

**The EDM™**

**Administrator's Guide**

# Legal Information

## Trademarks

Enterprise Desktop Manager (EDM), EDM Administrator, EDM Agent, EDM Client, EDM Manager, EDM Stager, and EDM Packager are trademarks of Novadigm, Inc. Other brand names and product names are trademarks or registered trademarks of their respective companies.

No investigation has been made of common-law trademark rights in any word. Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of completely fictitious scenarios, data, or other information, and are intended solely to document the use of Novadigm, Inc. software products.

The Novadigm Enterprise Desktop Manager (EDM) enables you to manage and distribute software throughout your organization. It does not provide you with the right to make copies of software that you have obtained from third parties. You should make sure that you have obtained the right to make and use the number of copies of each software program which you distribute using EDM.

## Confidentiality Statement

These materials contain the confidential, proprietary information of Novadigm, Inc., and are for the sole use of the party to which they are provided and solely for use with the Novadigm software with which they are provided, and as may be expressly agreed upon between that party and Novadigm, Inc. Any other dissemination, distribution, copying or use of the information disclosed hereunder is strictly prohibited.

## Restricted Rights

The software and accompanying documentation are provided with "Restricted Rights." Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph © (1) (ii) of the Rights in Technical Data and Computer Software clause at DFAR §252.227-7013; in subparagraphs © (1) and (2) of FAR §52.227-19, Commercial Computer Software-Restricted Rights; or FAR §52.227.14, Rights in General Data, Alternative III, as applicable. Contractor/Manufacturer is Novadigm, Inc., One International Blvd., Suite 200, Mahwah, NJ, 07495.

## Notices

The information contained in this document is subject to change without notice, and does not represent a commitment by NOVADIGM, INC. The software described herein is furnished under licensed agreement and/or nondisclosure agreement. NOVADIGM, INC. software can only be used, copied, or transmitted in accordance with the terms of the license agreement.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including but not limited to photocopying, scanning, or information storage and retrieval systems, for any purpose other than for the explicit use by the licensed organization and/or individual person, without the express written permission of NOVADIGM, INC.

Revised and reprinted with permission by Novadigm, Inc. 1998

Printed in the United States of America

# Table of Contents

---

<b>1</b>	<b>Tour of the EDM System Explorer .....</b>	<b>1</b>
	About the EDM System Explorer .....	6
	Getting Started with the EDM System Explorer .....	7
<b>2</b>	<b>Using EDM Packager .....</b>	<b>55</b>
	Making Packages Available .....	56
	EDM Packager vs EDM Publisher .....	57
	Packager Overview .....	58
	User vs Machine .....	59
	Using Packager .....	60
	Packaging an Application .....	63
	Summary .....	70
<b>3</b>	<b>Using EDM Publisher .....</b>	<b>71</b>
	Making Packages Available .....	72
	EDM Packager vs EDM Publisher .....	73
	Publisher Overview .....	74
	User vs Machine .....	75
	Using EDM Publisher .....	76
	Advanced Editing Options .....	82
<b>4</b>	<b>The EDM Client Explorer .....</b>	<b>87</b>
	Platform Availability .....	88
	The EDM Client Explorer At A Glance .....	89
	EDM Client Explorer Object Menu Map Reference .....	131
<b>5</b>	<b>Using the EDM Screen Painter .....</b>	<b>137</b>
	Before You Use the EDM Screen Painter .....	138
	The Screen Editor Screen .....	140
	Screen Creation Tutorial .....	144
	Controlling Screen Sequencing .....	189
	Screen Painter Reference .....	199
	Opening the EDM Screen Painter .....	200
	Opening an EDM Screen Object .....	201
	Opening a New Screen .....	202
	Saving a Screen .....	203
	Closing Without Saving Changes .....	204
	Changing the Screen Title .....	205
	Inserting Static Text .....	206
	Changing Fonts .....	208
	Inserting an Edit Box .....	209
	Inserting a Push Button .....	211
	Inserting a Check Box .....	214
	Inserting a Radio Button .....	216
	Inserting a Group Box .....	218
	Inserting a List Box .....	219
	Inserting a List button .....	221

Inserting a Bitmap Image .....	223
Selecting Controls .....	224
Horizontally Aligning Controls .....	225
Vertically Aligning Controls .....	228
Making Controls the Same Height .....	231
Making Controls the Same Width .....	233
Testing a Screen .....	235
Displaying Variable Names Without Resolving Objects .....	236
EDM Screen Painter Variable Reference .....	237
Internal Variables .....	238
Troubleshooting .....	240

## **6 Using EDM Extended Batch Commands..... 241**

The EDM Extended Batch Commands at a Glance .....	243
EDM Extended Batch Quick Reference .....	255
EDM Extended Batch Command Reference .....	258
ASK .....	259
CALL .....	260
CHDIR or CD .....	261
COPY .....	262
COPY .....	263
DELAY .....	264
DEL_VAR .....	265
DESTROY_IND .....	266
DESTROY_USER_WINDOW .....	267
DISABLE_MENU_EXIT .....	268
DISPLAY_USER_WINDOW .....	269
DROP_OBJECT .....	270
ECHO .....	271
EDM_ADD_INST .....	272
EDM_COPY .....	273
EDM_DEL_INST .....	274
EDM_DELETE .....	275
EDM_ERASE .....	276
EDM_MOVE .....	277
EDM_SET_INST .....	278
ENABLE_MENU_EXIT .....	279
ERASE or DEL .....	280
EXIST .....	281
MKDIR or MD .....	282
MOVE .....	283
RELEASE .....	284
REM .....	285
RENAME .....	286
RENAME or REN .....	287
RMDIR or RD .....	288
SET_ABORT_TITLE .....	289
SET_ABORT_WINDOW .....	290
SET_CANCEL_TITLE .....	292
SET_CANCEL_WINDOW .....	293
SET_CONTINUE_TITLE .....	295
SET_CONTINUE_WINDOW .....	296
SET_DECRYPT_ON .....	298

SET_DECRYPT_OFF.....	299
SET_ECHO_TITLE.....	300
SET_ECHO_WINDOW.....	301
SET_FRONT_WINDOW.....	303
SET_IND_TITLE.....	304
SET_PROCESS_NAME.....	305
SET_USER_BACKGROUND.....	306
SET_USER_FOREGROUND.....	307
SET_USER_TITLE.....	308
SET_USER_WINDOW.....	309
SET_VAR.....	311
SETIND.....	312
SINGLE_STEP_ON.....	313
SINGLE_STEP_OFF.....	314
STARTIND.....	315
TRAP.....	317
WRITE_USER_WINDOW.....	318
Using Installed EDM Extended Batch Programs.....	319

## **7 The EDM Dialog Command Reference..... 321**

Platform Availability.....	322
EDM Dialog Command Facility at a Glance.....	323
Understanding Dialog Syntax.....	324
Dialog Variables.....	325
EDM Dialog Command Reference.....	331
BREAK.....	332
CALC.....	333
CLOCK.....	334
DELAY.....	336
DEVICE.....	337
DLGBOX.....	339
ELSE.....	341
END.....	342
ENDIF.....	343
EXECUTE EXECUTE(p).....	344
FLICKER.....	346
GOSUB.....	347
GOTO.....	348
HOST.....	349
IF.....	350
INPUT.....	355
KEYDELAY.....	356
LINE.....	357
MANUAL.....	359
NOBREAK.....	360
OFF.....	361
OIA.....	362
ON BREAK GOTO.....	363
ON EVENT GOTO.....	364
ON PROG GOTO.....	367
ON RUNOUT GOTO.....	369
ON TIMEOUT GOTO.....	370
OPTION.....	371
PAUSE.....	372

PRINT .....	373
PROMPT.....	375
QUIT .....	377
REMARK.....	379
RETURN.....	380
RUN RUN(P).....	381
RUNOUT.....	383
SEND .....	385
SET .....	387
TIMEOUT.....	388
WAITFOR.....	389
Continuity Between Calls.....	392
Predefined Variables.....	393
Numeric Expressions.....	395
Send Keywords.....	398
Sample EDM Dialog Files.....	399
<b>A Understanding the Resolution Process .....</b>	<b>401</b>
<b>INDEX.....</b>	<b>413</b>

# Tour of the EDM System Explorer

This chapter introduces you to the EDM System Explorer—the EDM tool you work with to configure and administer your site's desktop environment. During this tour, you will learn the concepts of maintaining a distribution model in the EDM Database, and the actions you take when administering the EDM environment.

## The Distribution Model

Your *distribution model* records the identities of the desktop computers whose configurations are under management by EDM, and their intended configurations.

Your distribution model can be as sophisticated, or as simple, as you want. At a minimum, an EDM distribution model includes the following five elements:

- **Users**  
The identity of the desktops under management.  
(Example: a user's userid.)
- **Applications**  
What software is being managed by EDM?  
(Example: Microsoft Office 95.)
- **Application Files**  
What are the components that make up the application?  
(Example, for Microsoft Office 95 approximately 650 files, DLLs, EXEs, HLPs, ICOs, and 200 registry updates.)
- **Deployment Source**  
Where are the application components stored so they can be deployed to the users?  
(Example: the EDM Database, or a local staging server.)
- **Deployment Destinations**  
Where will the application and its files be deployed on the desktop or LAN (local area network) (Example: C:\MSOFFICE\, C:\WINDOWS\)
- 

The subsequent sections of this chapter explain how to use the EDM System Explorer to configure and maintain distribution models for your applications.

As you gain experience with EDM and become a more advanced user, there are other elements you will probably include in your distribution models. A few of these capabilities include: distribution scheduling, error handling, security, and collecting audit information from the desktops.

### The Distribution Model – Handling your unique requirements

EDM ships with the EDM Database pre-configured with a set of components that can be used for managing desktop computer configurations on an enterprise-wide scale. Using these components, you can build and maintain very sophisticated and complex distribution models.

However, it is important to note that EDM provides a highly customizable framework that you can modify and extend to meet your organization's individual requirements. For example, it is easy to add your own components to the EDM Database, extend EDM-supplied components, and/or integrate your own in-house developed or third-party-supplied executables to extend EDM's base functionality, in managing your distributed environment, or for some other purpose.

In general, EDM documentation describes how to use EDM as it is configured when it is first installed from its installation media.

In particular, this document describes the EDM System Explorer using the EDM Database as it is configured when shipped from Novadigm. If your organization customizes EDM, the System Explorer can be used as herein described, but the sample screens shown in this document will differ from what you will see in your live environment.



For purposes of maintainability, customization of EDM should be thoroughly documented and diligently kept up to date in a central location. You might want to create a project folder that is accessible to all of your enterprise's EDM administrators.

## EDM Terminology

The following terms will be explained in greater detail later, but you should familiarize yourself with these key concepts now.

The EDM Database records your distribution model and information regarding its deployment. The EDM Database is hierarchically structured, and its components consist of Files, Domains, Classes, Instances and Attributes.

Term	Short Description
<p>File</p> <p>Example: Primary File</p>	<p>Highest level in the hierarchy of the EDM Database. Groups like domains together.</p> <p>Used to define and maintain the distribution model. This is one of the five pre-configured files distributed with EDM and installed when you first install EDM. The other four are the Audit, History, Profile and Resource files. EDM administrators will do most of their work in the Primary file.</p>
<p>Domain</p> <p>Example: SYSTEMX Domain</p> <p>Class</p> <p>Example: User Class</p> <p>Class Instance or Instance</p> <p>Example: User Class Instance defining John Doe's computer</p> <p>Attribute</p> <p>Attribute Value</p> <p>Example: User Class Instance defining John Doe's computer, NAME attribute and USERID attribute</p>	<p>Logically partitions an EDM file. Groups like classes together.</p> <p>Contains the classes needed to deploy applications. EDM administrators will do most of their work in the SYSTEMX domain of the Primary file.</p> <p>A category of the distribution model to be managed. The class is a template for the attributes (also called "properties" and/or "fields") needed to create an instance of the class. It is conceptually similar to a schema in a relational database structure, or a file layout in a traditional flat file. Among others, each of the five required elements of a distribution model (Users, Applications, Application Files, Deployment Sources and Deployment Destinations) is defined in the EDM Database by its class.</p> <p>The category that defines users of EDM-managed applications. The User class contains all of the information attributes necessary to identify a desktop (client) computer to be managed by EDM.</p> <p>An object containing a specific occurrence of a class. The attributes of a Class Instance object contain the data describing one specific entity of that class.</p> <p>An object created from the User class containing the information needed to identify John Doe's EDM-managed computer</p> <p>An attribute is a data element (also called a "field" or "property") of a class. The class contains the definition (e.g. the name, data type, description and length) for each attribute comprising the class. Each class instance created from the class contains the attributes defined in the class. Each attribute of an Instance contains a value.</p> <p>The NAME attribute of the User class contains the name of the user, and the USERID attribute contains the user's user I.D., as specified by the EDM Administrator. In this example, the NAME attribute contains the value "John Doe", and the USERID attribute contains the value "JDOE"</p>

## About the EDM System Explorer

EDM enables you to configure applications, connect users to applications, and define hardware and software auditing requirements for desktop computers throughout your enterprise, based upon the policies that your organization sets for their desired configurations.

The EDM System Explorer is an interactive graphical tool for manipulating and inspecting the content of the EDM Database.

By working in the graphical EDM System Explorer, you can, among other things:

- Define users (i.e. identify desktop computers to be managed by EDM) in the database.
- Assign applications to individual users and workgroups.
- Define applications and their files.
- Define special file properties.
- Group applications based on your organization's policies.
- Grant users access to applications, or to groups of applications.
- Specify the software and hardware asset information you want to discover on the EDM Client desktop.
- Manage the configuration of users' desktops.

## Getting Started with the EDM System Explorer

---

This section describes how to open and operate the EDM System Explorer. Users of Windows 95/NT 4.0 will already be familiar with many of these operations.

### Using a Mouse to Manipulate Elements in the Window

You will probably be using a pointing device such as a mouse to work with objects in the EDM System Explorer. With its familiar graphical user interface, the EDM System Explorer supports standard mouse actions activated by clicking or double-clicking the left mouse button. Also, EDM System Explorer provides context sensitive pop-up menus activated by clicking the right mouse button.

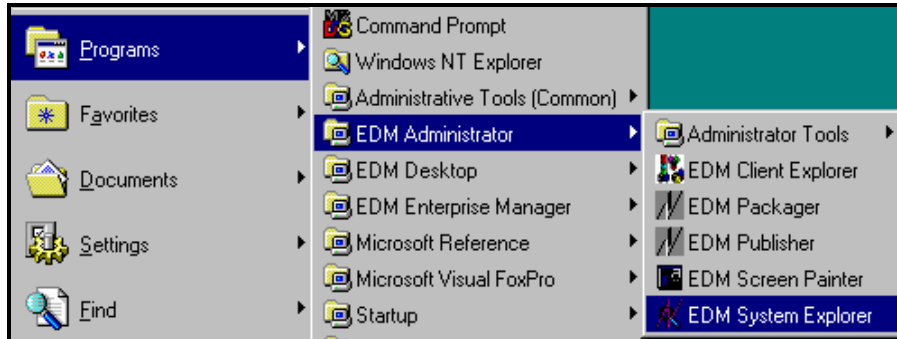
The following table provides a summary of mouse actions you can perform:

Mouse Action	Result
Click the left mouse button (In this chapter we refer to this as a left-click)	Highlight an icon, select a pull down menu command, and press a button.
Double-click the item using the left mouse button (In this chapter we refer to this as choosing an item.)	Open the next level display window or dialog box, depending on the item at which the mouse is pointed.
Click the right mouse button (In this chapter we refer to this as a right-click)	Display a pop-up context menu containing choices appropriate to the item right-clicked.
Point	Move the mouse so that the pointer displayed on your screen rests on the desired item.
Drag and drop by clicking on an object with the left mouse button, holding the mouse button down while moving the mouse pointer to the desired target, then releasing the mouse button	Create a connection between two instances of connectable classes.

When employing EDM System Explorer to configure your enterprise desktop environment, you will use many familiar desktop elements: windows, pull down menus, pop-up menus, and menu commands, dialog boxes, and so forth.

## Opening the EDM System Explorer

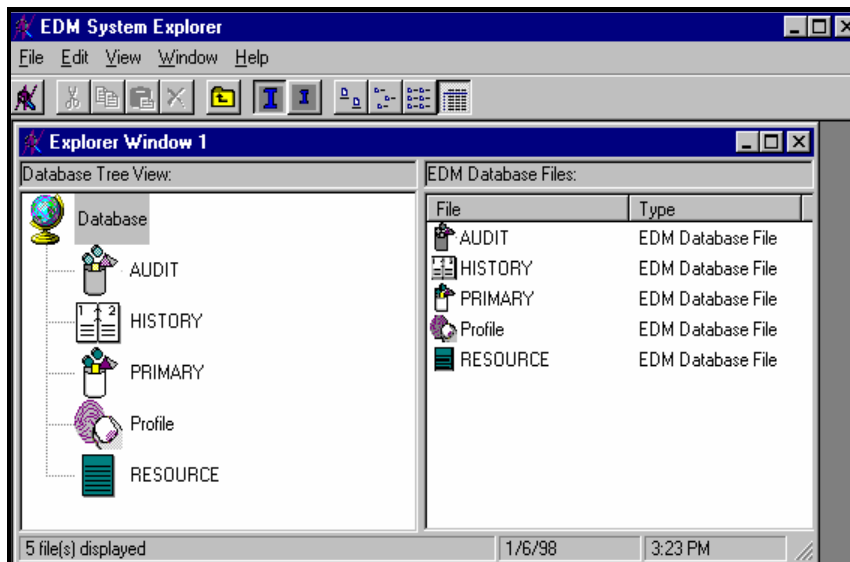
### ➤ To Open the EDM System Explorer:



- 1 From the EDM Administrator Program Group or Menu choose **EDM System Explorer**. You will be presented with a login screen similar to this:



- 1 Type in a functioning User ID and Password, and click on the **OK** button. When first installed, EDM will accept EDM\_MAST for the User ID, with a blank password. The user may change this later.

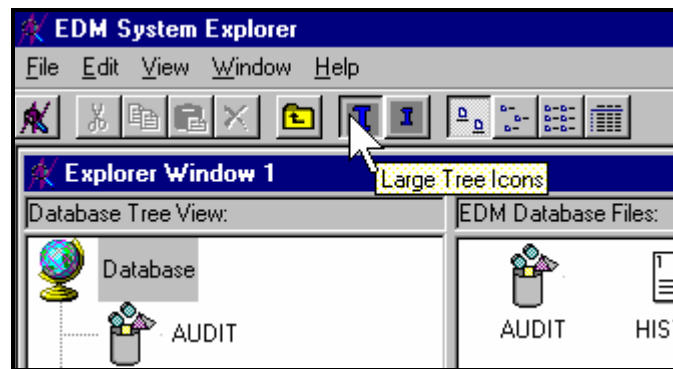


The EDM System Explorer will appear:

The EDM System Explorer has a familiar appearance within a multiple document interface (i.e. you can open multiple Explorer windows simultaneously). Beneath the title bar, there is a pull down menu, a tool bar, and a workspace containing Explorer windows. Each Explorer window contains a hierarchical tree view of the EDM Database on the left, and, on the right, a view of the contents of the EDM Database component currently selected in the tree view. A status bar appears at the bottom of the window.

When you first start up the EDM System Explorer, the tree view displays a top-level view of the EDM Database, listing the five files that comprise the EDM Database as shipped from Novadigm. You can expand and collapse the tree view to show or hide lower level components of the EDM Database, as described in "*Manipulating the Tree View*" in this chapter. With the Database icon selected in the tree view, the right side of the Explorer window displays a list of the files comprising the EDM Database.

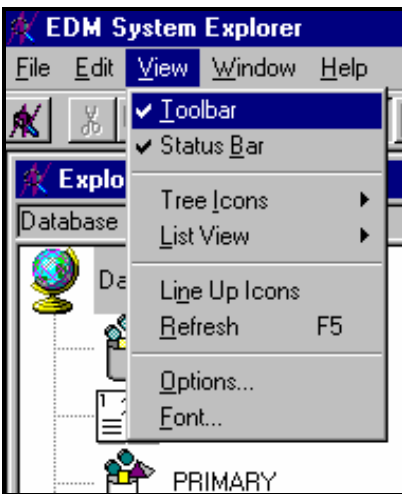
The EDM System Explorer has many user-friendly features that users of the Windows Explorer are accustomed to. For example, a number of items you see in the window provide tool tips. A tool tip is helpful information that pops up spontaneously when the mouse pointer is moved over such an item in the Explorer window, as shown here:





The text describes the function of the item under the mouse pointer. In this case it identifies it as the button to press for **Large Tree Icons**. The text box spontaneously closes when the mouse pointer is moved off the interface element.

## Controlling the Appearance of an Explorer Window





You can manipulate the appearance of the window and its contents by using customary Windows operations and functions on the View menu.



- You can re-size the window by clicking and dragging the window borders or corners.
- You can re-size the relative width of the tree view and the right side of the Explorer window by clicking and dragging the frame between them.
- You can choose whether or not to display the toolbar by clicking the **T**oolbar choice on the **V**iew menu.
- You can choose whether or not to display the status bar by clicking the **S**tatus **B**ar choice on the **V**iew menu.
- You can toggle the display of the tree view between large icons and small icons by clicking the **T**ree **I**cons choice on the **V**iew menu, or by clicking on the appropriate button on the Toolbar:

View	Toolbar button	Usage
Large Icon		Provides easy selection from a small list
Small Icon		Maximizes the number of items that can be seen in the tree at one time, and is useful for selecting from large lists

- You can choose from among four different displays (**Large Icons**, **Small Icons**, **List** and **Details**) for the information in the right side of the Explorer window. Select your choice from the **L**ist **V**iew sub-menu of the **V**iew menu, or by clicking on the appropriate button on the Toolbar.

View	Toolbar button	Usage
Large Icon		Provides easy selection from a small list
Small Icon		Maximizes the number of items that can be seen in the right side of the Explorer window at one time, and is useful for selecting from large lists
List		Maximizes the number of items that can be seen in the right side of the Explorer window at one time, and is useful for selecting from large lists
Details		Provides maximum information about each item displayed. Details view is the best choice for working with instances, because the instance attribute values appear in the right side of the Explorer window, avoiding the need to open a separate dialog in order to view them.

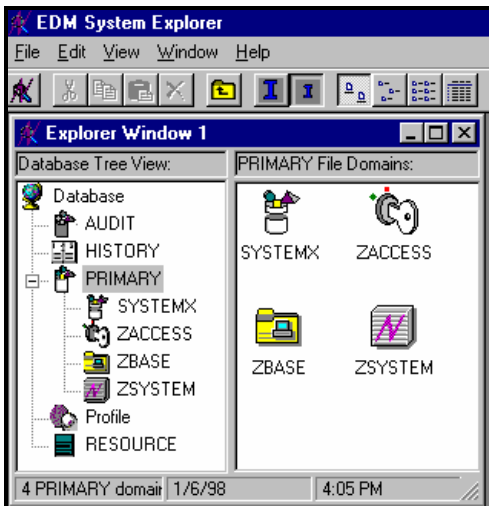
- You can change the font used for displaying text in Explorer window by selecting the **F**ont... choice from the View menu, and selecting a new font from the ensuing dialog.

## Manipulating the Tree View

The EDM Database is hierarchically organized and subdivided into files, domains, classes and instances. The tree view enables you to drill down into the hierarchy, and easily locate and work with specific EDM Database components.

For example, open the EDM System Explorer, and double-click on **Primary** in the tree view.

The window will appear something like this:



The four domains of the Primary file appear as branches beneath the its icon in the tree view, and simultaneously appear as individual icons in the right side of the Explorer window. A small box with a minus sign appears in the tree structure next to the Primary file icon, signifying that the tree is displaying the next level of the EDM Database hierarchy below the Primary file. You can click on the minus sign to close the tree display below that level. When you do so, the tree closes and the minus sign changes to a plus sign.

You can click on the plus sign, and the previous tree display will be restored.



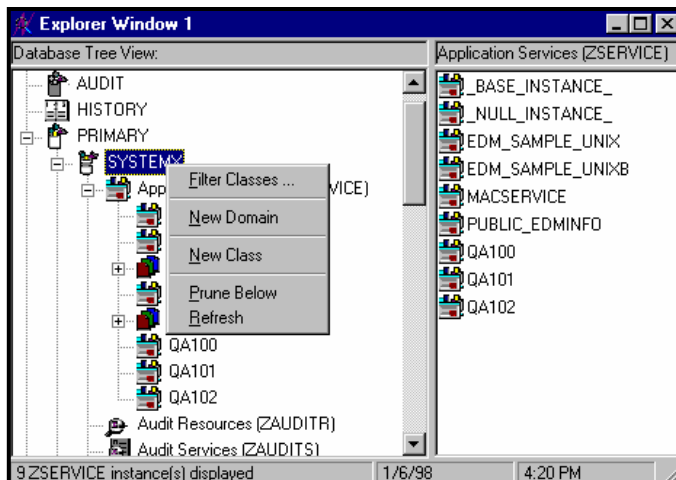
To open an EDM Database component, double-click it. For example, to display a component from the next level down the hierarchy, double-click on the **SYSTEMX** domain's icon in either the tree view or on the right side of the Explorer window. The classes of the SYSTEMX domain will be displayed in the right side of the Explorer window.

