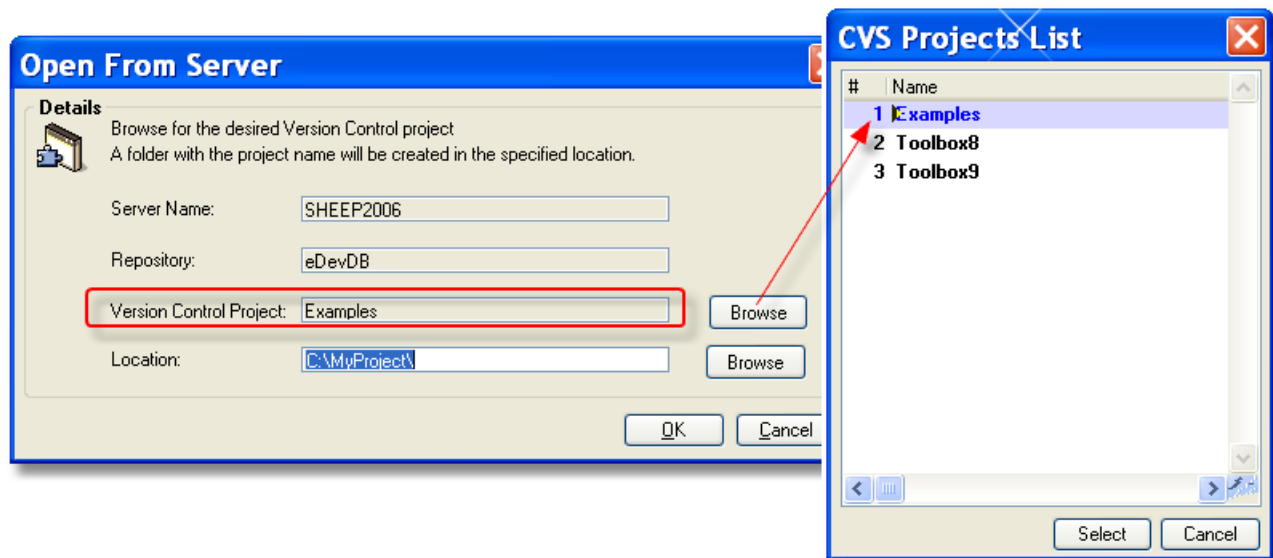
## How do I Add a New Developer to a Project Managed by Version Control?

When a new developer is added to the project, the developer will create a local copy of the project to work on.

1. Close any open project.

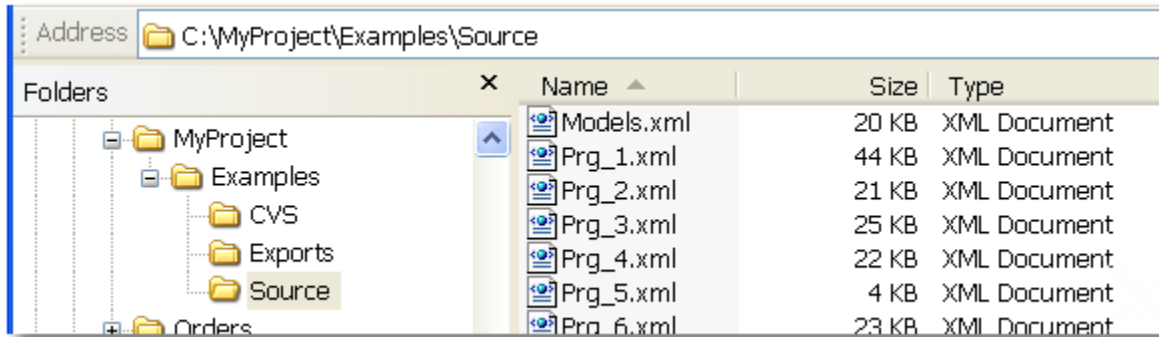


2. Select **File->Version Control->Open From Server**



3. The Open From Server dialog will appear.
4. Click on Browse to select which project to open. We selected “Examples”.
5. A copy of the project will be created in the path specified by “Location:”.

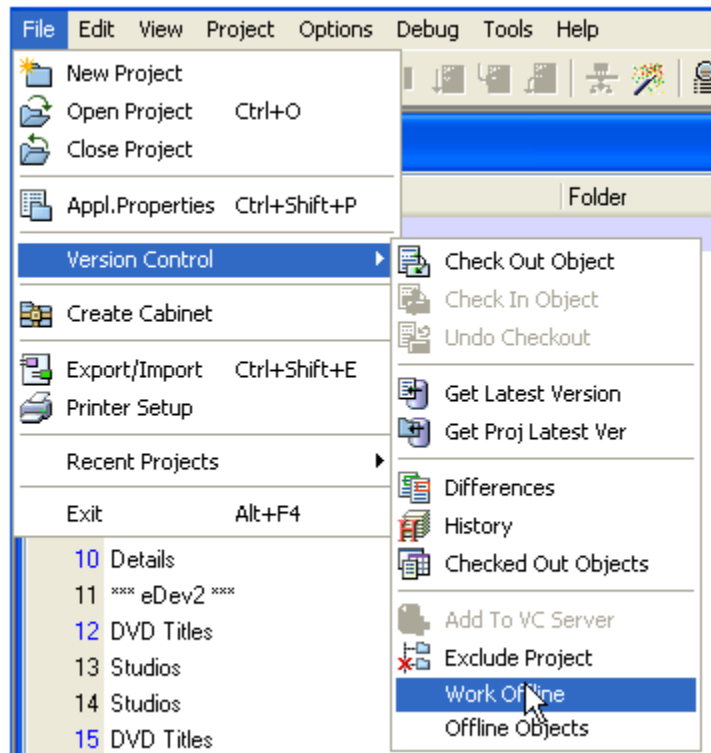




## How do I Develop a Project Managed by Version Control, When the Version Control Server is not Available?

Sometimes, when you are working with Version Control, the VC database may be offline for some reason. In this case, you won't want to stop development. You can still work with VC, by working with it offline.

When you work offline, you can change objects. eDeveloper will track the objects that were changed, and check these changes back in when you reconnect to the VC server.



Once you are in offline mode, the project acts as if it were never in a VC project. You can work with any repository, or any object in a repository.

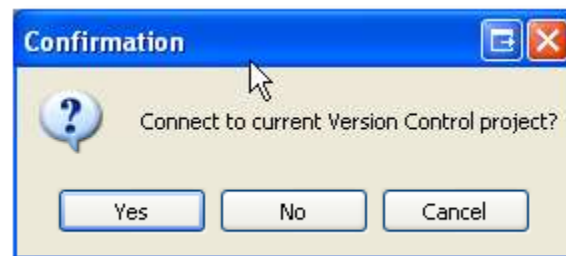
### Which items were changed while I was offline?

If you select *Files->Version Control->Offline Objects*, you will get a list of all the objects that have been changed since you chose to work offline.

Offline Changed Objects				
#	Object name	Checked out by	Conflict	Action
1	Project Modules	Unknown	Unknown	No Action
2	DVD Titles [#5]	Unknown	Unknown	No Action
3	Studios-Selection [#6]	Unknown	Unknown	No Action

In this mode, you don't have to be online to the VC Server to see the changed objects. But, only the Object name appears, since there is no other information available.

## Reconnecting to the VC



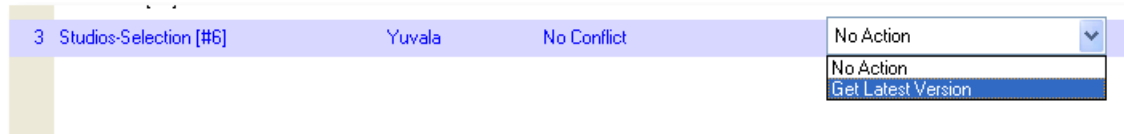
Whenever you are working in Offline mode, and start a new eDeveloper Studio session, eDeveloper will check to see if the VC server is available. If it is, you will get a confirmation dialog, to reconnect or not.

Offline Changed Objects				
#	Object name	Checked out by	Conflict	Action
1	Project Modules		No Conflict	No Action
2	DVD Titles [#5]		No Conflict	No Action
3	Studios-Selection [#6]	Yuvala	No Conflict	No Action Get Latest Version Check Out & Keep Modifications Check Out & Get Latest Version

If you say Yes, then you will again see the list of changed objects. This time though, you can choose what action to take with each of the objects you changed.