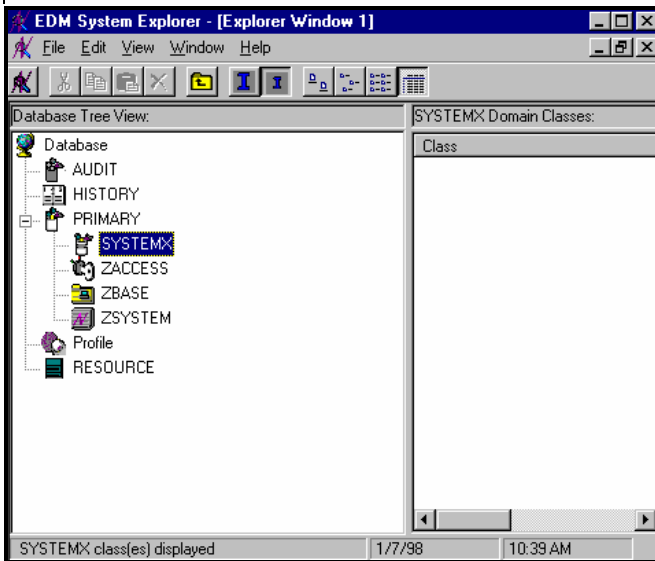
If you double-clicked the **SYSTEMX** domain icon in the tree view, the tree view opened to display the SYSTEMX domain classes below its icon. If you double-clicked the **SYSTEMX** domain icon on the right side of the Explorer window, a small box containing a plus sign appears next to the icon in the tree view, but the classes are not displayed beneath it.

As you work with the EDM System Explorer, opening and closing components, the tree view remembers which branches you've opened and closed, and marks previously opened and closed branches at each non-elementary level with a plus or minus sign in a small box adjacent to the component's icon. Clicking on the plus sign expands the branch of the tree view to the highest level of detail previously opened in the current EDM System Explorer session. Clicking on the minus sign collapses the tree view display below the clicked node.

To discard the history of which branches were previously opened beneath a particular component, use the **Prune Below** function, which can be accessed from the context menu that pops up when you right-click an icon in the tree view.

For example, refer to the SYSTEMX domain, whose icon has been right-clicked, in the following illustration. A context menu similar to the following pops up:



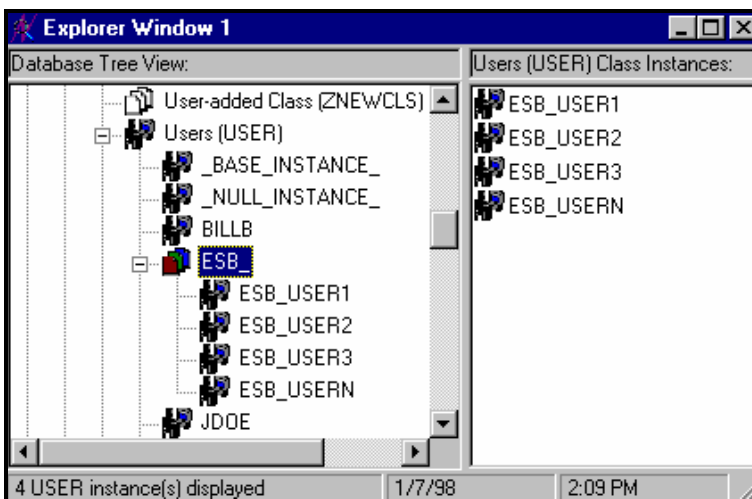


Click on **Prune Below**. The tree view collapses all open nodes below the SYSTEMX domain, and discards the history of their having been open:

As you configure more and more desktops with more and more applications, your EDM Database can grow quite large. For example, you will have an instance of the USER class for each desktop under management, and an instance of the ZRSOURCE class for each file to be distributed to any desktop. This can entail many thousands of instances.

Manipulating the tree view can grow unwieldy when large numbers of instances are involved. EDM helps you manage large numbers of instances by providing a filtering capability (described in the "*Filtering – Viewing Portions of the EDM Database*" section of this chapter). EDM also provides one, two or three additional collapsible levels in the tree view, based upon 1, 2 or 3 high-level qualifiers of the instance name.

To take advantage of the additional collapsible levels in the tree display, you must name your instances consistently with compound names. A compound name has 1, 2 or 3 prefixes; each separated from the rest of the name by an underscore character.



For example, you might want to group your users according to the building where they are located. Instance names for these users will be prefixed with a building identifier. If you had users in the Empire State Building, you might assign “ESB\_” as the instance name prefix to identify them. Each user in the Empire State Building will be assigned an instance name beginning with “ESB\_”. The tree view will automatically add a level, as shown here:



Like any other non-elementary level, clicking on the minus sign at the next higher level can collapse this level of the tree display.

The number of levels (0, 1, 2 or 3) of prefixes of instance names that will be displayed in the tree view is controlled by the **Show n Instance Prefixes in the Tree** option setting on the **General** tab of the **Options** dialog box, which can be accessed from the **View** menu. The **Options** dialog is described in the "EDM System Explorer Options" section in this chapter.

Each underscore in an instance name delimits a level for the tree view. Thus, the instance name ESB\_FLOOR10\_USER1 has two levels of prefix, which, if the setting **Shown Instance Prefixes in the Tree** were set to 2 or 3, would be displayed in two collapsible non-elementary levels (FLOOR10, within ESB).

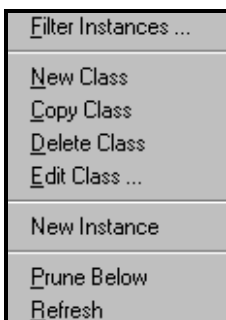
## Filtering – Viewing Portions of the EDM Database

As your distribution model grows, you will find it convenient to restrict the EDM System Explorer to view only a portion of the EDM Database. This saves effort in scrolling the window to locate the domain, class or instance you need to inspect or edit. The EDM System Explorer offers the capability of setting filters to provide user-defined restricted views of the EDM Database.

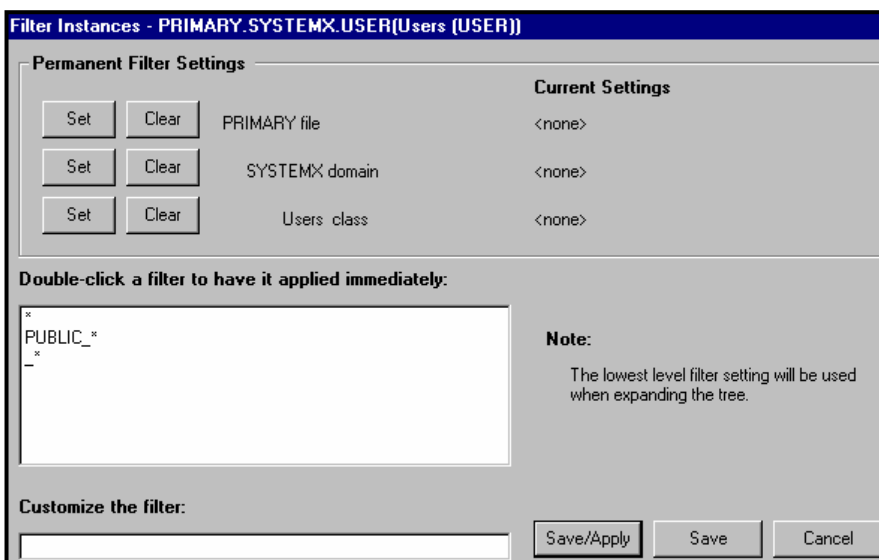
This is particularly valuable for classes that can have thousands of instances such as the USER class or the ZRSOURCE class.

You gain access to setting filters from the context menu that pops up when you right-click the file, domain or class icon in the tree view, whose domains, classes or instances you want to filter.

For example, double-click on the **SYSTEMX** domain in the tree view. You will see the classes of the SYSTEMX domain listed. Scroll down the tree view until you reach the User class. Right-click the USER class, and the following context menu pops up:



Click on **Filter Instances...** to open the **Filter Instances** dialog box.



You can enter a filter specification in the **Customize the filter:** text box.

The filter specification tells the EDM System Explorer which domains, classes or instances to display. Filtering is based upon matching the name of the domain, class, or instance with the filter specification you provide.

Filter specifications can contain wildcards, represented by the asterisk. An asterisk matches any character or characters in the name of the component being filtered.

For example, to filter the User class to display only users whose instance name begins with “ESB\_”, type `ESB_*` in the **Customize the filter:** field.

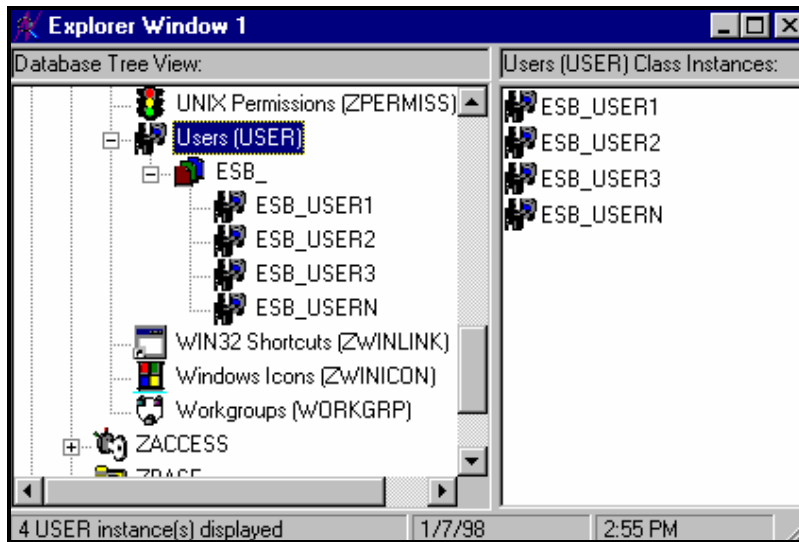
You can apply filters that remain in effect for an entire EDM System Explorer session (i.e. “Permanent Filters”), or filters that apply only to the current Explorer window.

Permanent Filter Settings			Current Settings
Set	Clear	PRIMARY file	<none>
Set	Clear	SYSTEMX domain	<none>
Set	Clear	Users class	ESB_*

Permanent filters are set and/or cleared at the top of the dialog in the **Permanent Filter Settings** area. If you want to filter the User class on “ESB\_\*” for the current EDM Explorer session, type `ESB_*` in the **Customize the filter:** text box, and press the **Set** button next to **Users class** in the **Permanent Filter Settings** area of the dialog box. The filter specification is displayed in the **Permanent Filter Settings** area:

To apply the filter, click the **Save/Apply** button at the bottom of the dialog box.

The dialog box will close, and the tree display will be updated with the filter applied.



This filter will be applied to the User class in any additional Explorer windows you open during the current EDM System Explorer session.

Each class can have its own permanent filter specification during an Explorer session. Right-click the class in the tree view to access the Filter Instances dialog box for that class.

To remove a permanent filter, click on the appropriate **Clear** button in the **Filter Instances** dialog box.

The **Filter Instances** dialog box also allows you to apply temporary filters. Temporary filters remain in effect only as long as the current Explorer window is open. You can accomplish this by entering a filter specification, and clicking on the **Save/Apply** button.

The box beneath the label **Double-click a filter to have it applied immediately** contains a list of potentially useful filter specifications which an EDM system administrator with appropriate authority at your site can customize.

These are express filter specifications. As the label indicates, simply double-click the specification of your choice to apply it immediately. The filter is temporary, applying only to the currently open Explorer window.

To remove a temporarily applied filter, apply \* as a filter specification. This matches any domain, class or instance name, effectively removing the filter. The \* filter specification is frequently found in the express filter list.

Here are a few examples of valid filter expressions, and what they select:

Filter specification	Selects
*	Wildcard to select all; removes any filter currently in effect
_*	Instances whose names' first character is an underscore.
_A	Instances whose names have a prefix and any level of the prefix is followed by an "A" (i.e. "_A" appears somewhere within the name).

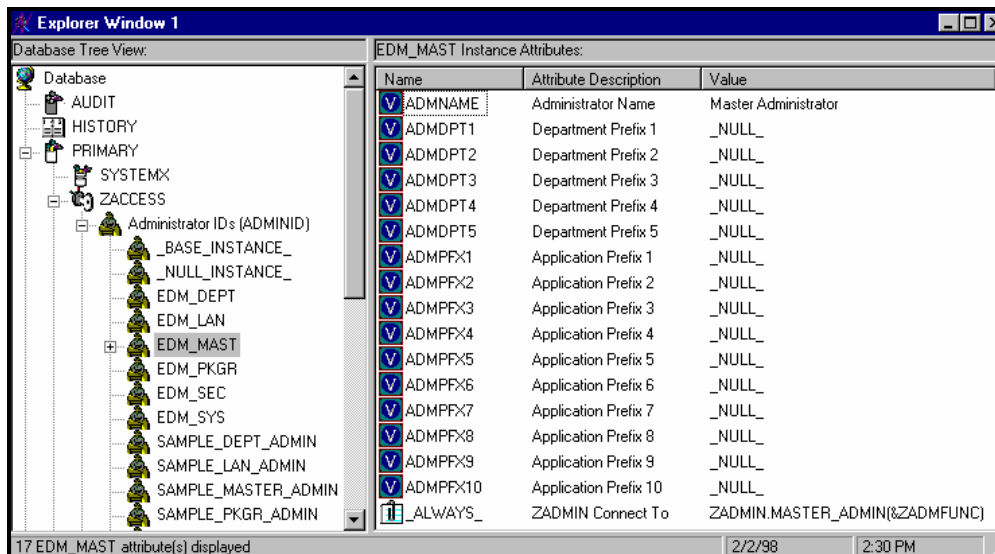
## Customizing Express Filters

You can specify the lists of express filters that appear for selection in the **Filter Instances** dialog by storing them in the EDM Database, and making any needed connections. The EDM System Explorer retrieves the list of express filters for a particular **Filter Instances** dialog box from the EDM Database as it opens.

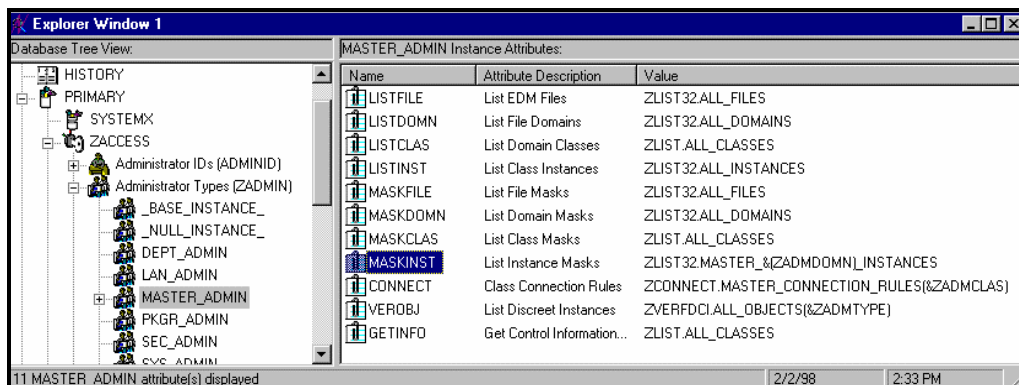
The EDM System Explorer locates the correct list in the EDM Database by executing a resolution process based upon the userid used when the EDM System Explorer session was started, and the EDM component (i.e. domain, class or instance) being filtered. Refer to *Appendix A, Understanding the Resolution Process*.

The list of express filters for a particular domain is drawn from an instance of the ZLIST32 class of the ZACCESS domain in the Primary file. Here's how.

Assume you've logged in with userid EDM\_MAST. The authority and defaults associated with an administrator's userid are stored in the ZACCESS domain of the Primary file. The ADMINID class contains an instance for each administrator's userid (in this case, the instance name is EDM\_MAST).



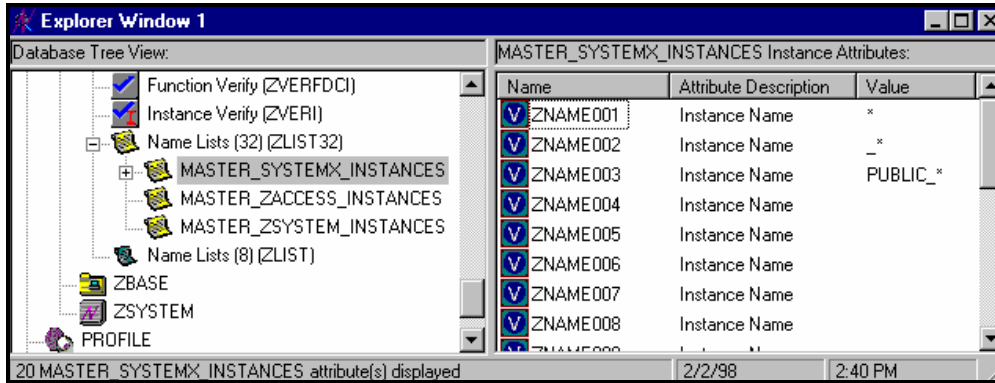
The EDM\_MAST instance contains a connection to an instance of the ZADMIN class, in this case ZACCESS.ADMINID.MASTER\_ADMIN.



The connection to ZACCESS.ZADMIN.MASTER\_ADMIN passes a parameter in the system message (&ZADMFUNC) that indicates which of the connections contained in the ZADMIN.MASTER\_ADMIN instance should be made. Only those connections whose names match the value of the system message (or whose name is ALWAYS) are actually made during a particular resolution execution. The EDM System Explorer sets the parameter's value before initiating the resolution. In this case the EDM System Explorer sets the value of the ZADMFUNC variable (and thus, after symbolic

substitution, the value of the system message) to MASKINST, because the Filters dialog needs to retrieve a list of instance masks.

This causes a connection to an instance of the ZLIST32 class, ZLIST32.MASTER\_&(ZADMDOMN)\_INSTANCES, based upon the domain to be filtered (SYSTEMX). The domain is dynamically set in the ZADMDOMN variable by the EDM System Explorer, based upon which class the administrator right-clicked to access the Filter Instances dialog. In the case of our example, the connection will be made to ZLIST32.MASTER\_SYSTEMX\_INSTANCES (after the resolution process performs symbolic substitution), which contains the desired list of default express filter specifications:



To change the list of express filters for a particular domain, simply edit the appropriate instance of the ZLIST32 class. The changes will affect all administrators whose userids connect to the ZLIST32 instance you modify.

To set custom express filters for particular administrators, use the information detailed above to connect their userid instances in the ZACCESS.ADMINID class to the appropriate ZLIST32 instance containing the desired list of express filters.



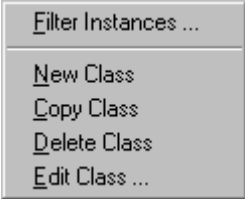
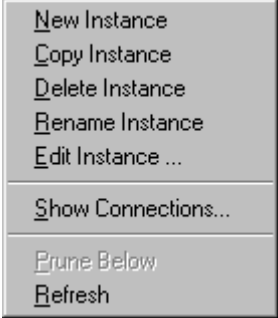
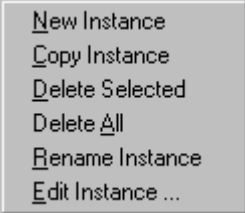

See the "Adding an Instance" and "Editing Instances" sections of this chapter for information on how to use the EDM System Explorer to make these changes.

### Manipulating EDM Database Components

The EDM System Explorer provides the capability for adding, copying, editing, renaming and deleting EDM Database components (domains, classes and instances).

You can access these functions from context menus that pop up when you right-click targets in the EDM System Explorer window. The menu that pops up contains different choices depending upon which target you right-click. The following table shows how the choices in the pop up menu can differ based upon the target of the right-click.

Target of Right Click	Menu that Pops Up
Primary File in the tree view	<div style="border: 1px solid gray; padding: 5px; width: fit-content;"> <p>Filter Domains ...</p> <p>New Domain</p> <p>Prune Below</p> <p>Refresh</p> </div>
File other than Primary File in the tree view	<div style="border: 1px solid gray; padding: 5px; width: fit-content;"> <p>Filter Domains ...</p> <p>Prune Below</p> <p>Refresh</p> </div>

Target of Right Click	Menu that Pops Up
Any Domain in the tree view	 <ul style="list-style-type: none"> <li>Filter Classes ...</li> <li>New Domain</li> <li>New Class</li> <li>Prune Below</li> <li>Refresh</li> </ul>
Any Class in the tree view	 <ul style="list-style-type: none"> <li>Filter Instances ...</li> <li>New Class</li> <li>Copy Class</li> <li>Delete Class</li> <li>Edit Class ...</li> <li>New Instance</li> <li>Prune Below</li> <li>Refresh</li> </ul>
Any Class in the right side of the Explorer window	 <ul style="list-style-type: none"> <li>Filter Instances ...</li> <li>New Class</li> <li>Copy Class</li> <li>Delete Class</li> <li>Edit Class ...</li> </ul>
Any Class Instance in the tree view	 <ul style="list-style-type: none"> <li>New Instance</li> <li>Copy Instance</li> <li>Delete Instance</li> <li>Rename Instance</li> <li>Edit Instance ...</li> <li>Show Connections...</li> <li>Prune Below</li> <li>Refresh</li> </ul>
Any Class Instance in the right side of the Explorer window	 <ul style="list-style-type: none"> <li>New Instance</li> <li>Copy Instance</li> <li>Delete Selected</li> <li>Delete All</li> <li>Rename Instance</li> <li>Edit Instance ...</li> </ul>
White space in right side of the Explorer window with any Class Instances displayed	 <ul style="list-style-type: none"> <li>New Instance</li> </ul>

As you can see, the target of a right-click can be a node in the tree view, an object in the right side of the Explorer window, or white space in the right side of the Explorer window. In all cases, the pop up menu choices is appropriate to the target of the right-click.

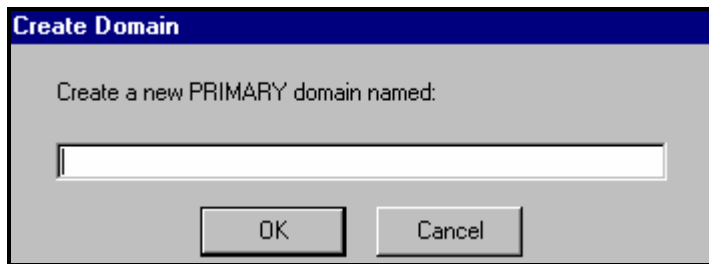
If you select any of the **Filter...** choices from a pop up menu, the Filter dialog appears. The Filter dialog is described in the "Filtering – Viewing Portions of the EDM Database" section of this chapter.

## Working with Domains

### Adding a Domain

#### ➤ To Add a Domain to a File:

- 1 Right click on the Primary file or any domain, and select **New Domain** from a context menu. The following dialog appears.



- 2 Enter a name for the new domain and click the **OK** button. Domain names can be a maximum of 8 characters including letters and numbers.

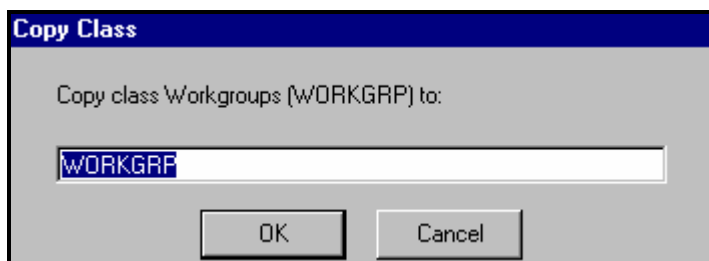
## Working with Classes

### Adding a Class

New classes can be added by copying an existing class, or adding a new class from scratch. Which to choose depends upon whether a class exists whose attributes are substantially similar to the class you want to add. If so, copy the existing class; otherwise, add a new class from scratch.