1. The **Object name** column is the eDeveloper Name. It also shows the sequence # in parenthesis.

2. The **Checked out by** column shows who checked it out.
3. The **Conflict** column indicates if there is a conflict. A conflict exists if two different users made different changes to the same object.
4. Finally, the **Action** column allows you to decide how to resolve the conflict, if any.



If you are dealing with an item that was already checked out, there are only two options: to do nothing (allow your changes to stand) or to override your changes with the latest version.

## How do I Track Changes?

Every time a programmer checks out or checks in a program, the VC tool keeps track of what changes were made. The programmer can (and should) write a short note about why those changes were made.

Now, when you want to see what changes were made, and when, and why, you can go to *File->Version Control->History*. The result will depend on which VC you are using. We'll show you some results in Source Safe, then in CVS.

## Checking out Models and Data Sources

You can't check out Models and Data Sources individually. When you select **Edit->Version Control->Check out Object** in these Repositories, you will check out the entire Repository.

## Checking Programs in and out

When you want to work on an object in a repository, you will use the command *File->Version Control->Check out object*. You will be prompted to enter a comment, then the item will be open for editing.

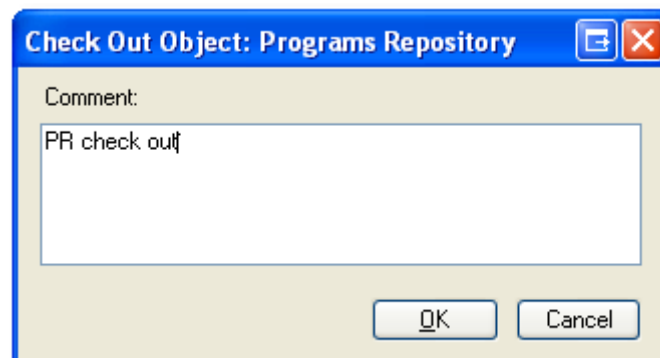
When you are finished, you will use the command *File->Version Control->Check in object*. Again, you'll be prompted for a comment.

These comments will show on the change history for the object.

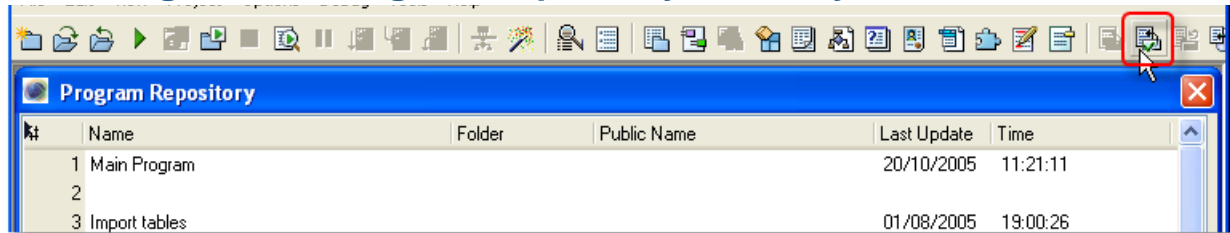
## Checking out the Program Repository Automatically

For some changes, such as adding or deleting an object, the entire repository needs to be checked out.

When the programmer tries to do one of these actions, eDeveloper will automatically attempt to check out the Repository. The programmer will be prompted for a comment, and the Repository will be checked out to that programmer.



## Checking out the Program Repository Manually



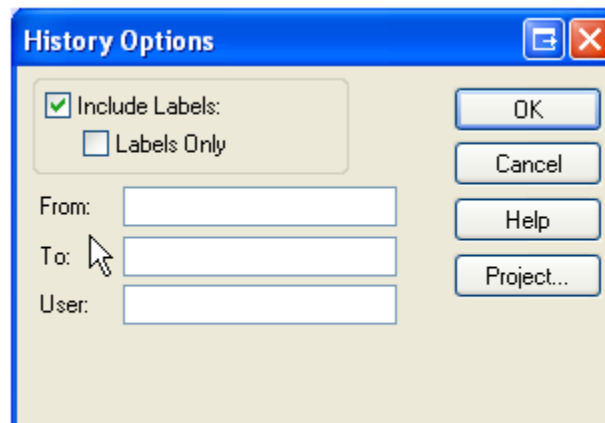
To click out the Program Repository manually:

1. Position the cursor above the top line, on the header line.
2. Click on the "Check out Repository" Icon.

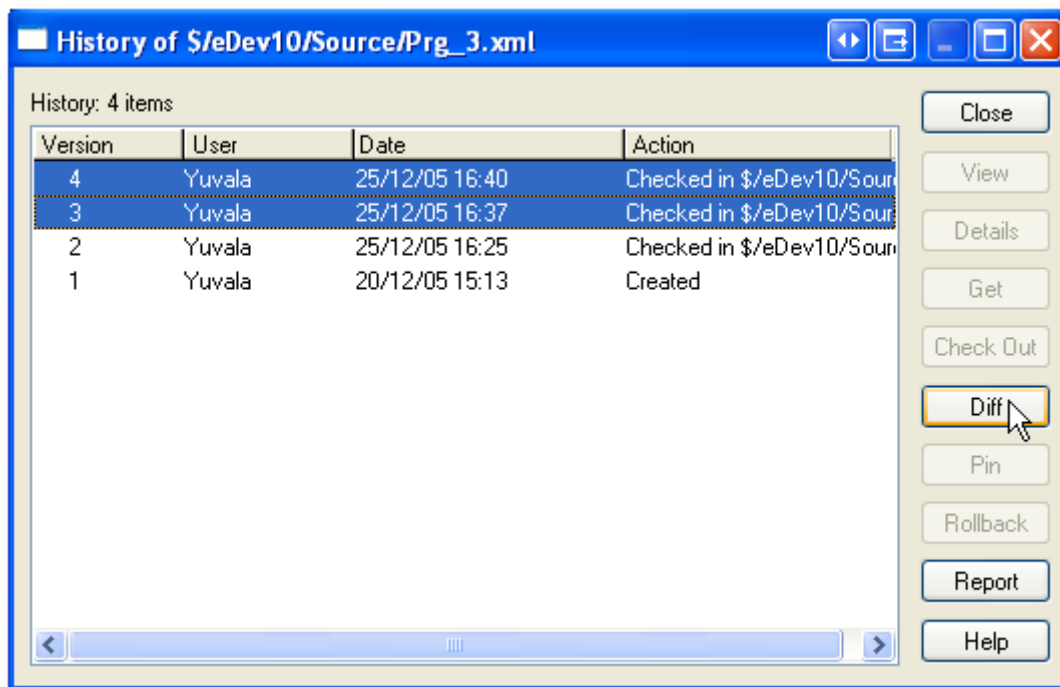
If you position on line zero (the line above the first line), you can check the repository in or out manually by clicking on the icon to the right.

## History in Source Safe

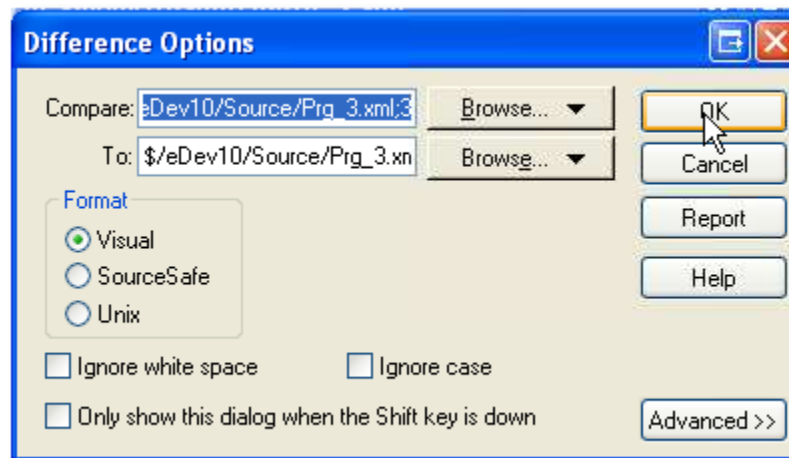
When you want to see what changes were made, and when, and why, you can go to *File->Version Control->History*.



The History Options dialog allows you to enter some filtering criteria, so you can display only the History you want to display. Fill in the options you want, and press OK.



Now you will see a list of all the changes. You can select two of the changes, then choose **Diff**.