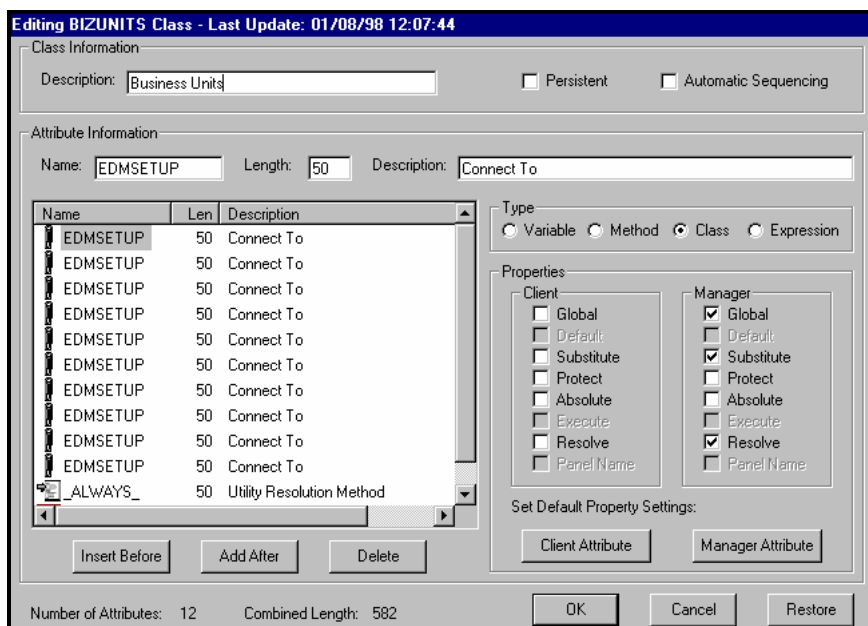
Copying a class creates a new class with the same attributes as the copied class, and a base instance for the new class with attribute values set as in the base instance of the copied class.

#### ➤ To Copy an Existing Class:



- 1 Choose **Copy Class** from a context menu. The following dialog appears.
- 2 Enter a name for the new class. For this example, enter BIZUNITS. Click the **OK** button.

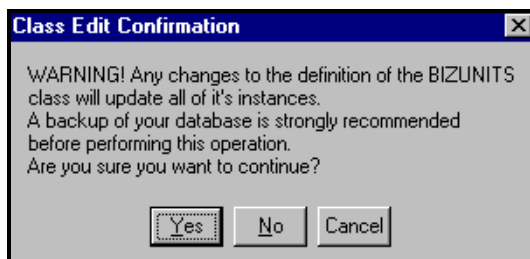
If the option to open the class editor when a class is copied is in effect (see the *"EDM System Explorer Options"* section of this chapter for details), the following will appear.



This is the **Editing Class** dialog box, which is described in the *"Editing a Class"* section of this chapter.

If the option to open the class editor when a class is copied is not in effect, the original class is copied with the new name, the class editor is not opened, and no warning (see next step) is issued. You will have to manually open the class editor to tailor the newly copied class.

To complete the addition of the copied class, change the **Description** field to identify the newly added class, edit the class to reflect the new class' requirements, and click the **OK** button. The following appears.

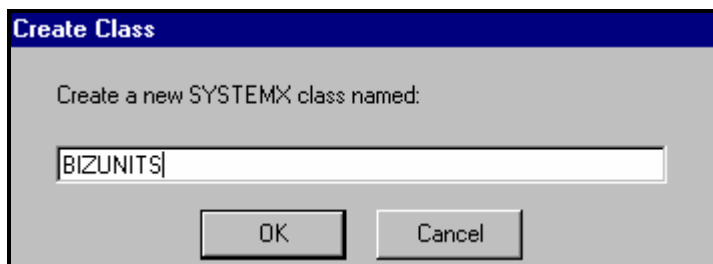


2 Click the **Yes** button, and the new class addition is complete.

### ➤ To Add a new Class from Scratch:

1 Choose **New Class** from a context menu. The following dialog box appears.



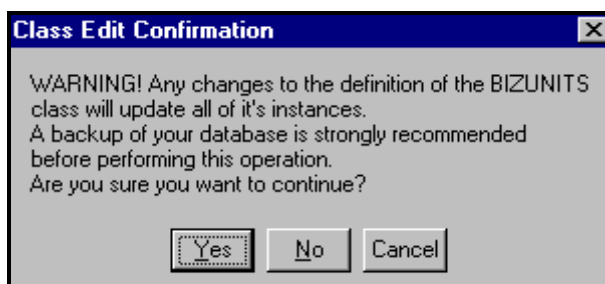


- 2 Enter a name for the new class, and click the **OK** button.

If the option to open the class editor when a new class is added, is in effect (see the "*EDM System Explorer Options*" section of this chapter for details), the **Editing Class** dialog box will appear. The **Editing Class** dialog is described in the **Editing a Class** section of this chapter.

If the option to open the class editor when a new class is added is not in effect, a new class is added with the specified name, the class editor is not opened, and no warning (see next step) is issued. You will have to manually open the class editor to tailor the newly added class.

- 3 To complete the addition of the new class, change the **Description** field to identify the newly-added class, edit the class to reflect the new class' requirements, and click the **OK** button. The following message appears.



- 4 Click the **Yes** button, and the new class addition is complete.

## Editing a Class

When you choose **Edit Class...** from a context menu, or during the addition of a new class as described in the "*Adding a Class*" section of this chapter, the **Editing Class** dialog box appears.

### ➤ To Edit a Class:

- 1 For example, right-click on the DEPT class, and choose **Edit Class...** from the resulting context menu. The **Editing Class** dialog appears.

**Editing DEPT Class - Last Update: 10/22/97 14:45:47**

Class Information  
 Description:   Persistent  Automatic Sequencing

Attribute Information  
 Name:  Length:  Description:

Name	Len	Description
<input checked="" type="checkbox"/> ACCTNO	12	Account Number
<input type="checkbox"/> EDMSETUP	50	Connect To
<input type="checkbox"/> EDMSETUP	50	Connect To
<input type="checkbox"/> EDMSETUP	50	Connect To
<input type="checkbox"/> _ALWAYS_	51	Utility Resolution Method
<input checked="" type="checkbox"/> DEPTNAME	32	Department Name

Type  
 Variable  Method  Class  Expression

Properties

Client	Manager
<input type="checkbox"/> Global	<input checked="" type="checkbox"/> Global
<input type="checkbox"/> Default	<input type="checkbox"/> Default
<input type="checkbox"/> Substitute	<input checked="" type="checkbox"/> Substitute
<input type="checkbox"/> Protect	<input type="checkbox"/> Protect
<input type="checkbox"/> Absolute	<input type="checkbox"/> Absolute
<input type="checkbox"/> Execute	<input type="checkbox"/> Execute
<input type="checkbox"/> Resolve	<input type="checkbox"/> Resolve
<input type="checkbox"/> Panel Name	<input type="checkbox"/> Panel Name






Set Default Property Settings:

Number of Attributes: 6 Combined Length: 245

The Class Information properties at the top of the dialog may be set as follows:

Class Information Property	Significance and Setting
Description	Text that describes the class and/or its purpose. This text appears next to the class name in the tree view.
Persistent	<p>This property is available to EDM architects with a thorough understanding of the EDM resolution process.</p> <p>Instances created from persistent classes are available to the EDM Client for storage on the desktop. The attributes of objects instantiated from persistent classes do not resolve into parent objects during resolution. Refer to <b>Appendix A, Understanding the Resolution Process</b>.</p> <p>If a class is not persistent, it is transient. Instances created from transient classes are not available to the EDM Client for storage on the desktop. The attributes of objects instantiated from transient classes can be incorporated into parent objects during resolution.</p>
Automatic sequencing	<p>Normally, resolution proceeds from attribute to attribute in the order the attributes appear in the list of attributes in this dialog. Refer to <b>Appendix A, Understanding the Resolution Process</b>.</p> <p>Marking this check box imposes a grouping on the order in which attributes of the class are processed during resolution:</p> <ul style="list-style-type: none"> <li>• Expressions</li> <li>• Variables</li> <li>• Classes (connections)</li> <li>• Methods</li> </ul> <p>Groups are processed in the order in which they appear, above, and within these groupings, the attributes are processed in the order in which they appear in the list of class attributes.</p> <p>Unless you need to alter the normal sequence of processing attributes during resolution, leave this check box unmarked.</p>

Class attributes can be one of four types:

Icon	Attribute type	Usage
	Expression	Contains a single line REXX command that is executed during resolution. In an attribute named ZSTOP, causes resolution to terminate if the expression evaluates to "true".
	Variable	A piece of named storage containing a variable value. The variable's value forms a part of the client's resolved distribution model, and can influence the resolution process through messaging or symbolic substitution.
 	Connection (available) Connection (set)	Class connections determine the path of resolution of a client's distribution model during the client connection process. Available connections are attributes into which a connection to another class may be set. Set connections are attributes containing a specification for a connection to another class. A class connection is, in effect, a branch in the resolution process.
	Method	Methods are programs executed as part of the resolution process. The method attribute identifies the program to be executed.

Attribute names have significance in the resolution process. Values of variables in child objects instantiated from transient classes can overlay the values of identically named variables in parent objects instantiated from persistent classes. Methods can be conditionally executed or skipped, and class connections can be conditionally followed or ignored based upon the name of the method or class connection attribute. If the attribute's name is **ALWAYS**, the method will be executed or the class connection will be followed unconditionally. Otherwise, the method will be executed, or the class connection will be followed only if the name of the attribute is identical to the current value of the system message. Refer to *Appendix A, Understanding the Resolution Process*.

- To modify the definition of an attribute, click on its name in the list to highlight it. The name, length and description of the highlighted attribute appear in the **Attribute Information** data entry fields, and the **Type** option is marked (as a Variable, Method, Class, or Expression) to indicate the type of the highlighted attribute.
- Type your desired changes into the **Attribute Information** data entry fields, and click on the desired **Type** option to set the attribute's type. As you make changes, they are immediately reflected in the attribute list.
- The **Properties** check boxes are either available to be marked ("enabled") or not available to be marked ("disabled") depending upon the attribute type.

All Client properties, except for **Protect** and **Execute**, are defined for future enhancements, and you should leave them set to their default values. The Client **Protect** property, when marked, indicates that the value of an attribute stored in an object on the Client desktop should be encrypted. The Client **Execute** property, which applies only to method attributes, indicates, when marked, that the method to be executed will be executed on the Client computer.

The following table summarizes the significance of the Manager properties:

Property	Significance and Setting
Global	When marked for a variable in a transient class, indicates that the variable will be incorporated into an object instantiated from a parent persistent class from an object instantiated from the transient child class, during resolution.
Default	Used internally by EDM and is not available to EDM Administrators. When marked, prevents default values in objects instantiated from transient classes from being incorporated into objects instantiated from parent persistent classes.
Substitute	When marked, indicates that symbolic substitution will occur for symbols (references to other attributes identified by an initial ampersand) in the value of the attribute. Unmarking this property treats the value of the attribute as literal text, preventing symbolic substitution from occurring.
Protect	When marked, indicates that value of attribute is to be encrypted during resolution. The attribute value exists in encrypted form in memory during resolution, but remains as plain text in the EDM Database.
Absolute	When marked for a variable in a persistent class, prevents identically named variables in objects instantiated from child transient classes from overlaying the values for those variables in the parent object instantiated from that persistent class.
Execute	When marked, indicates that the value of the attribute identifies a method to be executed by the EDM Manager.
Resolve	When marked for a connection attribute, indicates that the attribute value will effect a connection to another class; otherwise, the attribute is treated as a variable.
Panel Name	Used internally by EDM and is not available to EDM Administrators.

You can make sure that the property settings for the currently selected (i.e. highlighted) attribute have their default values by clicking the **Set Default Property Settings** buttons.

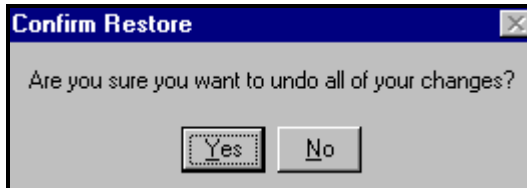
- 5 To add a new attribute, first determine where in the list the attribute should be inserted. Attributes are normally processed during resolution in the order in which they appear in the list, unless the **Automatic Sequencing** check box is marked, as described, above.
- 6 Click on an attribute adjacent to the insertion location, to highlight it. Then click the **Insert Before** button to insert the new attribute above the highlighted attribute, or click on the **Add After** button to insert the new attribute after the highlighted attribute.  
A blank attribute (a variable attribute by default) is inserted at the indicated location.
- 7 Enter a name, length and description in the Account Information data entry fields and mark the Type option to indicate the desired attribute type.

Attribute names can be up to eight characters in length. Refer to *Appendix A, Understanding the Resolution Process*.

The length of the attribute is specified in characters. Enter a number greater than or equal to the maximum number of characters for a value of the attribute. For class connection and method attributes, a length of approximately 50 is normally adequate. The length of a variable attribute depends on the data it is intended to hold.

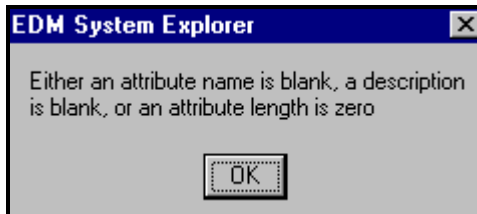
The description you enter will appear in the **Attribute Description** column of the **Edit Instance** dialog box when editing instances of this class. The clearer the description you enter now, the easier it will be for those who edit instances of this class later. For variable attributes, enter a description that clearly identifies the data to be held in the variable attribute. For class connections, you can enter “Connect to”, or one that may be more meaningful for your purposes. For methods and expressions, try to indicate the purpose of the method or expression in the description.

- 8 As you make changes to a class, you may change your mind and wish to start over. Once a change has been made during class editing, the **Restore** button will be enabled. Click this button to discard changes you’ve made to the class since the Editing Class dialog opened. The following appears:



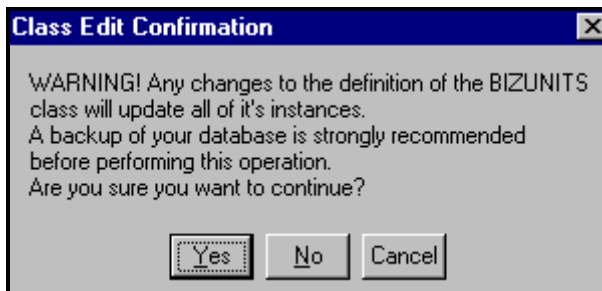
Click on the **Yes** button to discard changes and begin editing the class from the beginning. Click on the **No** button to dismiss the confirmation without discarding your changes.

When you have completed editing the class, click on the **OK** button to save your changes and exit the **Editing Class** dialog box.



If there are blank attribute names or descriptions in the class, or if an attribute's length is zero, the following warning will appear.

If there are no blank attribute names or descriptions in the class, and no attribute's length is zero, a change confirmation dialog similar to the following will appear:

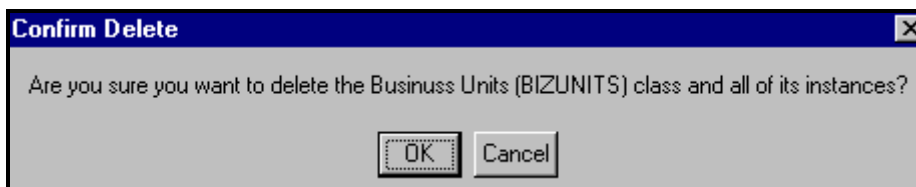


- 12 Click the **Yes** button to update all existing instances of the class. Click the **No** button to discard your changes to the class, and avoid updating the instances. Click the **Cancel** button to return to the **Editing Class** dialog box.

## Deleting a Class

### ➤ To Delete a Class and all of its Instances:

- 1 Right-click on the class in the tree view.
- 2 Select **Delete Class** from the context menu that pops up. A confirmation dialog similar to the following appears.



- 3 Click the **OK** button to delete the class and all of its instances. Click the **Cancel** button to dismiss the dialog without deleting anything.

## Working with Instances

Much of the work of EDM administrators consists of adding, editing and deleting instances of various classes, and setting or removing connections between them.

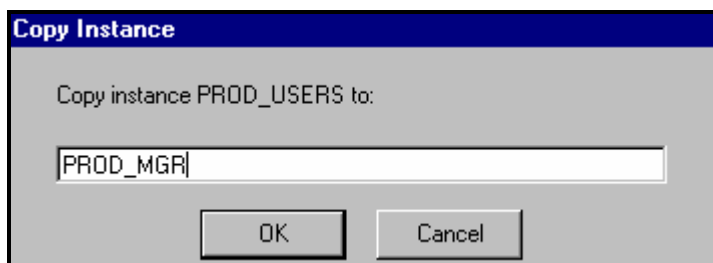
### Adding an Instance

New instances can be added by copying an existing instance, or adding a new instance from scratch. Which to choose depends upon whether an instance exists whose attribute values are substantially similar to the instance you want to add. If so, copy the existing instance; otherwise, add a new instance from scratch.

Copying an instance creates a new instance with the same attribute values as the copied instance. Adding a new instance from scratch creates a new instance with attribute values set as in the base instance of the class.

#### ➤ To Copy an Existing Instance:

- 1 Choose **Copy Instance** from a context menu. A dialog box similar to the following appears.

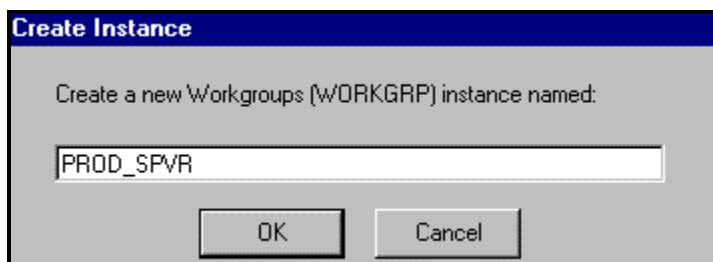


- 2 Enter a name for the new instance, and click the **OK** button.

The instance will be created and will appear in both the tree view and in the right side of the System Explorer window.

#### ➤ To Add a new Instance from Scratch:

- 1 Choose **New Instance** from a context menu. A dialog box similar to the following appears.



- 2 Enter a name for the new instance, and click the **OK** button.

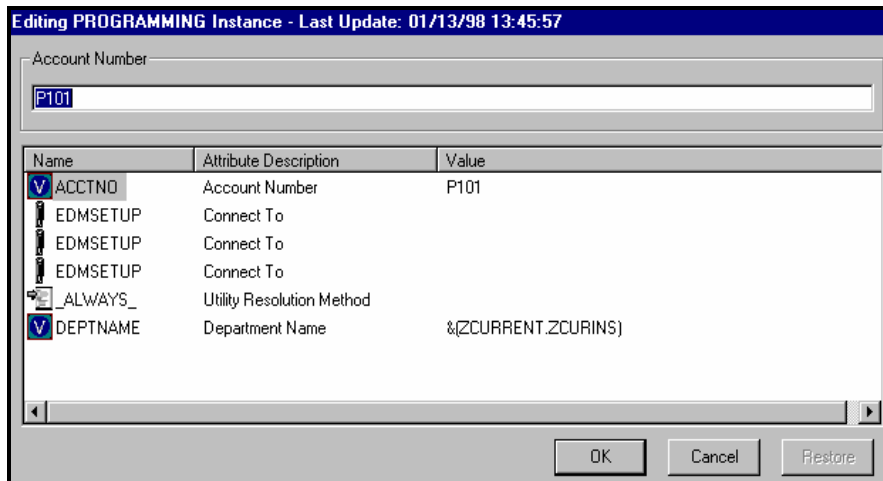
The instance will be created and will appear in both the tree view and in the right side of the System Explorer window.

You may now edit the new instance to contain attribute values intended for the new instance.

## Editing Instances

### ➤ To Edit an Instance:

- 1 Choose **Edit Instance** from a context menu, or double-click an attribute's name in either the tree view or in the right side of the System Explorer window. The **Editing Instance** dialog appears.



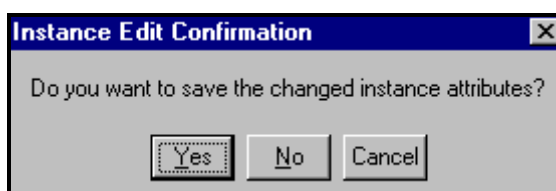
If you choose **Edit Instance** from a context menu, the dialog box opens with the first attribute in the instance opened for editing. If you double-click an attribute name to open the dialog box, the attribute whose name you double-clicked is open for editing.

The dialog lists the attributes comprising the instance, and their current values. The highlighted attribute is open for editing in the data entry field at the top of the dialog.

- 2 Type the desired value into the data entry field.
- 3 To highlight and select a different attribute, click on its name.

**Note:** When editing Connect to: attributes, notice the warning about editing class connections manually, at the bottom of the dialog. Class connection values can be complicated to type, and a single typographical error can produce unexpected and undesirable results during resolution. The recommended way to make class connections between attributes is to drag-and-drop them, which is described in the **Drag-and-drop connections** section of this chapter.

- 4 Continue to select each attribute you want to edit, and to type in the desired value for each, until you have completed editing the instance.
- 5 Click the **OK** button to close the dialog and save your changes. The following confirmation dialog box appears.

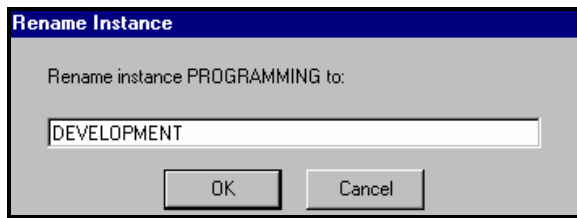


- Click the **Yes** button to complete the update of the instance. Click the **No** button to discard your changes to the instance, and return to the System Explorer main window. Click the Cancel button to return to the **Editing Instance** dialog box.

## Renaming Instances

### ➤ To Rename an Instance:

- Select **Rename Instance** from a context menu. The context menu can be accessed by right clicking on the instance to be renamed. A dialog similar to the following appears:



- Type in a new name for the instance, and click on the **OK** button to make the change.

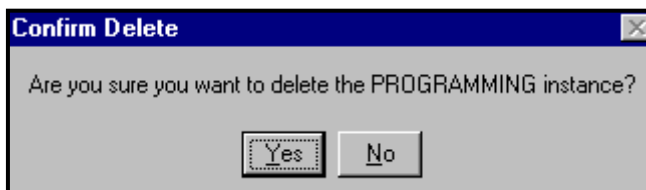
**Note:** Be careful when renaming instances. Instances are connected together via the class connection attributes they contain. When you rename an instance, any other instance connected to the renamed instance will have a broken connection. This can produce unexpected and undesirable effects during resolution. Before renaming an instance, make sure you know which other instances are connected to the renamed instance, and edit them appropriately.

## Deleting Instances

### ➤ To Delete an Instance:

- Select **Delete Instance** from a context menu. The context menu can be accessed by right clicking on the instance to be deleted.

A confirmation dialog box appears.



- Click on the **Yes** button to delete the instance, or the **No** button to dismiss the dialog without deleting the instance.

**Note:** Be careful when deleting instances. Instances are connected together via the class connection attributes they contain. When you delete an instance, any other instance connected to the deleted instance will have a broken connection. This can produce unexpected and undesirable effects during resolution. Before deleting an instance, make sure you know which other instances are connected to the instance to be deleted, and edit them appropriately.



## Drag-and-Drop Connections

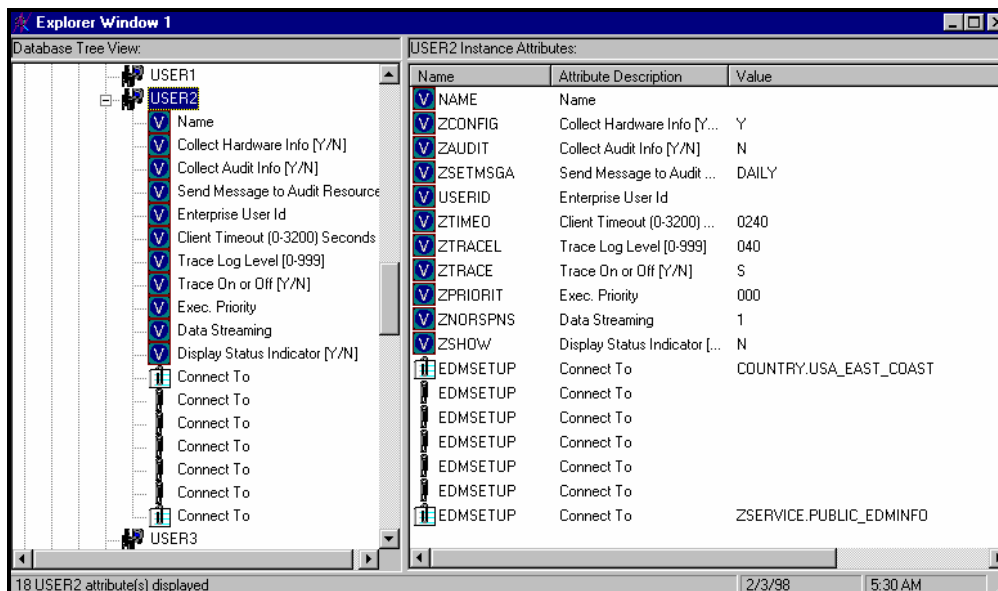
The EDM System Explorer features the ability to make drag-and-drop connections between instances of connectable classes. Using this feature, you will avoid two common errors that occur when you type in connection specifications between instances:

- Typographical error
- Typing in a connection to an instance of a class with which connection is prohibited.

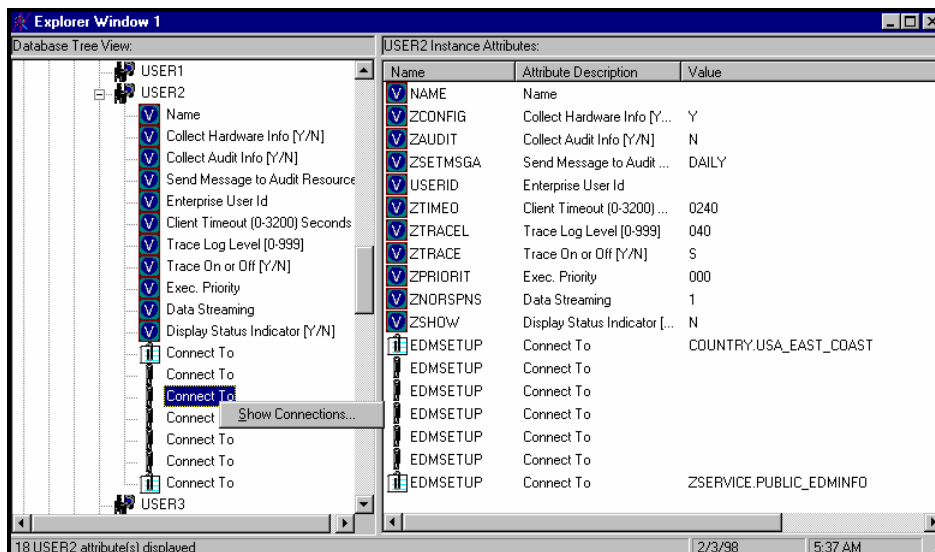
Thus, making drag-and-drop connections is the recommended method of connecting instances.

### ➤ To Make a Drag-and-Drop Connection:

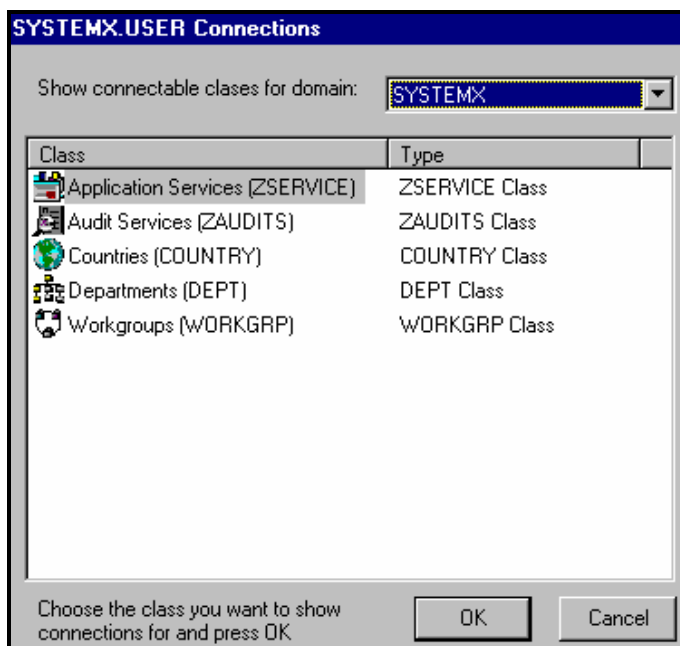
- 1 In the tree view, navigate to the instance that will contain the connection specification, and double-click the instance name. For example, we'll connect an instance of the Department class to an instance of the User class. Navigate to the desired User class instance in the tree view, and double-click to display its contents in the right side of the Explorer window. Then click on the plus sign next to the instance name in the tree view to list the attributes of the instance beneath the instance name in the tree view. The result will appear similar to the following window.



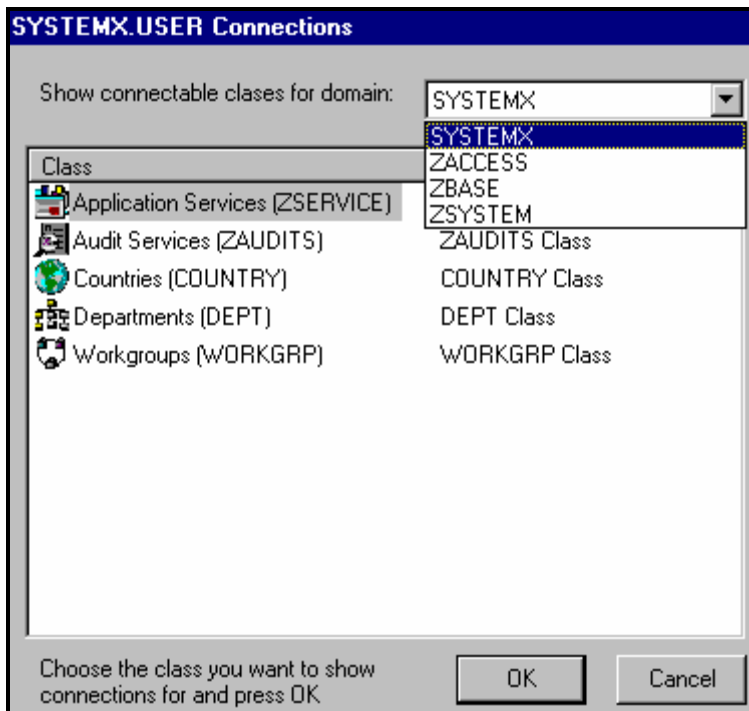
- 2 Identify which class connection attribute you want to set, and right-click it in the tree view. If the class connection attribute you choose is already set (i.e. has a value), the existing value will be replaced. A context menu pops up.



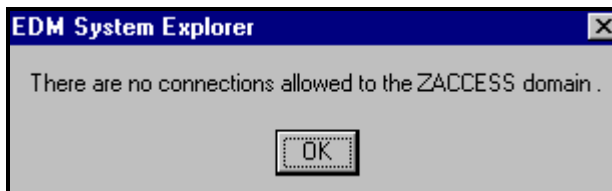
- 3 Click on the **Show Connections...** menu. A dialog box similar to the following appears.



This dialog lists the classes within the same domain, to which connection of the selected instance is permitted. To connect to a class in a different domain, click the pull down list labeled **Show connectable classes for domain:** and a list of available domains will appear.



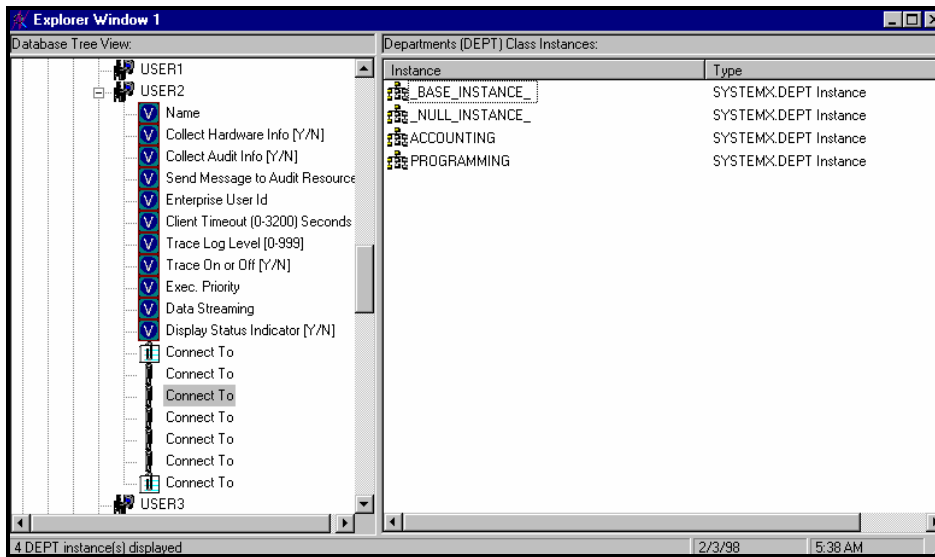
- 4 Select the desired domain. The list of permitted connectable classes from that domain will appear. If there are no permitted connectable classes in the chosen domain, an alert message appears.



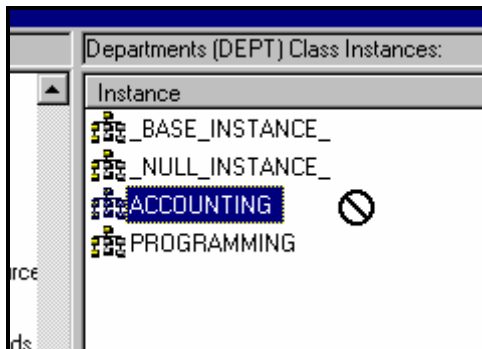
- 5 Click on the **OK** button to close the alert message.

Rules for which classes can be connected together are contained in the EDM Database. See the EDM Manager Administration Guide for details of how to maintain these rules.

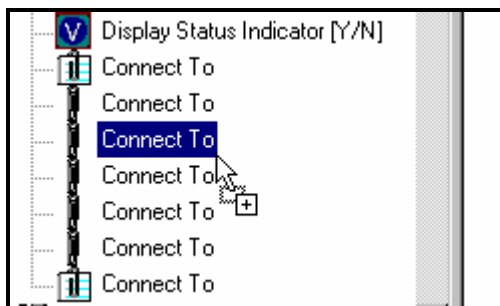
Double-click the desired class (in our example, the Departments class). The instances of the chosen class will appear on the right side of the Explorer window:



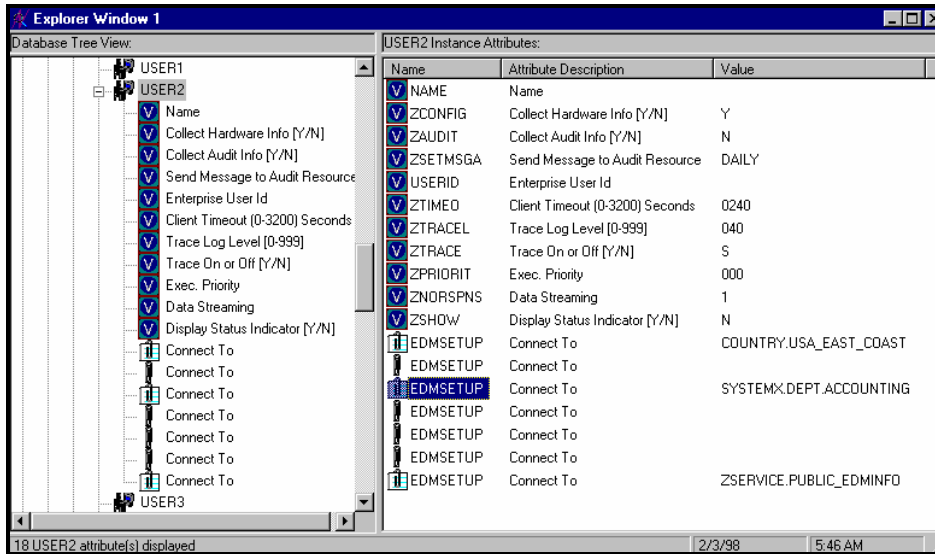
- 6 Click and drag the desired instance (choose Accounting for this example) from the right side of the Explorer window toward the desired connection attribute in the tree view. As you do so, the mouse pointer will change to a slashed circle, indicating that the mouse pointer has not yet been positioned over the drop target.



- 7 Drag the mouse to the tree view and position the pointer over the drop target. The mouse pointer will change to indicate you've reached an acceptable drop target.



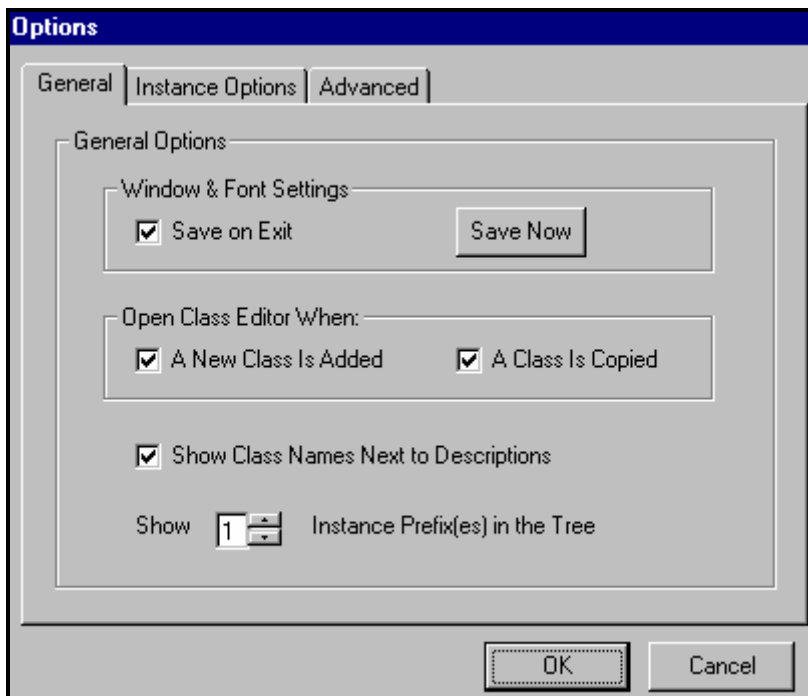
- 8 Release the mouse button to drop the connection on the target. The desired connection is set, and the results are displayed in the Explorer window.



## EDM System Explorer Options

The EDM System Explorer provides an Options dialog to allow you to control some of the features of the EDM System Explorer.

- 1 To open the **Options** dialog box, select **Options...** from the **View** menu. The following dialog box appears.



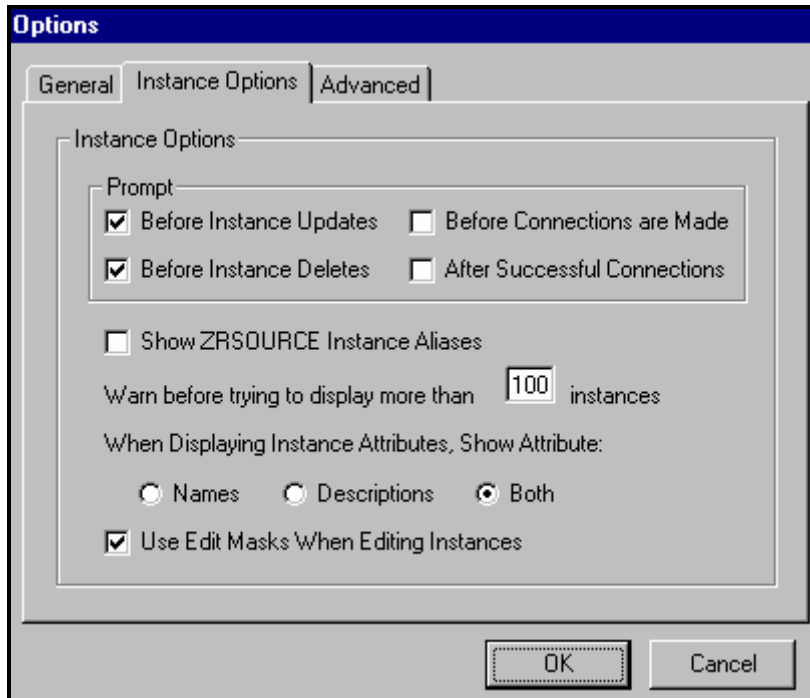
- 2 There are three tabs in the **Options** dialog box: **General**, **Instance Options** and **Advanced**. To display the settings associated with a tab, click on it.

The following settings are available on the General tab:

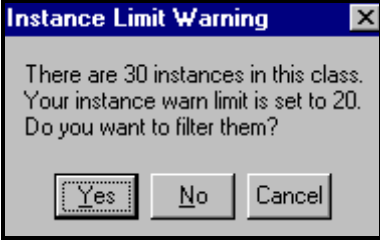
General Tab	What It Does and How to Use It
Window & Font Settings	Mark the <b>Save on Exit</b> check box to have the EDM System Explorer remember

General Tab	What It Does and How to Use It
	<p>certain window settings and the font to use when displaying text, when you exit from the an EDM System Explorer session.</p> <p>The window settings that are remembered include the view setting (i.e. large icons, small icons, list, or detail) for the tree view and the right side of the System Explorer window, and the relative width of each. Window settings are saved independently for each level of the database.</p> <p>Click on the <b>Save Now</b> button to have EDM System Explorer save the current window settings and font in use.</p> <p>You can change the font EDM System Explorer uses to display text by choosing a font from the dialog that opens when you select <b>Fonts...</b> from the View menu.</p>
Open Class Editor When	<p>Mark the <b>A New Class is Added</b> check box to have EDM System Explorer open the <b>Editing Class</b> dialog when you add a class.</p> <p>Mark the <b>A Class is Copied</b> check box to have EDM System Explorer open the <b>Editing Class</b> dialog when you copy a class.</p> <p>You will probably want to edit the newly copied or newly added class, so it is recommended that you have these two check boxes marked.</p>
Show Class Names Next to Descriptions	<p>Mark this check box to show the EDM internal name for classes next to their descriptions in the tree view and in the right side of the System Explorer window. The internal name will appear within parentheses next to the class description.</p> <p>This is useful if you ordinarily do not use the Details view for the right side of the System Explorer window. The Details view in the right side of the System Explorer window always shows the internal name for classes in the <b>Type</b> column.</p>
Show n Instance Prefix(es) in the Tree	<p>This option can be set to 0, 1, 2, or 3. It controls how many levels of instance name prefixes (delimited by the underscore character) will be compressible and expandable in the tree view. This is described in the "<i>Manipulating the Tree View</i>" section of this chapter.</p>

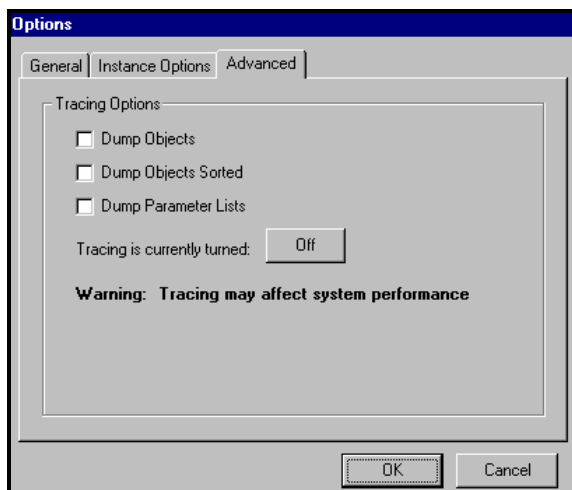
3 When you click on the **Instance Options** tab, the following appears.



The following settings are available on the **Instance Options** tab:

Instance Options Tab	What It Does and How To Use It
Prompt	These four check boxes control whether EDM System Explorer prompts you before completing certain actions, as indicated by the description next to each check box. The prompt consists of an alert asking you to confirm your desire to complete the action before anything is permanently changed in the EDM Database. Marking these is a matter of personal preference.
Show ZRSOURCE Instance Aliases	Mark this check box to add a column to the Details view of the right side of the System Explorer window, when the ZRSOURCE class is highlighted in the tree view. The additional column shows the contents of the ZRSCCFIL attribute, which is the name of the file represented by the ZRSOURCE instance.
Warn before trying to display more than n instances	<p>In larger distribution models, certain classes such as ZRSOURCE or USER can grow to contain many instances.</p> <p>If you attempt to view the instances of a class that contains more instances than this option setting, a warning similar to the following appears:</p>  <p>This gives you the opportunity to filter the class instances, before they are displayed. This is a performance issue. If you try to display too many instances, you will wait while the EDM System Explorer retrieves them from the EDM Database, and you will be inconvenienced by the need to scroll within a large list to find the instance(s) you want to work on or inspect.</p> <p>Click the <b>Yes</b> button to open the <b>Filter Class</b> dialog, and enter an appropriate filter.</p> <p>Click the <b>No</b> button to display all of the instances.</p> <p>Click the <b>Cancel</b> button to close the warning without displaying any instances.</p>
When Displaying Instance Attributes, Show Attribute	This setting determines which columns to display (Name, Attribute Description, or both) in the Details view of the right side of the System Explorer window, when you open an instance by double-clicking its name in the tree view.
Use Edit Masks When Editing Instances	<p>This setting determines whether the Edit Instance dialog will be sensitive to the attribute type in the data entry area of the dialog.</p> <p>If this check box is marked, attributes which hold logical (i.e. Yes/No) data will appear as a check box in the data entry area, and attributes that have designated multiple choices will appear as a drop-down list in the data entry area.</p> <p>You can set the desired mask by entering the valid choices for an attribute within square brackets at the end of the attribute's description in the class definition.</p> <p>If this check box is not marked, all data entry in the Edit Instance dialog will be plain text.</p>

4 When you click on the **Advanced** tab, the following appears.



**Note: These settings are only used for troubleshooting the EDM System Explorer. Do not change these option settings without consulting Customer Support.**



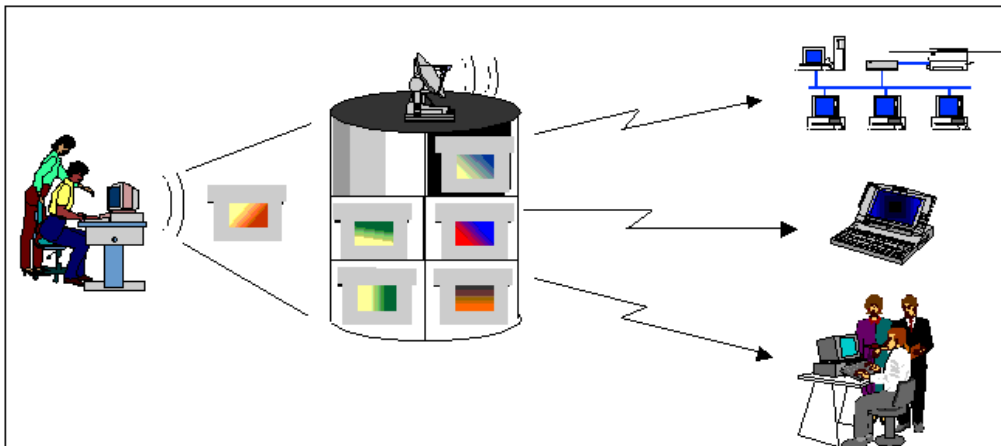
# Using EDM Packager

EDM Packager is used to publish software applications to the EDM database so you can distribute them to your EDM Clients. Packager automates the process of identifying the components of any software application.

## Making Packages Available

EDM gives you the power to distribute files and software programs throughout your network environment.

Use either Packager or Publisher to publish your packages to the EDM database so they can be distributed to your EDM Clients.



## EDM Packager vs EDM Publisher

The table below explains when to use each tool to publish your packages.

Tool:	Used to Publish:	Benefit
Packager	Complete software applications. Complete software applications, software updates, and other products with many dispersed, and unknown components should be published with Packager.	Automatically gathers all of the components of an application (including DLLs, INIs, and EXEs).
Publisher	Data files, or software application upgrade files. When you use this tool, you need to manually identify the components required for the package. If you are certain that you can identify all of the components for the package, you will want to use Publisher.	You must manually select the components you want to package and publish.

## Packager Overview

---

Packaging an application involves four main steps:

- 1 Select the components you want to package.
- 2 Edit the components you want to package.
- 3 Edit the properties of the package itself.
- 4 Define how you will publish the package.

The table below will help you locate the information you need to complete the process outlined above. The steps referenced are in the section titled "*Packaging an Application*" in this chapter.

If you want to learn how to:	read steps:
Begin a Packager session	1
Select the components for your package	2 – 10
Edit the components of your package	11
Edit the properties of your package	12
Define how the package will be stored in the database	13 – 14

### Before You Start

You should use Packager:

- 1 If you possess a basic understanding of the software you are packaging. You should be aware of the files that are likely to be altered by the installation program. (Consult the software installation manual for this information.)
- 2 On a desktop with ample free disk space. Packager's scanning process will take longer to complete if the machine is heavily loaded with applications.
- 3 On a desktop that has never had the software you are packaging installed on it.
- 4 If you have an EDM Administrator ID.