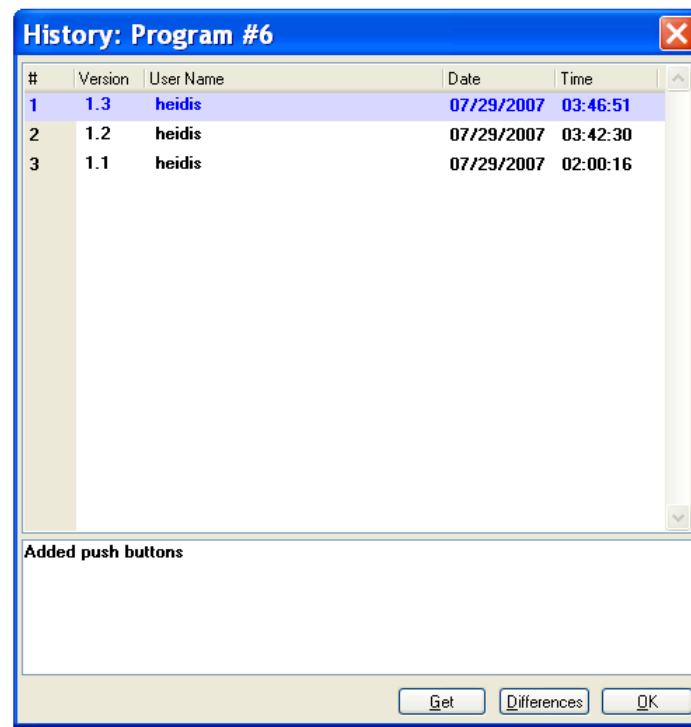


This dialog allows you to choose how to compare the versions. Make your selections, then press OK.

<Data FieldID="2"/>	281	<Data FieldID="2"/>
<ControlName val="Name"	282	<ControlName val="Name"
<MustInput val="Y"/>	283	<MustInput val="Y"/>
<Modifiable/	284	<Modifiable val="Y"/>
<Style val="1"/>	285	<Style val="1"/>
< Orientation/>	286	< Orientation/>

You'll see the two files you were comparing, side by side. Lines that have been changed, or added, or deleted, will be color-coded.

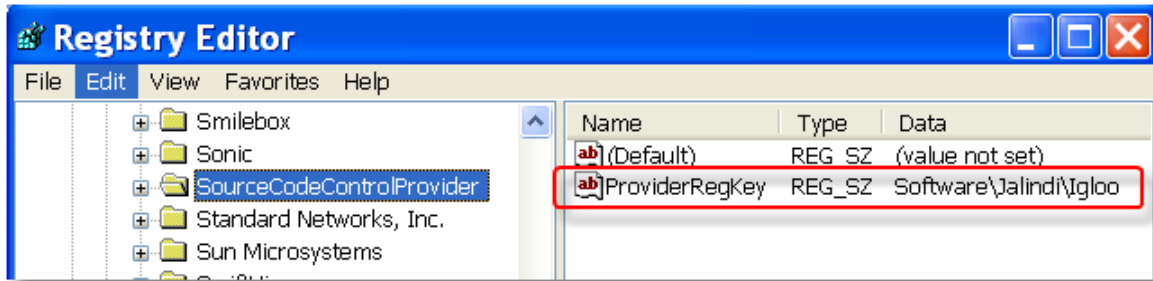
## History in CVS



If you are using CVS, selecting *File->Version Control->History* immediately brings up the history for the object being parked on. If the cursor is on the header line, then it brings up the check-in check-out history for the repository itself.



## How do I Determine the Version Control Provider?



The Version Control Provider that is currently being used by a particular machine can be found in the Registry.

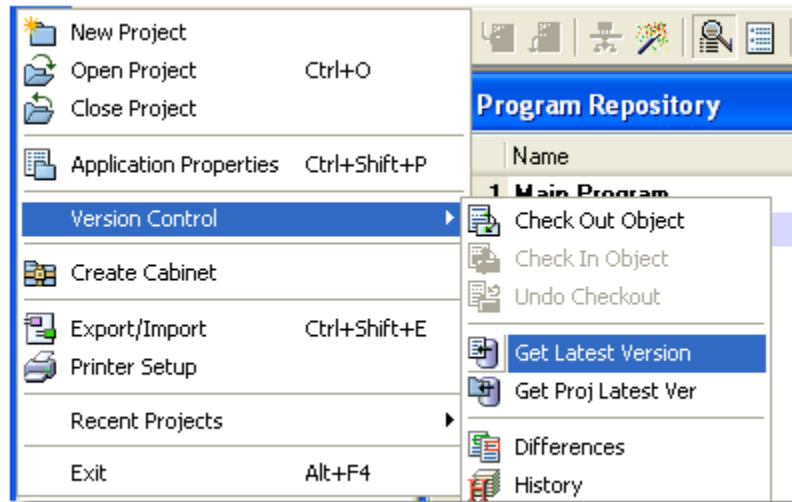
1. Go to Start->Run.
2. Type in: `regedit`. Then press OK. The Registry editor will open.
3. Go to **HKEY\_LOCAL\_MACHINE-> SOFTWARE->SourceCodeControlProvider->ProviderRegKey**

This will show the Source Code Control Provider that will be used by eDeveloper. Other Version Control systems might be installed, but only one can be in use at a time.

You can also see a list of all the providers at:

**HKEY\_LOCAL\_MACHINE-> SOFTWARE->SourceCodeControlProvider->InstalledSCCProviders**

## How do I Rollback to a Prior Copy of an Object?



“Rolling back” in this context means going back to a previous version of an object, or to a previous version of the entire project.

To get the latest version of one object:

1. Position on that object, and select **File->Version Control->Get Latest Version**.
2. You will get a confirmation dialog: Press OK.
3. Now you will have the previously checked in version of that object.

## Rolling back the entire project

To roll back the entire project:

1. Select **File->Version Control -> Get proj latest version**
2. You will be prompted to approve each object that is replaced.
3. When the process is finished, you will get a list of all the objects that were replaced.

## How do I Work with CVS?

CVS is an open-source source-control tool, that comes bundled with eDeveloper. There are a few key pieces of it that you might want to be familiar with.

### CVS Control Panel

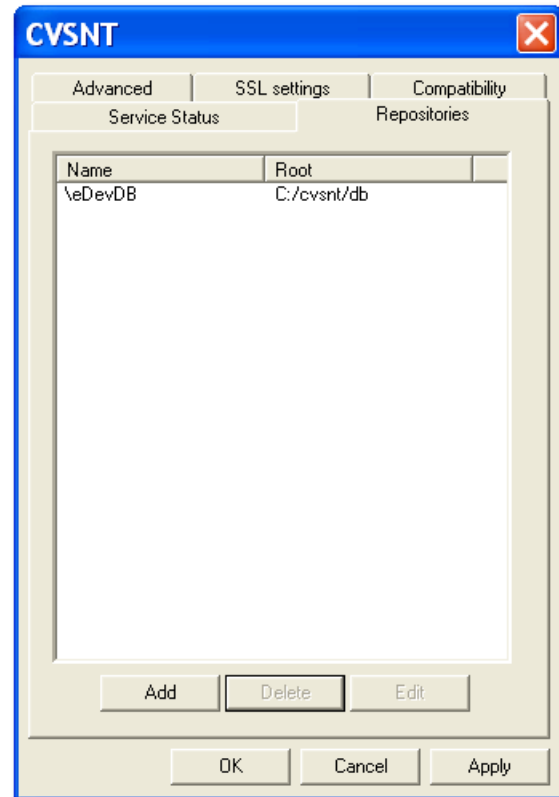
First, CVS has a Control Panel that you will use to set it up. You can access this from **Start->Control Panel**, or from **Start->CVSNT->Service Control Panel**.

From here, you can start or stop CVS, and make changes to the installation. If you make changes, be sure to stop it and restart it before continuing.