## User vs Machine

---

If you are packaging an application for EDM Clients that are using the Windows NT operating system, you can configure the component to deploy to either the machine, or the userID on the machine.

When a component is deployed to a machine, all users will have access to it. When you deploy a component to specific users, it will only be available to those specific userIDs.

## Additional Considerations

Applications can only be packaged and deployed to individual userIDs under the Windows NT operating system.

EDM Clients receiving components for specific userIDs are required to conduct the EDM Client stand-alone installation. Refer to *Welcome to EDM*, Chapter 3, "Installing the EDM Client for Windows NT and Windows 95."

EDM Clients receiving components for a machine can conduct the EDM Client stand-alone, or multiple NT user installation.

If You Package Software On	You Can Deploy To:
Windows 95	A Windows 95 EDM Client
Windows NT for a machine	A Windows NT user or multi-user installed EDM Client.
Windows NT for a userID	A Windows NT user installed EDM Client.

When you deploy components to a machine with multiple users, you must establish a userID in the user class of the EDM database for the machine, and the user (i.e., if there were two users on one machine, you would need three IDs). The components you deploy to the machine will only be received after the machine performs the EDM Client connect process.

## Using Packager




Packager is installed with the EDM Administrator. The table below explains how you can open it.

In Windows	Select
NT 4.0 and 95	<b>Programs</b> from the <b>Start</b> menu, then <b>EDM Administrator</b> , then <b>EDM Packager</b> .
NT 3.51	The <b>EDM Administrator</b> program group from the Program Manager, then choose the <b>EDM Packager</b> icon.

**Note:** When packaging an application, you must use the same operating system as your target EDM Clients.








Packager's dialog boxes guide you through the process of packaging an application. You can complete a session simply by specifying your information in the dialog boxes that appear.

The buttons shown in the table below appear on each of the Packager dialog boxes, and can be used to navigate through a Packager session.

To:	Choose:
Return to a previous Packager phase	
Proceed with the Packager session	
Exit the Packager session	

## Packager Phases

Packager's phases are shown in the table below. If you are completing an existing session, you can jump to a phase by choosing its icon.

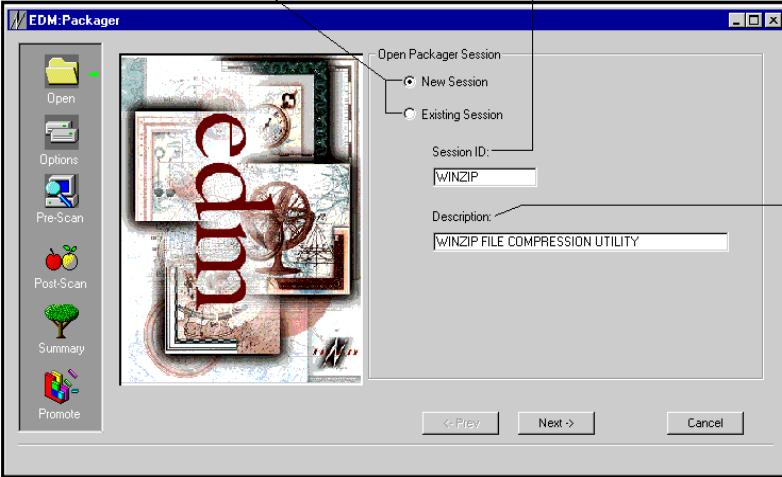
Phase	Description
	<b>See step 1</b> Assign an ID and description for your Packager session during this phase. If you want to return to this phase, you can choose the <b>Open</b> icon at any time.
	<b>See steps 2 – 4</b> Specify the system components you want to scan. During this phase, you also can backup files that may be altered by the software installation process. Packager creates the backup directory off your EDMSYS directory. The phase's second dialog box lets you specify the drives you need to scan.
	<b>See step 5</b> Packager scans the drives you specified and creates an image of them. When the image has been created, you will be prompted to install your software application.
	<b>See steps 6 – 8</b> After you've installed your software, you will double-click on this icon to enter Packager's Post-Scan phase. In NT 4.0, the icon resides in your system tray. In NT 3.51, the minimized icon is displayed in the program manager.
	<b>See steps 9 – 10</b> Packager conducts another scan and compares it to the first one. The differences between the two scans will be identified as the package's components.
	<b>See step 11</b> The three tabs of this phase can be used to edit the package components.
	<b>See steps 12 – 14</b> This phase lets you define how you will publish the package. This dialog box will also provide you with the ability to use the following options: <ul style="list-style-type: none"> <li>• Storage on an EDM Stager</li> <li>• File compression</li> <li>• Version numbering</li> <li>• NTFS property encapsulation</li> <li>• Instance identification according to CRC file name</li> </ul>

## Packaging an Application

When you open Packager, you enter its Open phase.

Select one of these two radio buttons to open a **New Session** or an **Existing Session**.

You can assign a **Session ID** by typing a name in this text box. The Session ID can be a maximum of 6 characters long.



By typing a description in this text box, you will be able to remember which software application you established the session for.

The example outlined in this chapter packages WINZIP (a file compression program). In this session, all default values were accepted, and no editing was performed in the Summary dialog box.

**Note:** Each Packager session should have a unique ID.

- 1 To continue, choose **Next->** The Options phase dialog box is displayed.
- 2 Use the dialog box to select the system components you want to scan and to backup files that the software installation will modify.

Use the:	To:
<b>What To Scan</b> group box	Specify the system components you want to scan—Registry, File System, and Desktop. By default, all options are pre-selected. If you decide not to scan a component, remove its check mark by selecting its check box. If the box is empty, the component will not be scanned.
<b>File Content Comparison</b> check box	Copy the files in the <b>File Contents To Be Scanned</b> list box to the backup directory during the Pre-Scan process. The backup directory is located off your EDMSYS directory, and uses the session ID as the first 6 characters of its identifier. You can void the use of a backup directory by selecting the <b>File Content Comparison</b> check box to remove its checkmark.

**Note:** By default your AUTOEXEC.BAT and CONFIG.SYS files are included in the backup directory, as well as any .INI files in your Windows system directory.

- 3 If you decide to use a backup directory, you can view the files it will contain in the **File Contents To Be Scanned** list box.

To:	A File:
Add	Type the file name in the <b>File Name</b> text box, and choose <b>Add-&gt;</b> or choose <b>Browse</b> and select the file you want from the dialog box that appears. When the name of the file you want is in the <b>File Name</b> text box, choose <b>Add-&gt;</b> .
Remove	Highlight it in the <b>File Contents To Be Scanned</b> list box, and choose <b>&lt;-Remove</b> .

- 4 Choose **Next->**. The second Options phase dialog box is displayed.
- 5 From the **What to** list box, select the drives your software application is being installed onto, and the drive your operating system is located on. Specific drive directories can be scanned.

**Note:** It is recommended that you do not select network drives due to the time they require to be scanned. Ensure that the drive your operating system is located on is scanned.

To:	A Drive In/From the Scan:
Include	Select it from the collapsible tree-view in the <b>What To</b> list box, and choose <b>Add-&gt;</b> . (For example, C: .)
Remove	Select it from the <b>Directories To Be</b> list box, and choose <b>&lt;-Remove</b> .

6 Choose **Next->**. The Pre-Scan dialog box is displayed.

7 To begin your Pre-Scan, choose **Begin Scan**. A snap shot of your system is created.

As each system component is scanned, its text changes color from its default of gray. The following table lists the status that each text color represents.


Color	Status
Green	Component was scanned successfully.
Red	There was an error scanning the component.
Yellow	You previously removed this component from the scan list.

**Note:** If an error occurs, the scan must be conducted again. The session can not be completed unless the scan is successful.

8 Choose **OK** from the message box that appears when the Pre-Scan is complete.

9 When you are returned to the Pre-Scan dialog box, choose **Next ->**. Another message box will appear. Choose **OK**, and install the software application.

10 If the software installation process requires you to reboot your machine, do so now.

11 After you install the software, continue with the session by choosing the Novadigm icon  from the system tray. The system tray is located in your display's lower left corner.



In NT 3.51, this minimized icon is located in the Program Manager.

12 If the software installation was successful, choose **Yes** from the message box that appears. The Post-Scan dialog box is displayed.

**Note:** If the software installation was not successful, choose **No**. You cannot proceed with a Packager session until you have successfully installed the software.

13 Choose **Begin Scan** from the Post-Scan dialog box. Packager will create another snap shot of your system, and compare it to the first one.

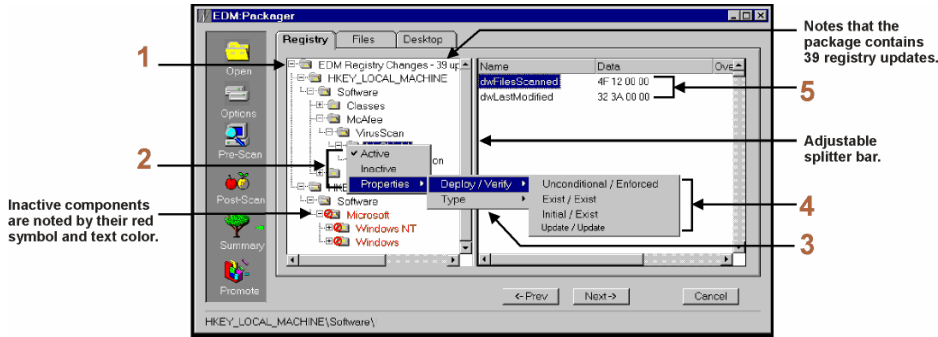
Packager assumes that the differences between the scans represent the components of the software you are packaging.

**Note:** As each system component is scanned, its text will change color as it did during the Pre-Scan. If there is an error, you must conduct the scan again.

14 When the Post-Scan is complete, choose **OK** from the message box that appears. The Post-Scan dialog box is displayed again.

This time, the dialog box contains a summary (which is easily identified by its blue text color) listing the differences found between the Pre-Scan and the Post-Scan. This summary is a list of your package's components.

15 To continue, choose **Next->**. The Summary dialog box is displayed.



**Note:** You are not required to perform any procedures within this dialog box.

16 This dialog box presents the components of your package in a tree view format so you can edit them. Choose a system component's tab to edit it.

Area	Description
1	Double click on a branch to expand it, or right click on the initial branch (i.e. <b>EDM Registry Changes</b> ) and select a menu option to <b>Expand All</b> or <b>Collapse All</b> of the branches.
2	By default all components are <b>Active</b> . Choose <b>Inactive</b> to exclude components from the package. Choose <b>Active</b> to include components in the package. Choose <b>Properties</b> to edit Active registry components. The Properties submenu is discussed in areas 3 and 4.
3	<b>Deploy/Verify</b> accesses deployment and verification option editing. See 4. <b>Type</b> deploys the package to either a <b>User</b> or <b>Machine</b> (default). <b>User</b> , deploys the component to a specific User ID. When the user logs on to a machine, only they will receive the component. <b>Machine</b> deploys the component to a machine so that any user logging onto it will receive the component. <b>Convert File to Update</b> (.INI files only) appends the .INI file instead of overwriting it upon delivery.
4	<ul style="list-style-type: none"> <li><b>Unconditional/Enforced</b> initially deploys the component and later verifies and enforces it</li> <li><b>Exist/Exist</b> deploys the component only if it does not already exist. (enforces existence only.)</li> <li><b>Initial/Exist</b> initially deploys the component and later enforces existence only. (Registry tab only.)</li> <li><b>Update/Update</b> initial and future deployments occur if the component is newer than the current one (.DLL files only).</li> </ul>
5	Change a registry value by double clicking on it in the name column. Type the new value in the <b>New Value</b> text box of the dialog box that appears. (Registry tab only.)

17 To continue, choose **Next->**. The Promote dialog box is displayed.

18 You can edit how your package is published in the **Promote Options** group box. The text boxes below this group box define where your package is published in the database.

### Promote Options Group Box

To:	Select the:
Store the package on a Stager	<b>Dataless Promote</b> check box.
Encapsulate the component's NTFS properties	<b>Encapsulate</b> check box.

Use the generated CRC name as the Instance name.(i.e. WINZIP_FF96A21)	<b>CRC Filename</b> check box.
Compress the component's data	<b>Compress</b> check box.
Assign a version number to the package	<b>Use Versioning</b> check box, and type the number in the text box provided.

## Publish Location Options

Type the:	For the Package in the:
Domain	<b>Resource Domain</b> text box. i.e: SYSTEMX
Instance prefix	<b>Instance Name Prefix</b> text box. i.e: WINZIP_W95
Service	<b>Service Name</b> text box. i.e: WINZIP_95
Stager location	<b>Stager</b> text box. (Not available at this time)

**Note:** If you specify a **Version Number** (Up to 7 alphanumeric characters), make sure that this number plus the **Instance Prefix** (Up to 19 alphanumeric characters), and the **Instance Name** (Up to 8 alphanumeric characters) does not exceed 32 characters. **VN + IP + IN ≤ 32 characters**

Your instance and service should hold the following naming standard:

Vendor\_Name\_Operating System i.e. MS\_Office\_W95

You can view the locations where components will be distributed to an EDM Client's desktop in the **Relocation Paths** list box.

If you want to change the relocation paths, you can use the EDM System Explorer to edit the Service after it is created by the Packager session.

**Warning:** If more than 12 paths are shown, you must keep Packager open, and use the System Explorer to add additional ZSVDIR variables to the Applications class.

- 19 Choose **Build Promote Object** to continue. To publish the package, choose **Promote** from the message screen that appears.
- 20 In the **EDM Security Checkpoint** dialog box, type your userid and password in the text boxes, and choose **OK**. Choose **OK** from the next two message boxes that appear.

Packager has published your package.

Now you can use the EDM System Explorer to connect EDM Clients to the package from within the database.

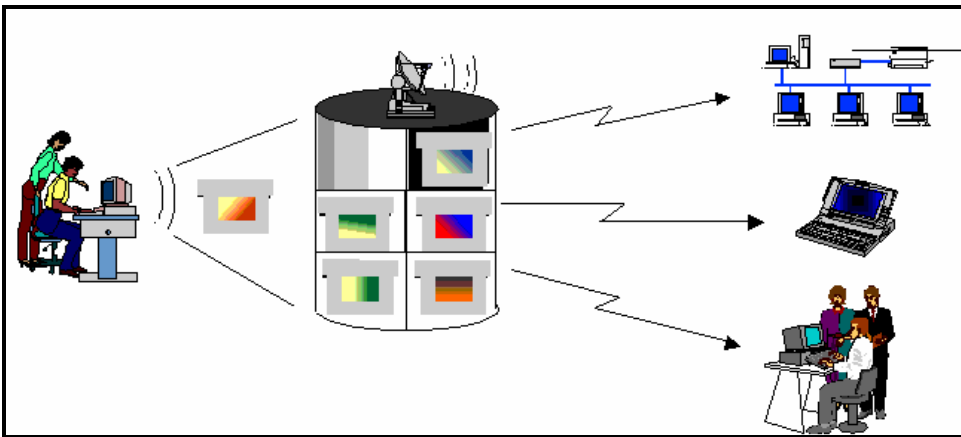


# Using EDM Publisher

EDM Publisher is used to publish packages to the database so they can be distributed to EDM Clients. Publisher requires you to manually identify the components of the data you are packaging.

## Making Packages Available

EDM gives you the power to distribute files and software programs throughout your network environment.



Use Packager or Publisher to publish your packages to the EDM database so they can be distributed to your EDM Clients.

## EDM Packager vs EDM Publisher

The table below explains when to use each tool to publish your packages.

Tool:	Used to Publish:	Benefit
Packager	Complete software applications, software updates, and other products with many dispersed, and unknown components should be published with Packager.	Automatically gathers all of the components of an application (including DLLs, INIs, and EXEs).
Publisher	Data files or software application upgrade files. When you use this tool, you need to manually identify the components required for the package.  If you are certain that you can identify all of the components for the package, you will want to use Publisher.	You must manually select the components you want to package and publish.

## Publisher Overview

---

EDM Publisher involves four main steps:

- 1 Select the components you want to package.
- 2 Edit the properties for the package and its components.
- 3 Edit the deployment properties for the package components.
- 4 Publish the package.

The steps in the following table refer to those in the section titled *"Using EDM Publisher"* in this chapter.

If You Want To:	See:
Open EDM Publisher	Step 1
Select the components for your package	Step 2
Edit the properties for the package and its components	Step 3
Edit the deployment properties for the packages components	Step 4
Publish the package	Step 5

### Before You Start

You should make sure that your database is current, and configured for your needs. During the Publisher session, information will be retrieved from the database concerning:

- Domains and Classes you've added to the database.

ZLOCLNT information (relocatability). Make sure that you have defined your relocation paths. EDM Publisher will read and list the ZLOCLNTs that are established on its database so you can select and apply them to the package you are publishing.

## User vs Machine

---

If you are packaging an application for EDM Clients that are using the Windows NT operating system, you can configure the component to deploy to either the Machine, or the User's ID on the machine.

When a component is deployed to a machine, all users will have access to it. If you deploy components to specific Users, they will only be accessible to that person's User ID.

### Additional Considerations

Applications can only be packaged and deployed to individual User IDs under the Windows NT operating system.

EDM Clients receiving components for specific User IDs are required to conduct the EDM Client Stand-alone installation. Refer to *Welcome to EDM*, Chapter 3, *"Installing the EDM Client for Windows NT and Windows 95."*

EDM Clients receiving components for a machine can conduct the EDM Client Stand-alone, or Multiple NT User installation.

If You Package Software On:	You Can Deploy To:
Windows 95	A Windows 95 EDM Client
Windows NT for a machine	A Windows NT User or Multi-User installed EDM Client.
Windows NT for a UserID	A Windows NT User installed EDM Client.

When you deploy components to a machine with multiple users, you must establish a User ID in the User class of the EDM database for the Machine, and the User (i.e., if there were two users on one machine, you would need three IDs). The components you deploy to the machine will only be received after the machine performs the EDM Client Connect Process.

## Using EDM Publisher

### 1 Opening EDM Publisher

Publisher is installed with the EDM Administrator, and can be opened from the Windows Taskbar by selecting **Programs, EDM Administrator** then **EDM Publisher**.

Type your EDM UserID and password in the text boxes of the **EDM Admin Security Information** dialog box, and choose **OK**. The EDM Publisher dialog box is displayed.

### 2 Component Selection

Select a tab (**File, Desktop, Registry**), and select the components you want to publish.

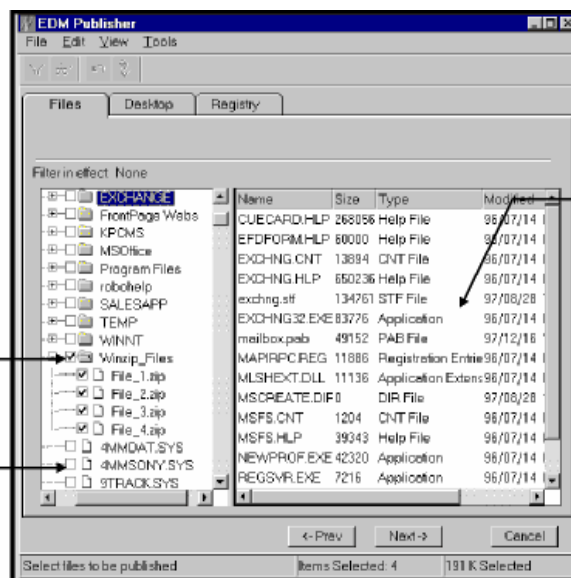
From the left view box, double click on the **Local Drives** or **Network Drives** branch, then select the drive letter where the components are located.

Select a	By:
Directory	Selecting the check box next to the directory you want to publish.
File	Double clicking on the directory the file is located in and selecting its check box.

Note: When you select a component, all of its subcomponents are also selected. The subcomponent can be removed by selecting its checkbox to remove the checkmark.

This component has been selected for packaging.

This component has not been selected for packaging.



This view area lists subcomponents, and details about the component itself. Items cannot be selected for packaging from this view area.

When you've finished selecting the components you are going to publish, choose **Next->**.

### 3 Setting Properties

Now you must define where the components will be published.

Expand the tree view until you see the components you want to publish. Select a file with your right mouse button, and choose **Set Properties** from the menu that appears.

You must define all of the information in the **Database** information tab view before you can publish your components. The other tab views shown are optional, and are discussed in “*Advanced Editing Options*” later in this chapter.

When you define the properties for a component, all of its sub-components are defined as well.

You Can:	By Selecting:
Define a Domain	A domain from the <b>Domain:</b> drop down list box. <b>For example: SYSTEMX.</b>
Define a Class	A class from the <b>Class:</b> drop down list box. <b>For example: ZRSOURCE.</b>
Define an Instance Prefix	A prefix from the <b>Instance Prefix</b> drop down list box, or typing one in. Your prefix should hold the following format: Vendor_Name_ Platform. i.e. MS_OFFICE_W95 <b>In this example: SALESAPP_W95</b> (Windows 95_No vendor_Sales application.)
Make instance names unique	The <b>Force instance names to be unique</b> check box. This is selected by default, and will warn you before any instance names are overwritten.

When you’ve defined all of the information required, choose **OK** to save the settings and return to the main dialog box.

### 4 Setting Locations

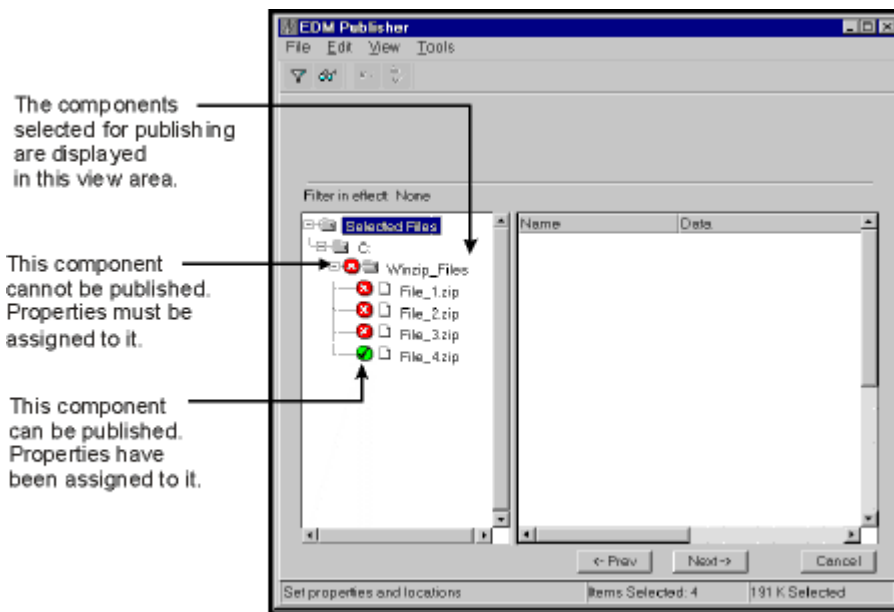
Now you must define where the package will be distributed to on an EDM Client’s desktop.

Returning to the main dialog box. Select the component with your right mouse button, and select **Set Locations** from the drop down menu that appears.

You must define the information in this dialog box before you can publish a component.

In the:	Drop Down List Box, Select a Location:
Client	To specify where the package will be deployed on a Subscriber’s desktop (drive and directory). This value is the ZLOCLNT value.
Manager	To specify where the package will be stored on the Software Manager.
Stager	To specify where the package will be stored on a Stager.

When you’ve finished specifying the information required, select **OK** to return to the main dialog box.



Notice that the icons next to the components you edited are now green. This means you have defined sufficient information for it to be published.

To proceed with a publisher session, choose **Next->**.

### 3 Publish the Package

You are now ready to publish the components to the EDM database.

The components should now have white icons next to them signifying that they are ready to be published.

Choose **Promote** to publish the components. After your package has been published, choose **Finish** to exit EDM Publisher.

Now, when you view the database through the EDM System Explorer, the components will be listed in the SYSTEMX domain's ZRSOURCE class.

You can connect all of the components to a common ZSERVICE to consolidate the application them. This will make connecting your users to the components a more efficient task.

## Summary

The steps you took to publish your application are reviewed in this section.

Steps	Description
<b>What do you want to Publish?</b>	Before opening Publisher you identified the data files, or in-house software application you wanted to publish. Remember that you must be able to identify all of an application's components if you are going to publish them with EDM Publisher.
<b>Select the Components</b>	You opened Publisher and manually selected the components you wanted to publish. (All of the selected components are referred to as a package.)
<b>Edit the Components</b>	You defined where the package would be published on the EDM database.
<b>Deployment Options</b>	You defined where the package would be deployed on an EDM Client's desktop.
<b>Publish the Package</b>	You published the package to the EDM database.
<b>What Now?</b>	Now that your package is published, you can use the EDM System Explorer to configure the database so that your users are connected to it.

## Advanced Editing Options

The additional tab views of the **Instance Properties** dialog box provide you with several options that enhance your control over how components are published and deployed. The table below lists the functionality of each tab.

Tab View	Functionality
Client Management	You can define the verification and delivery options for the package.
Data Options	You can compress the packages data, configure it to store on an EDM Stager, Encapsulate its NTFS properties, and retain it full path statement.
Client Behaviors	You can invoke the use of methods on the class you are storing the package in.

The following sections explain each tab view in more detail.

### Client Management

Verify options let you define when and if the package will be updated after its initial delivery. Verification occurs during the EDM Client Connect Process.

By default, the component is not verified, and it is distributed to both the UserID and the machine. This means that EDM will not check for the existence of the package after the initial delivery, and that anyone who uses the machine will have access to it.

#### Verification Options

To Verify:	Select the:
Using the EDM Manager default value	<b>Use the default specified on the EDM Manager</b> radio button.
Using the Date/Time stamp	<b>Verify Statistics Equal To:</b> radio button then select the <b>Date/Time stamp</b> check box. If the time stamps don't match, the package will be deployed again.
Using the File size	<b>Verify Statistics Equal To:</b> radio button then select the <b>File size</b> check box.
Using the CRC value	<b>Verify Statistics Equal To:</b> radio button then select the <b>Content (CRC Check)</b> check box
For existence only	Select <b>Check for existence only</b> check box. <b>Default.</b>

To void verification, ensure that the **No verification** check box is empty.

Delivery options let you deploy the package to a UserID or a machine and define a deployment priority.

### Delivery Options

To:	Select the:
Use the EDM Manager default priority number	<b>Use default priority (10)</b> check box if it does not have a check mark. <b>Default</b>
Set a new priority number for the component	<b>Use default priority (10)</b> check box to remove the check mark, and type the new priority number in the <b>Override Priority (01 – 99):</b> text box.
Make delivery of the component mandatory	<b>Mandatory</b> radio button.
Make delivery of the component optional	<b>Optional</b> radio button.
Deliver the components to a User ID	<b>User</b> radio button.
Deliver the components to a machine	<b>Machine</b> radio button.
Deliver the components to a User ID and a machine	<b>Both</b> radio button.

When you deliver the components to a User ID, only the specific subscriber will be able to use the component. All other users on the machine cannot use the component.

### Data Options

You can control how the data of the component is stored within the package. By default, the component's data is encapsulated.

If You Want the Component's:	Select the:
Data to be compressed	Compression setting you want from the <b>Compression Setting</b> drop down list box.
Data configured so it can be stored on an EDM Stager	<b>Promote instances without data</b> check box.
NTFS properties to be retained	<b>Encapsulate file/Folder properties within instances</b> check box. <b>Default</b>
Data encrypted	<b>Use Encryption for instances</b> check box.
Path statement to be retained	<b>Include subdirectories in filename</b> check box.

### Client Behaviors

You can execute methods on a desktop during the deployment of the package. Some examples are provided below.

If You Want To:	Type a Name In This Text Box	Example
Run a method before the component is deployed	<b>Resource Initialization Method:</b>	EDMKILL (Kills programs that are running before the package is delivered).
Specify an install method	<b>Method to Install Resource</b>	EDMCPAK (For text files)
Specify a de-install method	<b>Method to De-install Resource</b>	EDMDPAK (For text files)
Use a specific method to update an instance	<b>Instance Update Method</b>	
Use a specific method to update or add a file	<b>File Update/Add Method</b>	

You can also use locally developed methods in these text boxes.

## Global Default Values

EDM Publisher gives you the option to define default values. Unless other wise specified, these values will be used on all components selected for publishing.

The values defined here will be retained the next time you begin a Publisher session. You will need to erase the values if you don't want to employ them (leave text boxes blank).

Global default values are defined before you choose **Next->** after completing step 1.

The **Global Default** dialog box looks like the **Instance Properties** dialog box. You can open it by selecting **Edit** from the main menu then **Change Global Defaults**.

There are several differences from the **Instance Properties** dialog box. The table below notes these differences.

In This Tab View:	Note This Change:
Database Information	Only standard domains and are available as list options. Domains that you've added are only available in the Instance Properties dialog box.
Client Behaviors	No method text boxes have been completed for you.

## Summary

---

The steps you took to complete a Packager session are reviewed in this section.

Steps	Activity
What do you want to Publish?	Before opening Packager, you identified the software application you wanted to publish.
Identify the Components	You began a Packager session, and configured its scanning process to identify the components of the software you wanted to publish.
Edit the Components	You edited the package component properties in the summary dialog box.
Edit the Package	You edited the package properties to define where it would be transferred to in the database, and how its data would be stored.
Publish the Package	You published it to the EDM database.
What Now?	Now that your package is published, you can use the EDM System Explorer to configure the database so that your subscribers are connected it.



# The EDM Client Explorer

This chapter explains how an administrator can use the EDM Client Explorer to view, edit, and create local objects on an EDM Client desktop, or objects configured for other EDM Client desktops in the enterprise.

**Terminology:** This chapter uses the terms *heap* and *multi-heap object*. A *heap* is another name for an instance. The elements you want to manage in your enterprise desktop environment are represented as instances. A *multi-heap object* is an object with multiple instances.

## Platform Availability

---

The EDM Client Explorer is available on the following EDM Client platforms:

Platform Availability				
DOS	<input type="checkbox"/>		NCR MP/RAS	<input checked="" type="checkbox"/>
Windows	<input checked="" type="checkbox"/>		Sun OS	<input type="checkbox"/>
Windows NT	<input checked="" type="checkbox"/>		Unix AIX	<input checked="" type="checkbox"/>
Windows 95	<input checked="" type="checkbox"/>		Unix HP-UX	<input checked="" type="checkbox"/>
OS/2	<input checked="" type="checkbox"/>		Unix Solaris	<input checked="" type="checkbox"/>
Macintosh	<input checked="" type="checkbox"/>		UnixWare	<input type="checkbox"/>
NetWare	<input type="checkbox"/>			

## The EDM Client Explorer At A Glance

---

The EDM Client Explorer is one of several powerful EDM tools that allows an administrator to view and manage objects in an enterprise from the desktop.

The EDM Client Explorer is installed automatically with the EDM Administrator. A user with the EDM Administrator installed can use the Client Explorer as a diagnostic utility to view local objects on the desktop, and if desired, to edit existing objects, or create new objects.

If you have multiple EDM Clients that rely on a single file server in your EDM environment, you can use the Client Explorer to edit objects stored on that file server.

The EDM Client Explorer can also be used to edit objects on other EDM Client desktops. For diagnostic purposes, you can view and edit managed objects configured for other EDM Clients to which you are connected on a local area network (LAN).

**Warning:** EDM screen *objects* can be viewed with the EDM Client Explorer, however, the actual screens are not displayed. EDM screens must be created and edited by using the EDM Screen Painter. Be careful not to corrupt the variables of screen objects while viewing them with the EDM Client Explorer.

The EDM Client Explorer is supported on each platform that supports the EDM Administrator—that is, Windows, Windows NT, Windows 95, OS/2, Macintosh, and Unix.

## Before You Use the EDM Client Explorer

The EDM Client Explorer is a powerful tool for viewing, querying, and adding client-based objects. These interactions directly impact the configuration of the desktops in your client/server network. Before editing objects with the EDM Client Explorer, make sure that you understand the EDM-managed software configuration currently defined for the target EDM Client desktop.

To help you use the EDM Client Explorer as a diagnostic tool, this chapter provides information on the following tasks:

- Opening the EDM Client Explorer, and understanding how it interacts with local EDM objects.
- Adding new EDM objects.
- Viewing and editing local objects that reside in the directory where the EDM Client or EDM Administrator is installed.
- Switching to another drive/directory to view and edit additional local objects on your EDM Client desktop, or on another LAN-attached EDM Client desktop.
- Sorting objects, and searching on variables.

Finally, if you use symbolic substitution to represent the value of a variable, you can resolve the value in the EDM Client Explorer, and view the results of the resolution before saving it.

## Editing Objects Created on Other Platforms

Objects created on platforms that utilize big endian ordering can be viewed and edited only on big endian platforms. Operating systems that support the big endian method include: Mac OS, Unix AIX, Unix HP-UX, Unix Solaris, and Unix Sun OS.

Objects created on platforms that utilize little endian ordering can be viewed and edited only on little endian platforms. Operating systems that support the little endian method include: DOS, Windows, Windows NT, Windows 95, OS/2, NetWare, Unix NCR MP/RAS, and UnixWare.

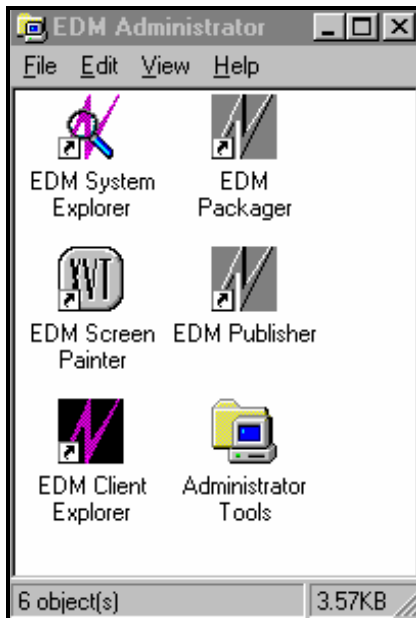
Objects Created in	Can Be Edited With the EDM Client Explorer For								
	Windows	Windows NT	Windows 95	OS/2	Macintosh	NCR MP/RAS	Unix AIX	Unix HP-UX	Unix Solaris
DOS	✓	✓	✓	✓		✓			
Windows	✓	✓	✓	✓		✓			
Windows NT	✓	✓	✓	✓		✓			
Windows 95	✓	✓	✓	✓		✓			
OS/2	✓	✓	✓	✓		✓			
NetWare	✓	✓	✓	✓		✓			
Macintosh					✓		✓	✓	✓
NCR MP/RAS	✓	✓	✓	✓		✓			
Unix AIX					✓		✓	✓	✓
Unix HP-UX					✓		✓	✓	✓
Unix Solaris					✓		✓	✓	✓
Unix Sun OS					✓		✓	✓	✓
UnixWare	✓	✓	✓	✓		✓			

## Opening an EDM Object

As mentioned, the EDM Client Explorer is supported on all GUI platforms that support the EDM Administrator—Windows, Windows NT, Windows 95, OS/2, Macintosh, and Unix.

### ➤ To Open an Existing EDM Object, or to Create a New Object:

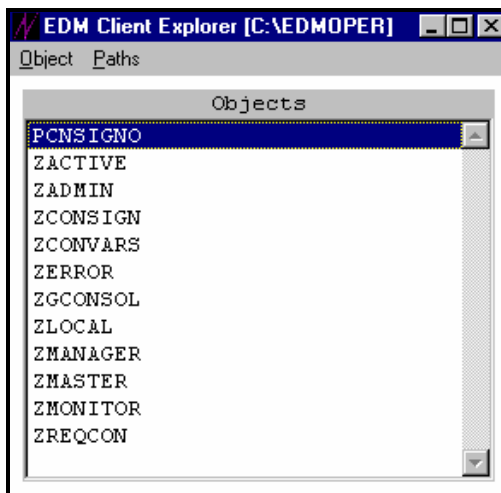
- 1 Choose the **EDM Client Explorer** icon from the EDM Administrator folder.



**Note:** The Client Explorer icon is only displayed on EDM Clients with the EDM Administrator installed. To run the EDM Client Explorer from a command line, or from an EDM Client desktop that does not have the EDM Administrator folder, execute EDMOBJED.EXE, which is located in your EDM Client directory.

The object list window is displayed.

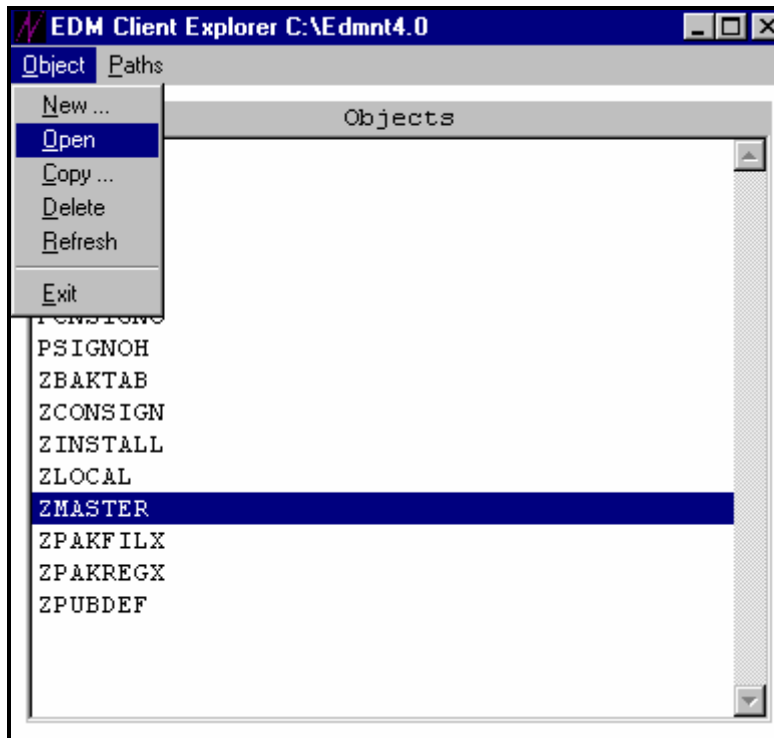
### Object List Window



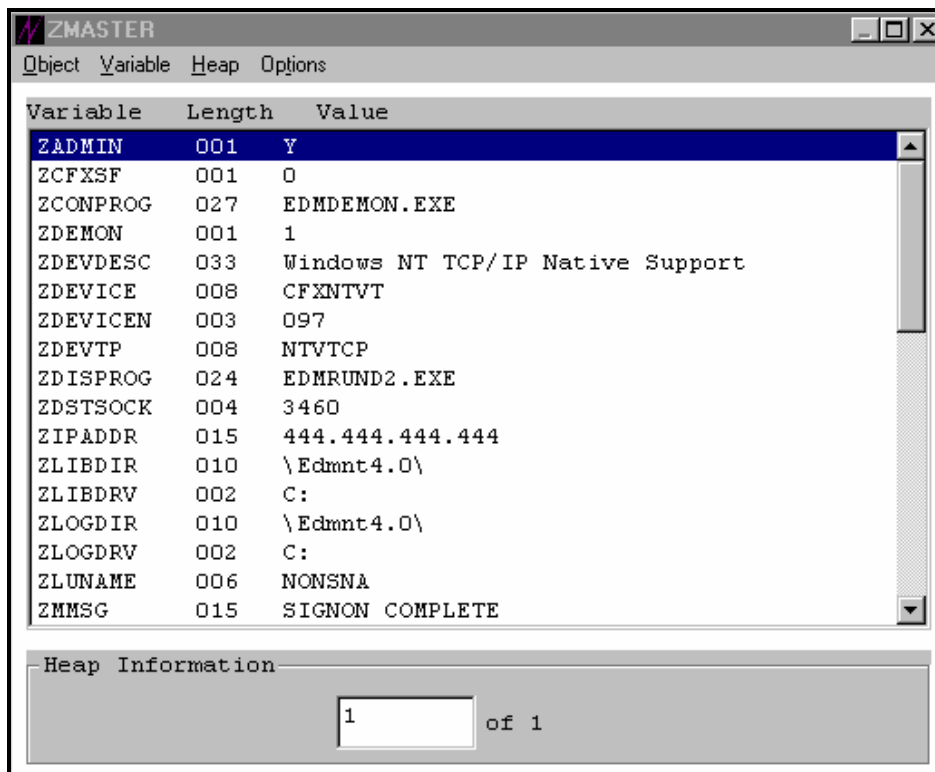
All of the local EDM objects found in the EDMLIB directory are displayed in the scrollable object list window.

**Notes:** If you have EDM objects stored in a directory other than the EDM Client directory, you can view them by choosing **Change EDM Object Path...** from the **Paths** menu. You can then switch the directory path to view these additional objects. For more details, see “*Changing Your Drive\Directory Path*” in this chapter.

- 2 Select an object from the list.



Choose **Open...** from the **Object** menu to display a list of the object variables associated with the selected object. In this case, the ZMASTER object variables are displayed.

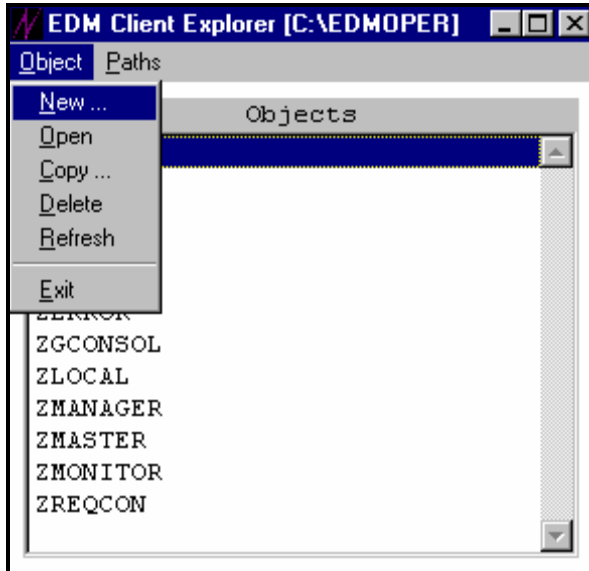


See “EDM Client Explorer Object Menu Map Reference” in this chapter for a description of each editing command.

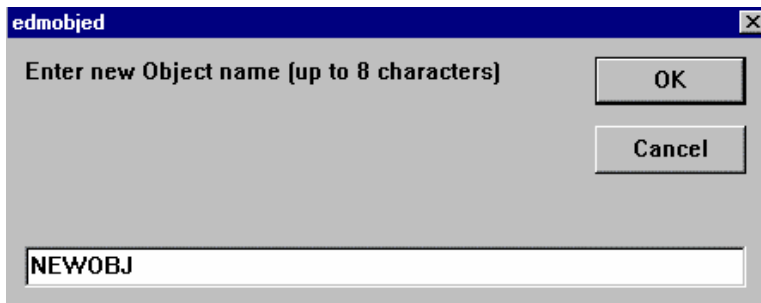
## Creating a New EDM Object

➤ **To Create a New EDM Object:**

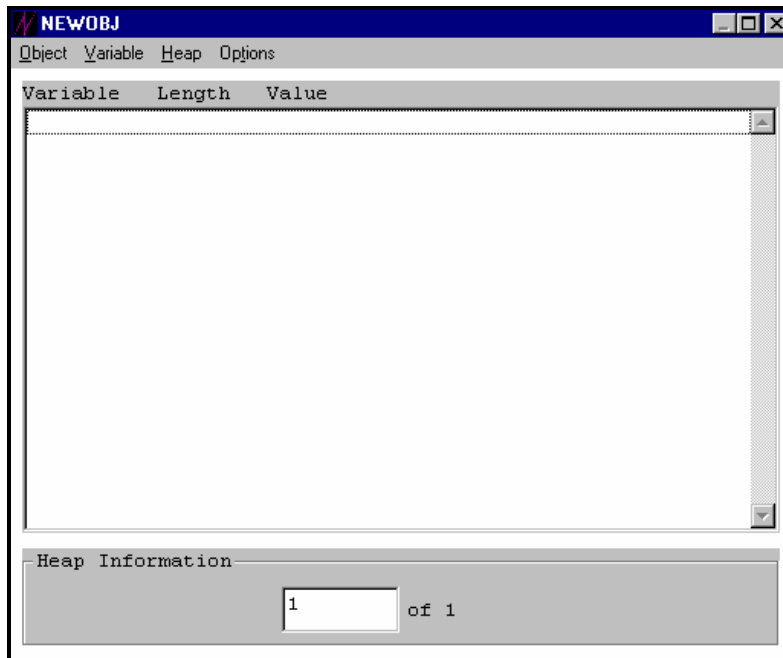
- 1 From the **Object** menu, choose **New**.



- 2 Specify an object name up to 8 characters in length. Do not specify an extension. In the example below we call the new object NEWOBJ.



**Note:** Objects that begin with Z are used by EDM. When naming an object that you create, we recommend that you choose a first letter other than Z.



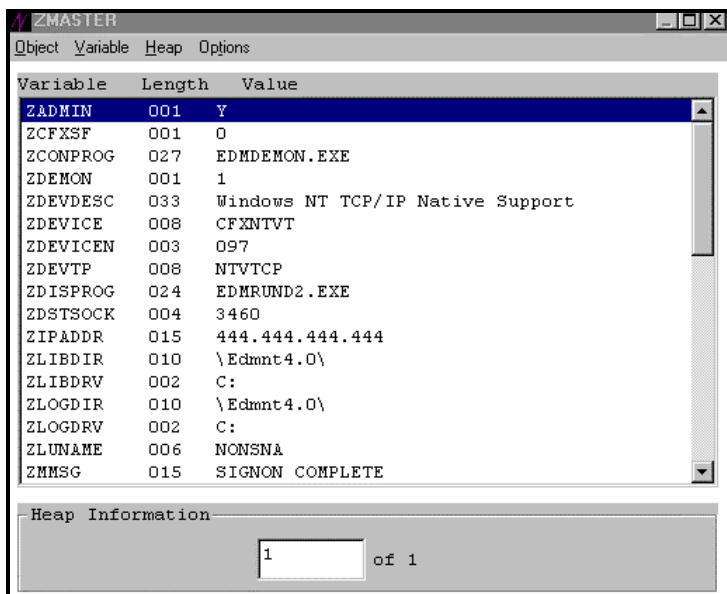
You now have a new object called NEWOBJ.

To add variables to it, see *"Adding Object Variables"* in this chapter.

## Viewing and Editing Object Variables

The Object View Window displays a list of the object variables associated with the selected object.

### Object View Window



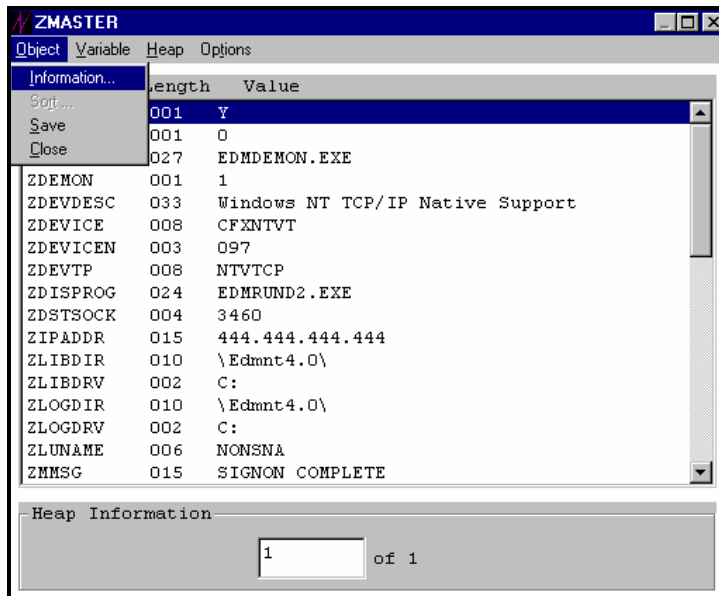
With the four menus on the Object View Window—**Object**, **Variable**, **Heap**, **Options**—you can perform a number of editing and viewing tasks.

**Note:** To save new or changed objects, you must use the **Save** option from the **Object** menu in the Object View Window.

## Displaying the EDM Object Information Screen

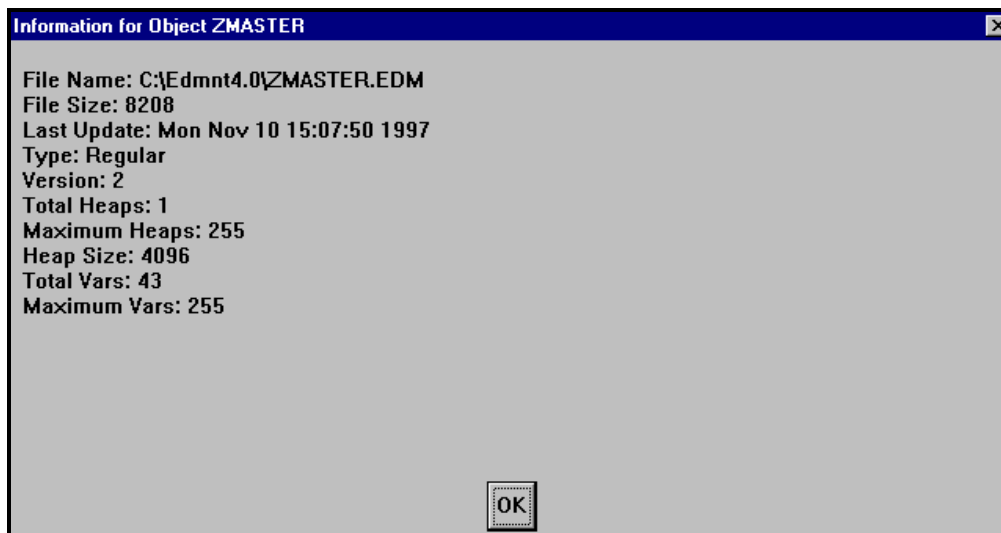
### ➤ To Display the Information Screen for an EDM Object:

- 1 Open an EDM object and select a variable.