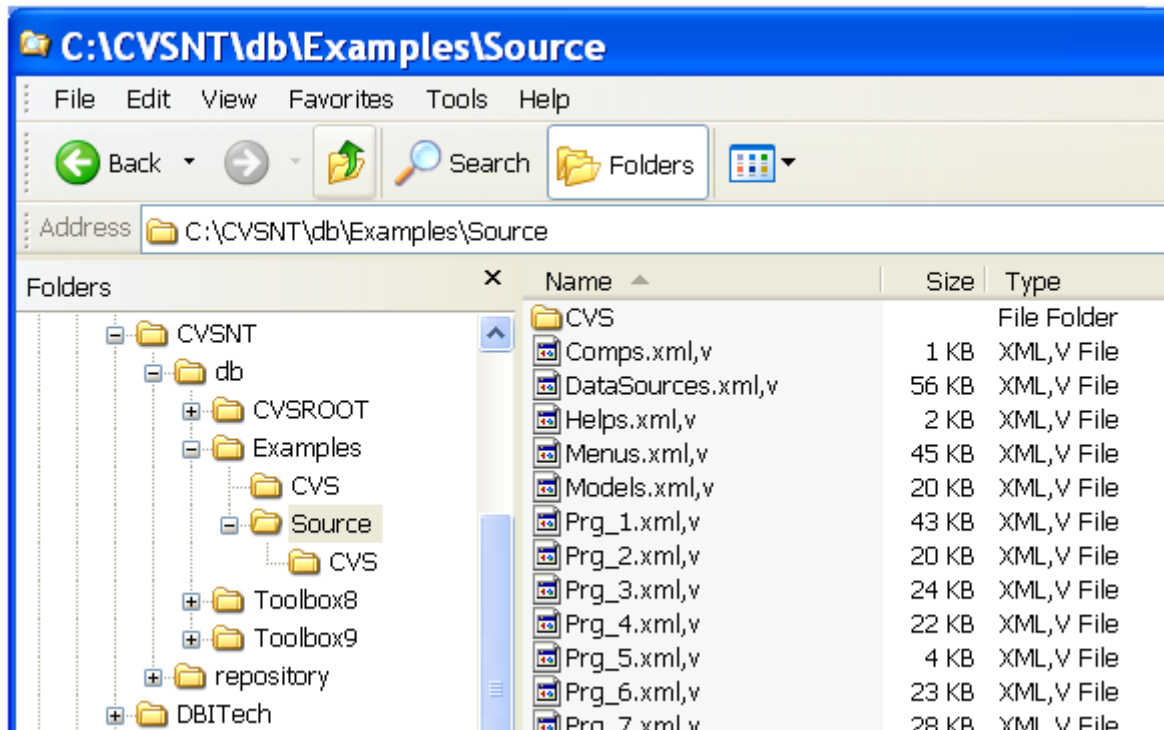
You want to pay particular interest to the Repositories setting. Here, “eDevDB” is the Repository name eDeveloper uses by default, and you should see it already entered after eDeveloper installs CVS.

CVSNT requires the “\” in front of the name.

On the right you will see the “Root”. This is where the CVS data will actually be stored, so it has to be a valid file system location. Here, we are storing our data in “C:/cvsnt/db”.

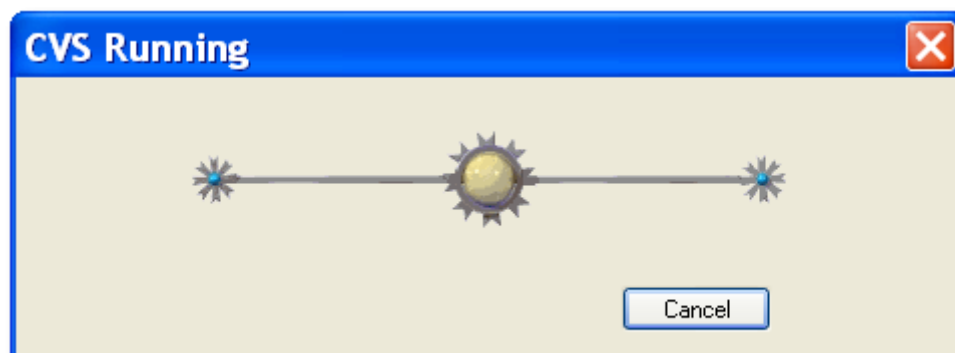


## File Location



Now, when a Project is checked in to CVS, you will see the project's XML files listed in CVS. Here, you can see the "Examples\Source" XML files listed under "CVSNT/db". These files are read-only, but you can look at them. They are the same XML files that are listed in our Examples\Source directory, plus some other data that CVS uses to track them.

## CVS Running



When CVS is running, it has its own visual, as shown above. Some processes can take a fair amount of time to complete, so just be patient.