- 2 Choose **Information...** from the **Object** menu to display the Object Information Screen.

The Object Information Screen is displayed:





The following table describes each entry in the Object Information Screen.

### Object Information Screen Entries

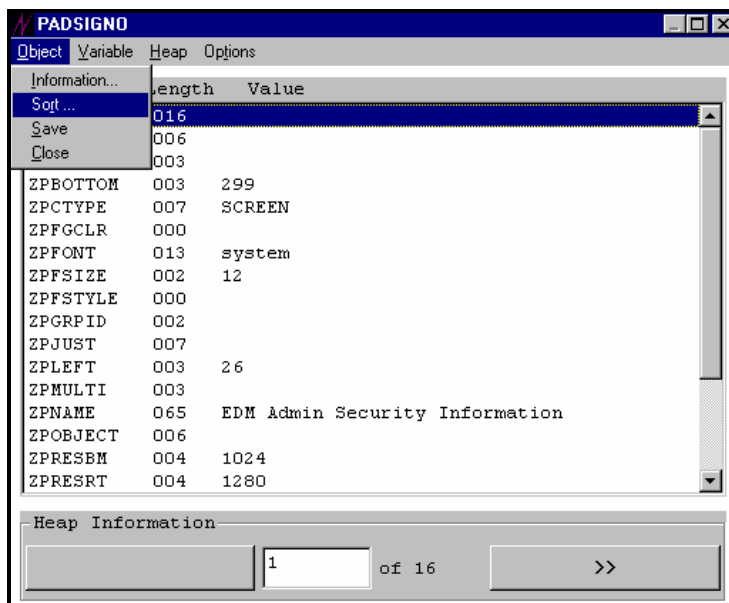
Entry	Description
File Name	The object file name and the drive/directory where it is installed on the EDM Client.
File Size	The size of the object.
Last Update	The last time the object was updated.
Type	One of two types: regular object, or screen object.
Version	The version number of the installed object.
Total Heaps	The number of instances defined for this object.
Maximum Heaps	The maximum number of instances allowed for this object.
Heap Size	Size of a single heap.
Total Vars	The total number of variables defined for this object.
Maximum Vars	The maximum number of variables allowed for this object.

### Sorting Object Heaps

The **Sort** option allows you to sort a multi-heap EDM object (rearrange the order of the heaps) by the values of a variable or several variables.

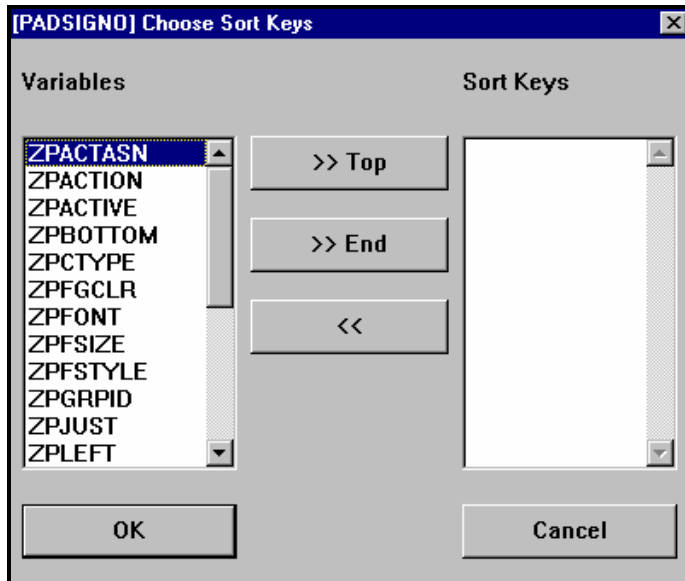
#### ➤ To Sort the Heaps of a Selected EDM Object:

- 1 Open an EDM object and select a variable.
- 2 Choose **Sort...** from the **Object** menu to display the variable sort dialog box.



**Notes:** The **Sort** option is only available for objects with more than one heap.

The Variable Sort Dialog Box is displayed.



- 3 Select the variable you want to sort by, and then choose **>> Top**. The variable name is moved from the **Variables** box to the **Sort Keys** box.

To sort heaps by more than one variable, select another variable from the Variables box and choose **>>End** to move it to the bottom of the list in the **Sort Keys** box.

To remove a variable from the **Sort Keys** box choose **<<**.

**Note:** The sorting process will make the top most variable in the Sort Keys box first and the bottom most variable in the Sort Keys box last.

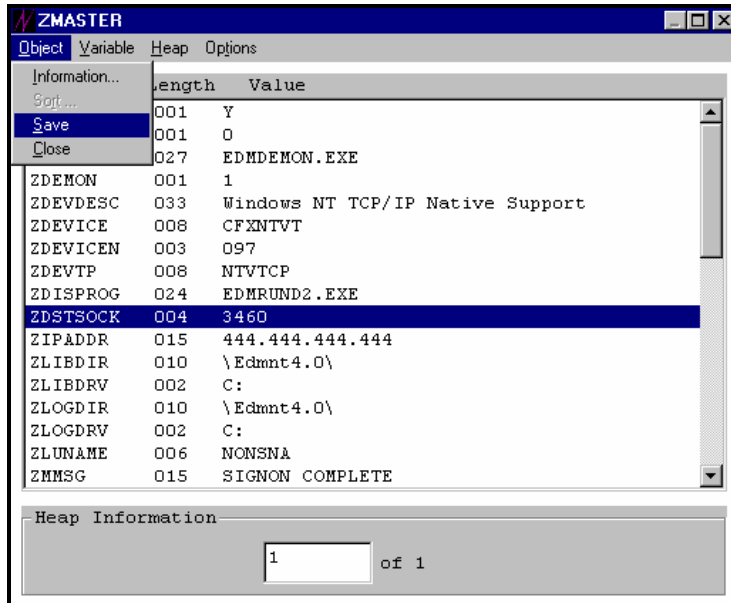
- 4 Once you have adjusted the variable list in the **Sort Keys** box, choose **OK**.

The heaps are rearranged according to the sort keys you specified, and changes are saved.

## Saving New or Changed Objects

### ➤ To Save a New EDM Object, or Make Changes to an Existing EDM Object:

While the object is open, choose **Save** from the **Object** menu.

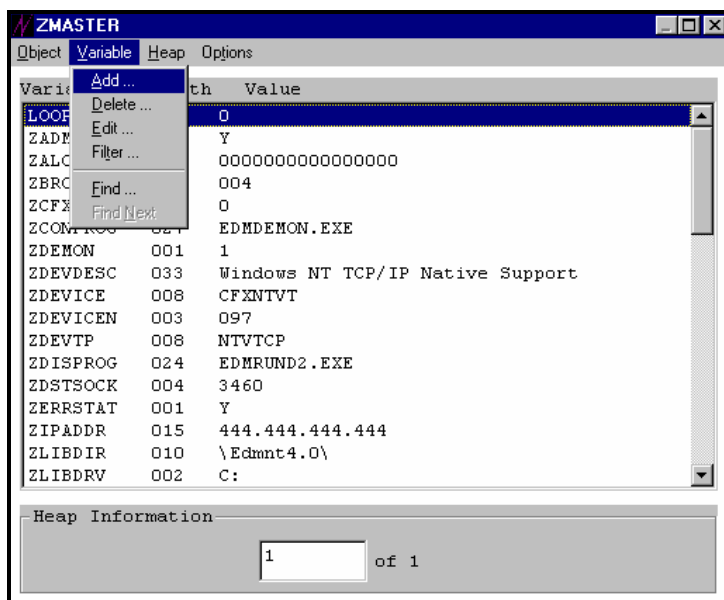


## Adding Object Variables

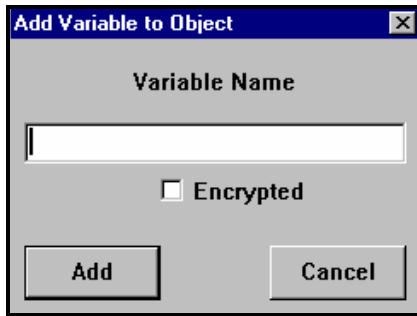
The **Add** option allows you to add a new variable to an EDM object. The EDM Client Explorer arranges all variables alphabetically.

### ➤ To Add a Variable to an EDM Object:

- 1 Open an EDM object and select a variable.
- 2 Choose **Add...** from the **Variable** menu.

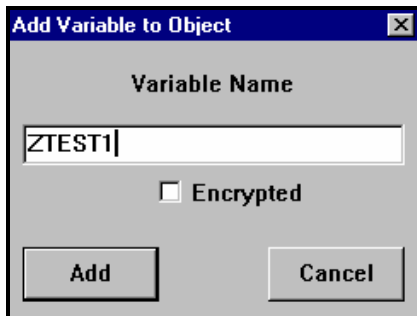


- 3 The **Add Variable to Object** dialog box is displayed.

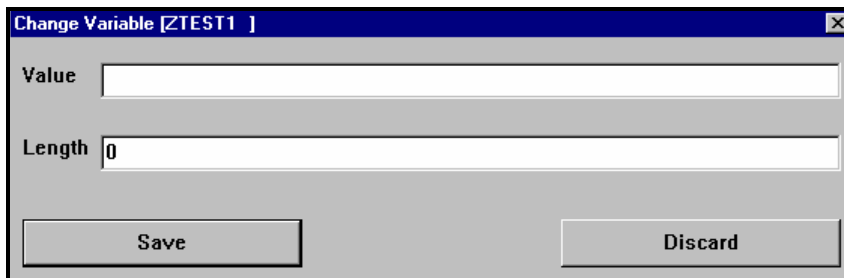


- 4 Enter a variable name, up to 8 characters in length, without an extension. (The ZTEST1 variable was added in the example). Select the **Encrypted** check box if you do not want the variable data displayed (for example, password variables).

Then choose **Add** to add the variable name to the list of variables.



The **Change Variable** dialog box is displayed.



- 5 Enter the value (variable data) in the **Value** text box, and choose **Save**.

After adding and saving a new variable, you can choose from the **Variable** menu to **Edit...** the variable again, or **Delete...** the variable.

**Note:** To save any additions or changes to an EDM object, you must choose **Save** from the **Object** pull down menu before exiting the EDM Client Explorer.

## Deleting Object Variables

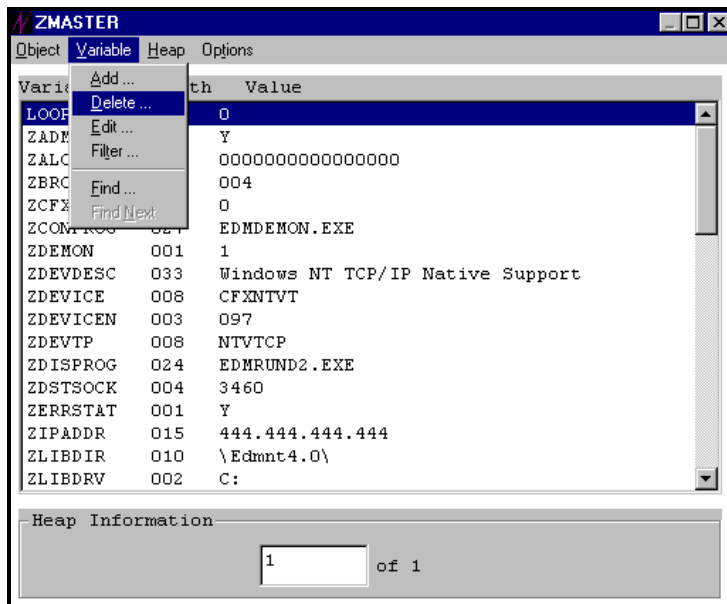
The **Delete** option allows you to delete a variable from an EDM object

**Warning:** The **Delete** option permanently deletes the selected variable from the object. Once a variable is deleted, it can not be recovered.

To replace a variable that was deleted, use the **Add...** option in the **Variable** menu. For details, see “*Adding Object Variables*” in this chapter.

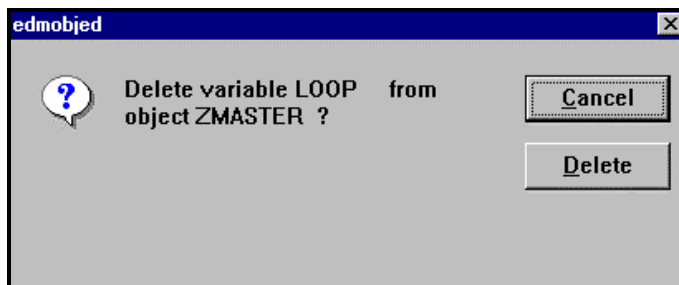
➤ **To Delete a Variable from an EDM Object:**

- 1 Open an EDM object and select a variable.



- 2 Choose **Delete...** from the **Variable** menu.

A dialog box is displayed.



- To confirm that you want to delete the variable, choose **Delete**.

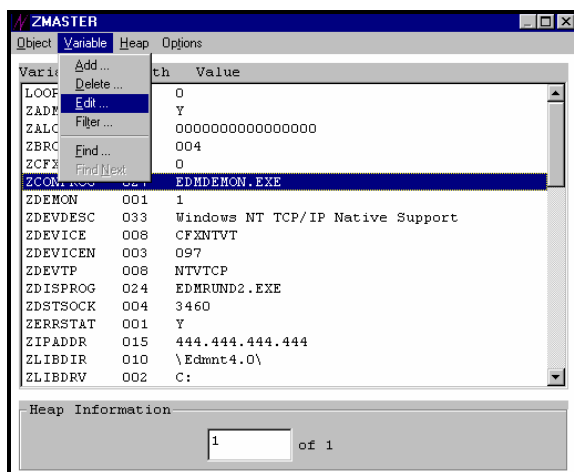
**Cancel** stops the deletion process and returns you to the Object View Window.

## Editing Object Variables

The **Edit** option allows you to change the value of the EDM object variable.

### ➤ To Edit a Variable Associated with an EDM Object:

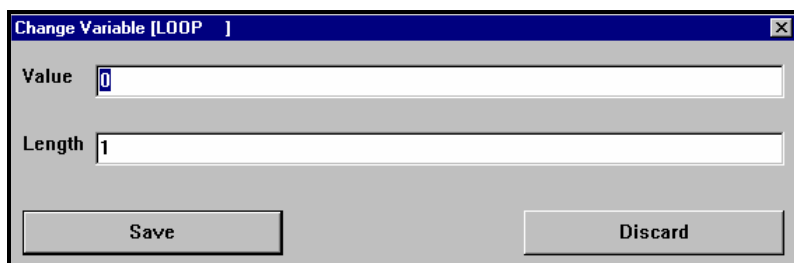
- Open an EDM object and select a variable.



**Note:** A faster way to edit a variable is to double-click the variable line.

- Choose **Edit...** from the **Variable** menu

The **Change Variable** dialog box is displayed.



- Enter the value of the variable, and choose **Save**.

**Discard** returns you to the list box without changing the current value of the object variable.

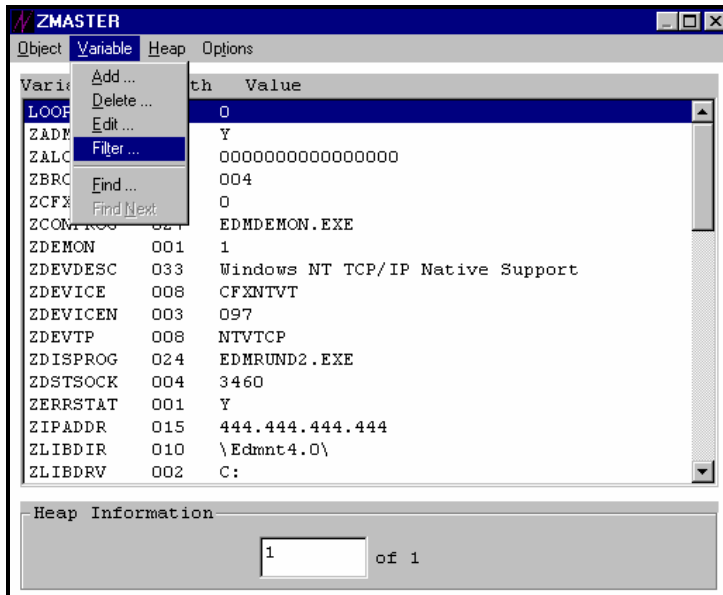
**Note:** You must choose **Save** from the **Object** pull down menu before exiting the EDM Client Explorer to save any changes to an EDM object.

## Filtering an EDM Object

The **Filter** option allows you to display a specific variable, or list of variables, that match a specified character string.

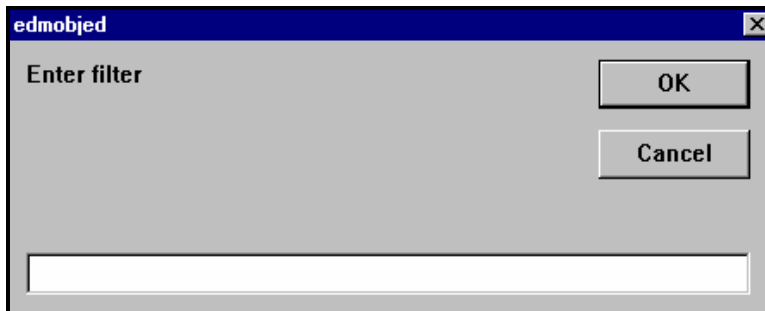
### ➤ To Filter an EDM Object:

- 1 Open an EDM object and select a variable.



- 2 Choose **Filter...** from the **Variable** menu.

A dialog box is displayed.

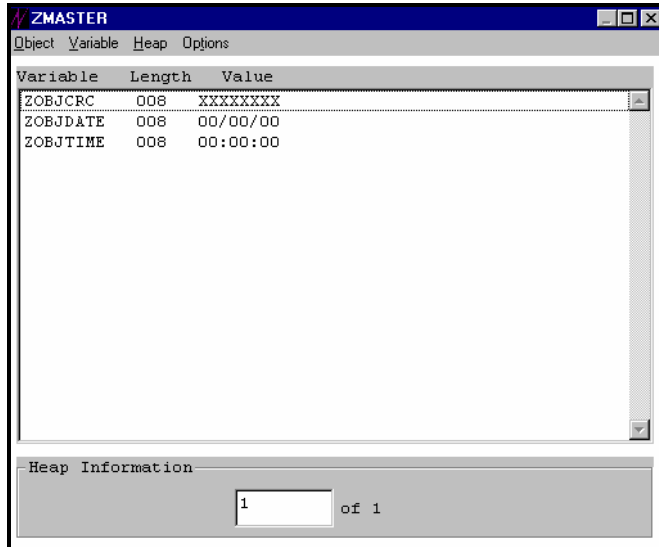


- 3 Type the character string that identifies the variable, or variables, you want to display, and then choose **OK**.

In the example, ZOBJ is used as the filter. Because the **Filter** option is case sensitive, use uppercase characters.



The matching variables are displayed in the list box:



**Note:** The Filter option is case sensitive. The character string case must match the case of the variable or variables.

## Searching On an Object Variable

The Find option allows you to search on a variable in a multi-heap object for a specified string.

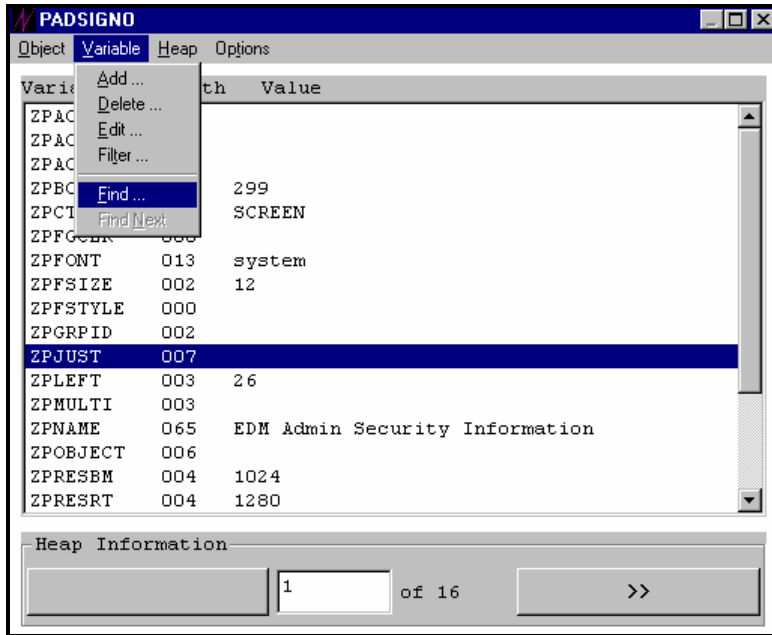
For example, if you have a multi-heap ZPJUST object, and you want to find the heap with the word “CENTER” in it, use the Find option to search on an object variable.

**Note:** The Find option searches only on the variable selected. It does not search through all the variables for the desired string.

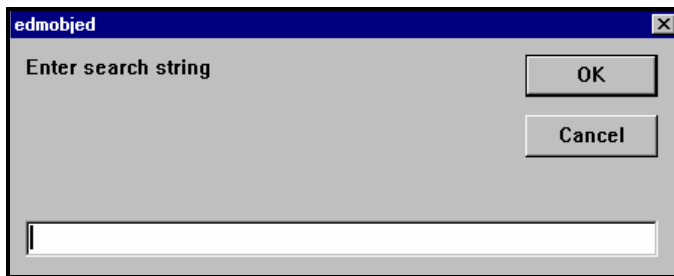
### ➤ To Search on a Variable:

- 1 Open an EDM object and select a variable.

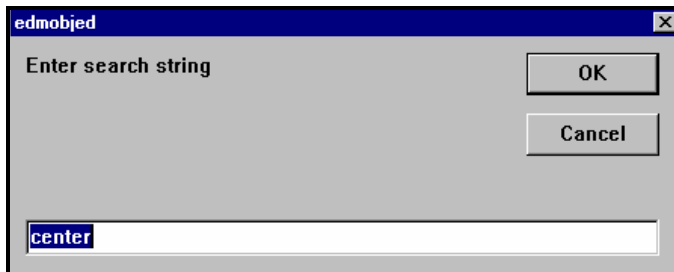




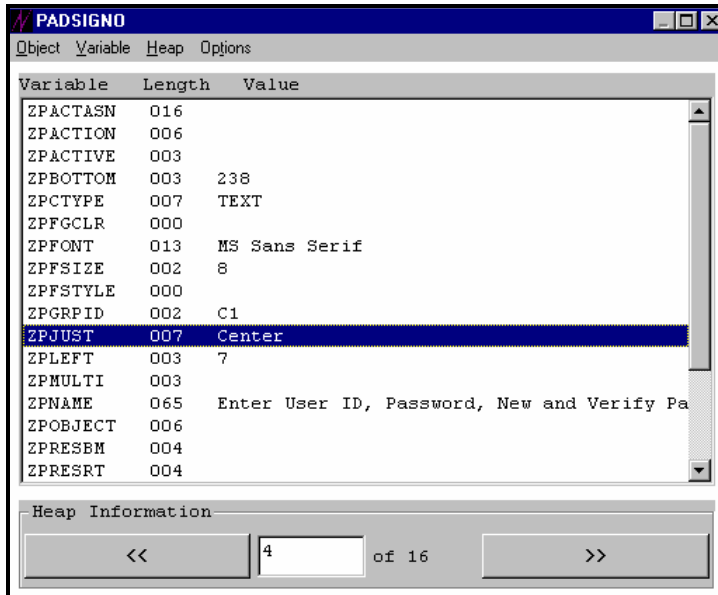
- 2 From the **Variable** menu choose **Find...**. A dialog box is displayed.



- 3 Enter the search text that identifies the character string you want to find.



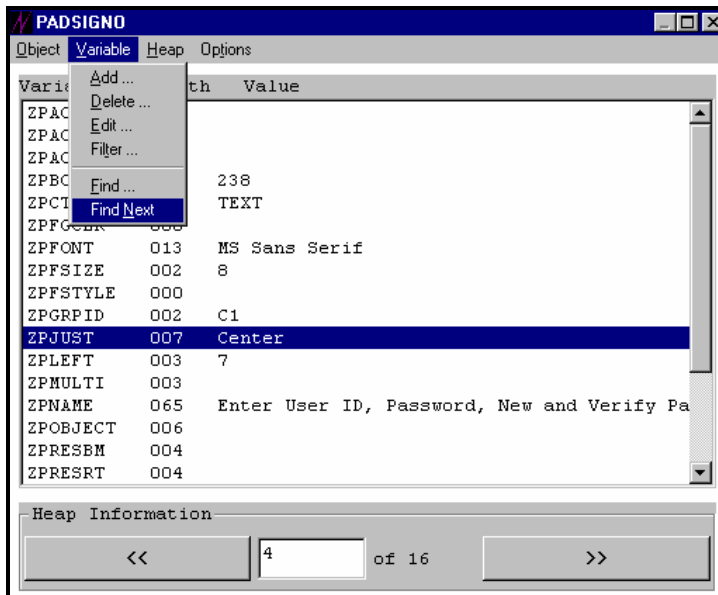
If the search text is found, it is displayed as a highlighted line in the list box.



In the above example, the first occurrence of the search text in the ZPJUST variable is highlighted. The **Heap Information** area indicates that the occurrence is in heap 4 of a 16-heap variable object.

**Note:** The Find option is NOT case sensitive.

- From the Variable menu choose **F**ind Next to find other occurrences (if any) in the remaining heaps.



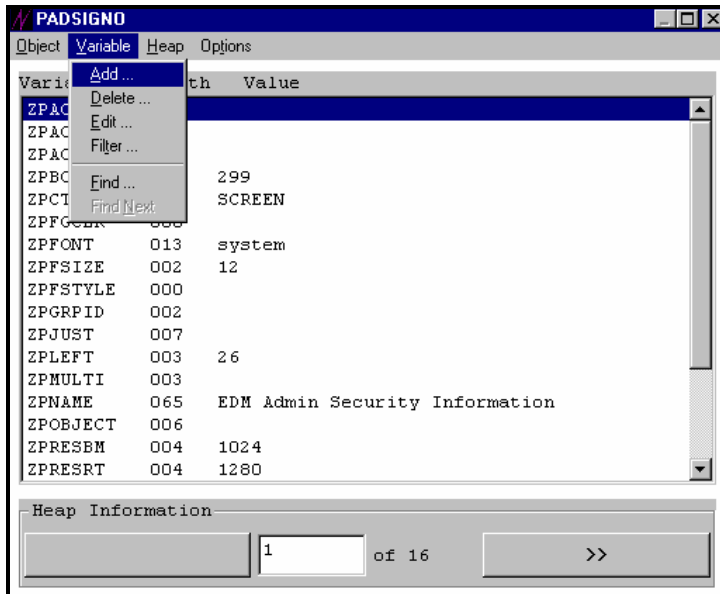
If there are no further occurrences of the searched text in the selected variable, the message “Not Found” is displayed.

## Adding a Heap to an Object

The **A**dd... option allows you to append a heap to the end of an EDM object.

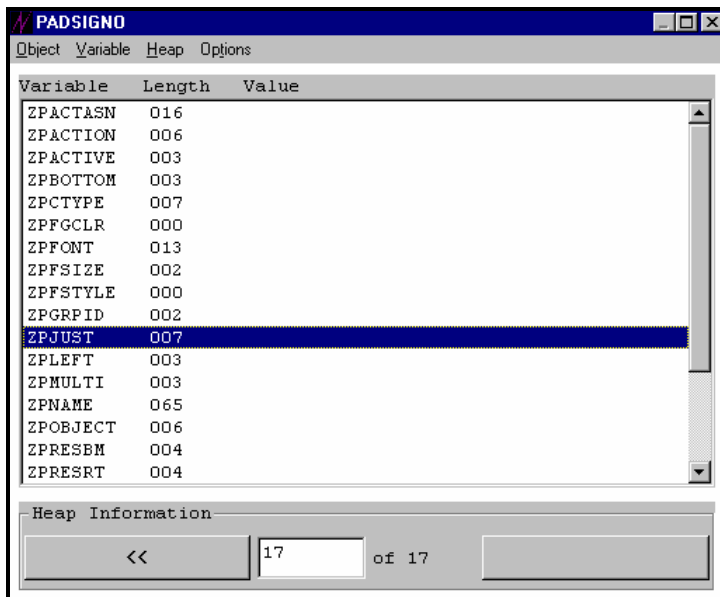
➤ **To Add a Heap to an EDM Object:**

- 1 Open an EDM object and select a variable.



- 2 Choose **Add...** from the **Heap** menu.

The heap is appended to the end of the object.



In the above example, the **Heap Information** area indicates that the new heap is number 17 of 17 heaps in the PADSIGNO object.

Notice that the variables in the new heap you have just added are the same, but their values are empty. To add a value to a variable in the new heap, select the variable you wish to edit, choose **Edit....** from the **Variable** menu, and enter the value of the selected variable.

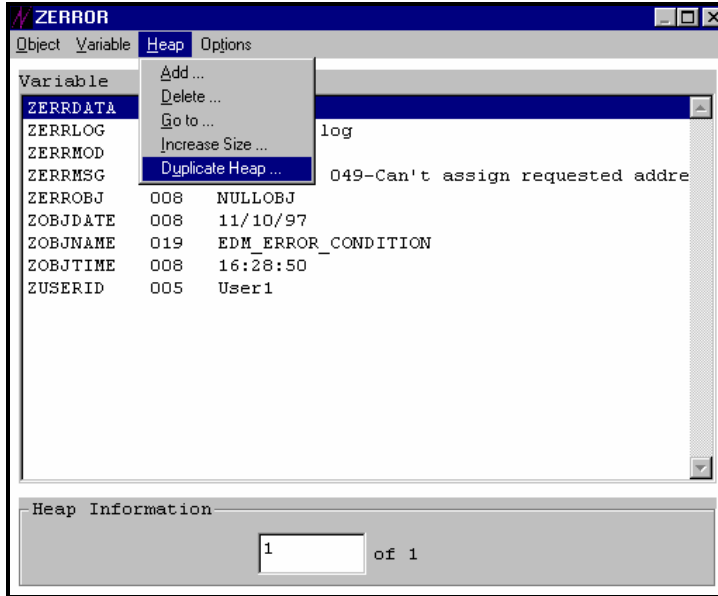
See “*Editing Object Variables*” in this chapter for a description of the **Edit....** option.

## Duplicating a Heap in an Object

The **Duplicate Heap...** option also allows you to append a heap to the end of an EDM object. However, the variables in this added heap have values that are identical to the heap you selected.

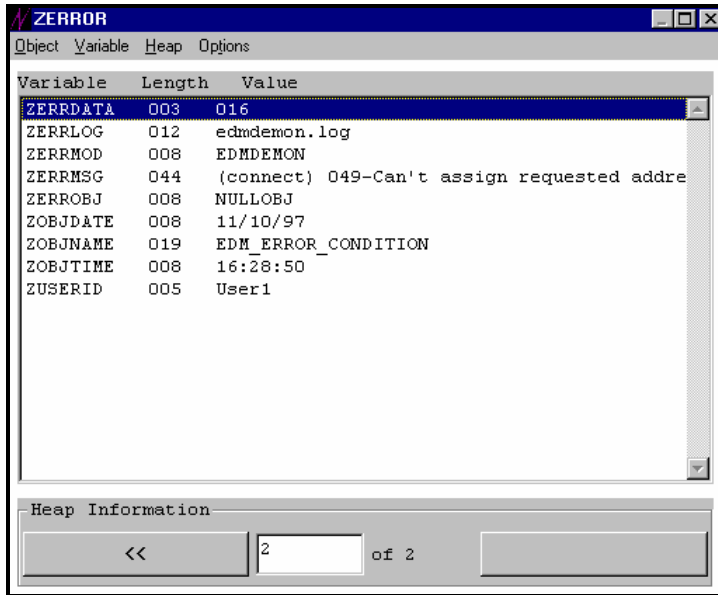
### ➤ To Duplicate a Heap in an EDM Object:

- 1 Open an EDM object and select a variable.



- 2 Under **Heap Information**, select the heap you want to duplicate by clicking >> (forward) or << (backward).
- 3 Choose **Duplicate Heap...** from the **Heap** menu. An identical heap is appended to the end of the object.

In the following example, the **Heap Information** area indicates that the new heap is number 2 of 2 heaps in the ZERROR object.



Notice that the variables in the new heap you have just added are the same, and that their values are identical to the heap you duplicated.

To modify the values, select the variable you wish to edit, choose **Edit...** from the **Variable** menu, and enter the value of the selected variable.

See *"Editing Object Variables"* in this chapter for a description of the **Edit...** option.

## Deleting a Heap from an Object

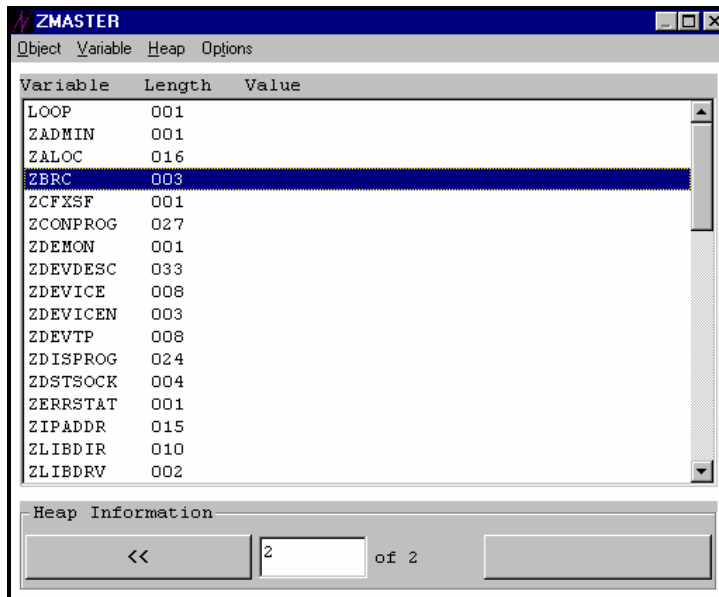
The **Delete...** option allows you to delete a heap from an EDM object.

**Warning:** The **Delete...** option permanently deletes the selected heap from the object. Once a heap is deleted, it can not be recovered.

To replace a heap that was deleted, use the **Add...** option in the **Heap** menu. For details, see *"Adding a Heap to an Object"* in this chapter.

### ➤ To Delete a Heap from an EDM Object:

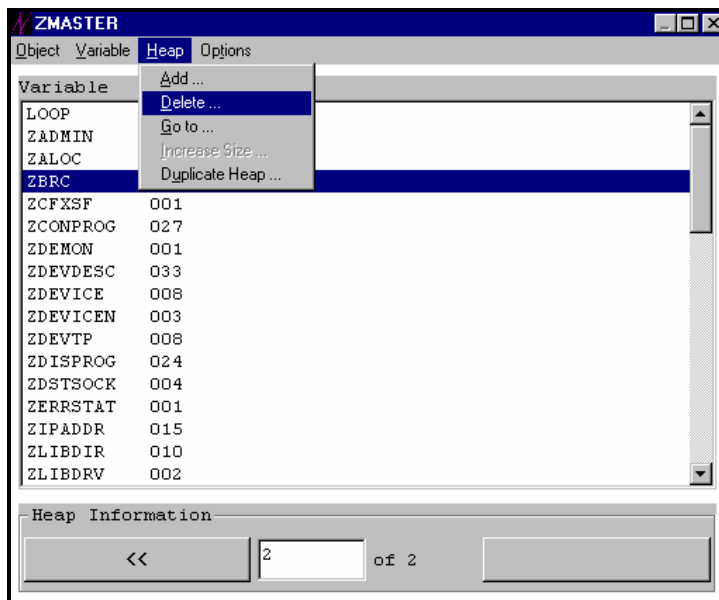
- 1 Open an EDM object and select a variable.



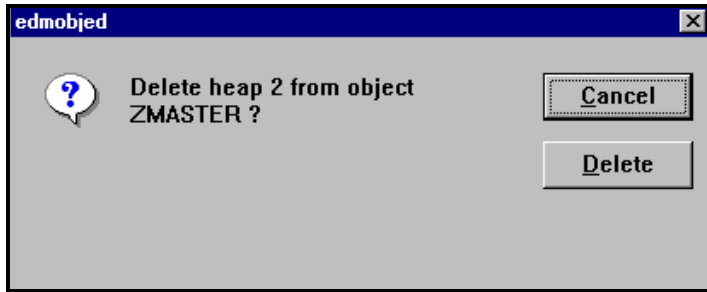
- In the **Heap Information** area of the list box, choose >> (forward) or << (backward) to select the heap you want to delete.

In the previous example, heap 2 is selected.

- Choose **Delete...** from the **Heap** menu.



A dialog box is displayed.



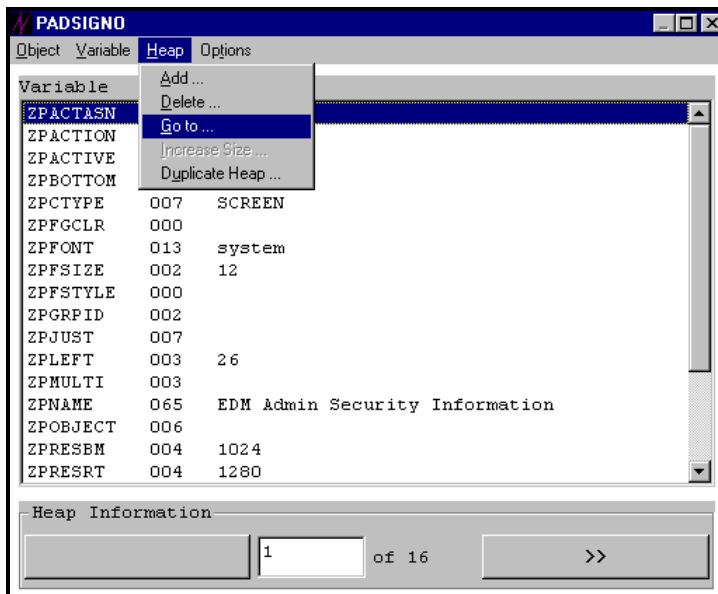
- 4 Choose **Delete** to confirm that you want to delete the variable.  
**Cancel** stops the deletion process, and returns you to the list box.

## Finding a Heap in an Object

The **Go to...** option allows you to find a specific heap number of an object.

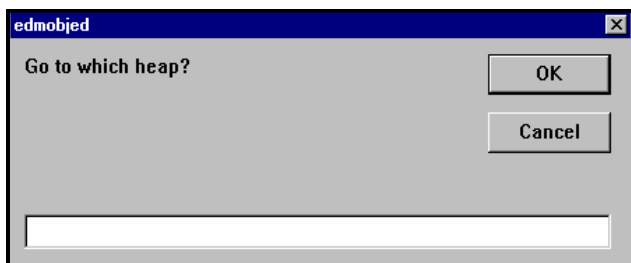
### ➤ To Go to a Specific Heap of an EDM Object:

- 1 Open an EDM object and select a variable.



- 2 Choose **Go to...** from the **Heap** menu.

A dialog box is displayed.



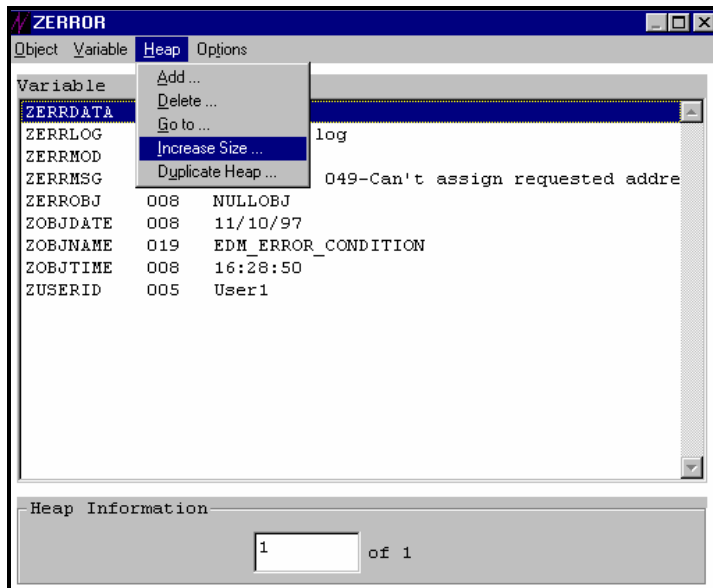
Enter the number of the heap you want to select, and choose **OK**, or choose **Cancel** to return to the Object View window.

## Increasing the Size of a Heap

The **Increase Size...** option allows you to increase the size of a heap. This option is only available in a single heap object.

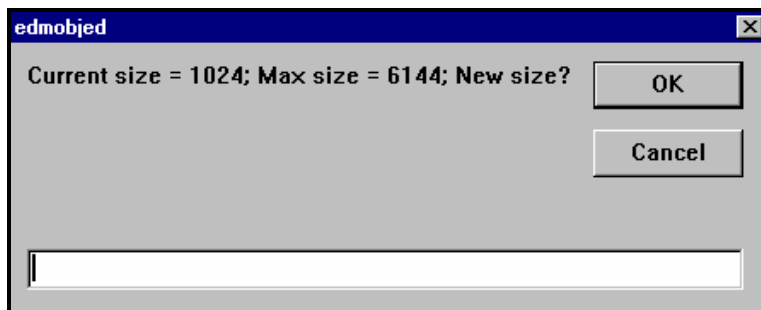
### ➤ To Increase the Size of a Heap:

- 1 Open an EDM object and select a variable.



- 2 Choose **Increase Size...** from the **Heap** menu.

A dialog box is displayed.



- 3 Enter the new size, which must be a numeral between the **Current size** and the **Max size**.

## Resolving Object Variables

If you use symbolic substitution to represent the value of a variable, you can resolve the value, and view the results of the resolution before saving.

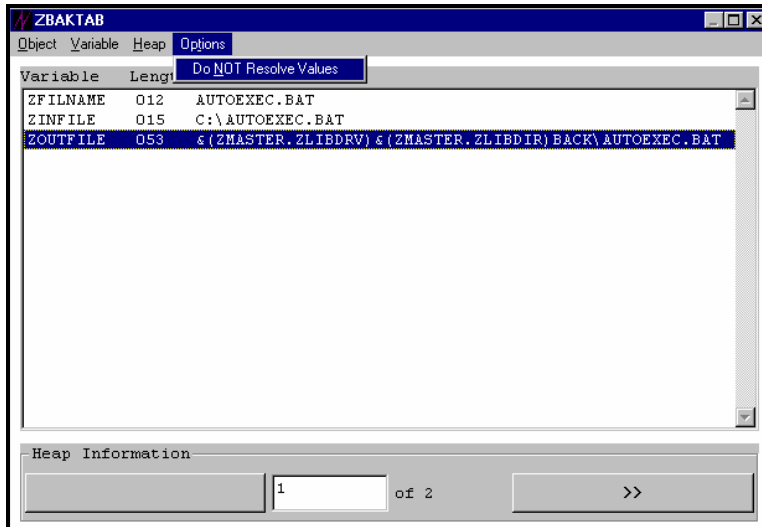


➤ **To Resolve the Value of a Variable that uses Symbolic Substitution:**

- 1 Open an EDM Object and select a variable.
- 2 Choose the **Options** menu to display the current status of all values that are expressed in symbolic notation.

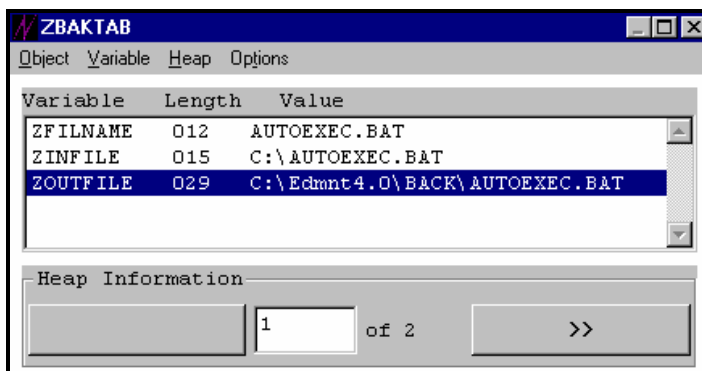
If the values are resolved, **Resolve Values** is displayed.

If the values are unresolved (as indicated in the example), **Do NOT Resolve Values** is displayed:



- 3 Choose **Do NOT Resolve Values**. The resolved values of the symbolic substitution appear in the value field of the variables.

In the example, &(zmaster.zlibdrv)& (zmaster.zlibdir)back\autoexec.bat resolves to the value displayed below.



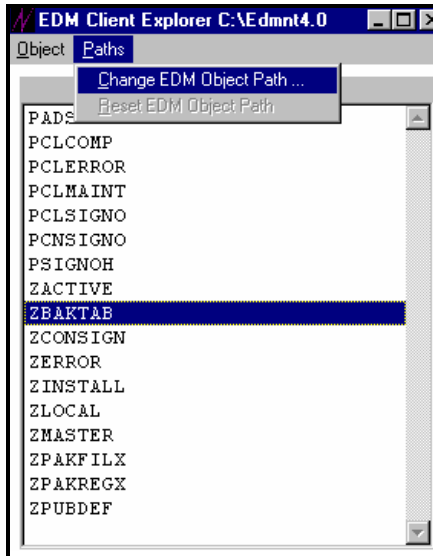
To toggle back to the symbolic notation for the values, choose **Resolve Values**.

## Changing Your Drive\Directory Path

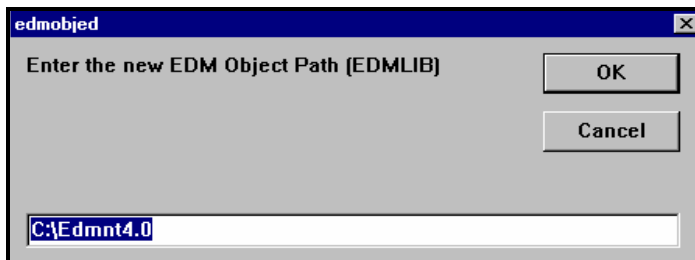
You can change your drive\directory path to view other EDM objects on your own desktop, or on another EDM Client desktop to which you are connected on a local area network (LAN).

➤ **To View EDM Objects that Reside on Another Drive\Directory Path:**

- 1 Choose **Change EDM Object Path...** from the **Paths** menu.



A dialog box is displayed containing the current path for the objects in the list box:



- 2 Enter the new EDM object path, and choose **OK**. All of the EDM objects found in this directory appear in the scrollable object list box.

**Cancel** returns you to the list box.

- 3 Choose **Reset EDM Object Path** to view objects in the EDM Client directory of your desktop.

**Note:** By changing the object path, you are only switching to a different directory to view, or possibly edit, additional objects. This command option does not change the directory on which the objects reside, or move objects to another directory.

## Working with Local Objects

When you open the EDM Client Explorer, the local EDM objects residing on your EDM installation directory appear in the scrollable object list box.

The following table contains a partial list of local objects that are normally present on a configured EDM Client desktop.

## EDM Client Local Objects

Object	Description
ZMASTER	Stores the information that uniquely identifies this EDM Client.
ZSERVICE	Keeps track of the applications for this EDM Client.
ZRSOURCE	Keeps track of the files that make up a service for this EDM Client.
ZADMIN	Stores information that results from a connection between the EDM Administrator and the EDM Manager (or the EDM Class Browser).
ZEMULDAT	Stores the variable information about the communication emulator software options that is available at the time that EDM Client software is installed.
ZERROR	Created by EDM Client processes whenever an error occurs during the processing of objects. Error-specific details are stored in variables, and reported to the EDM Manager.
ZINSTALL	Stores details about the installation components for this EDM Client, including single vs. multiple-user, directory, EDM Packager release level, hard drive ID, space requirements, and more.
ZLOCAL	Stores local processing information that must be kept between EDM Client executions.

## Working With Objects From Remote EDM Desktops

The EDM Client Explorer can be used to diagnose problems by viewing objects from other EDM Client desktops.

The following section summarizes the steps used by an administrator to view the ZRSOURCE and ZSERVICE objects located on a remote EDM Client desktop that is experiencing a problem.

### Problem Determination Scenario

After a successful EDM Client Connect Process, an EDM Client user discovers that no icons are displayed for a particular application.

- 1 The user communicates the problem to an administrator.
- 2 The administrator copies the ZRSOURCE and the ZSERVICE objects from the affected EDM Client desktop to a common server drive in the EDM environment network.
- 3 The administrator opens the EDM Client Explorer, and from the **Paths** menu, switches to the network server drive/directory to which ZRSOURCE and ZSERVICE were copied.

Both objects appear in the object list box.

- 4 The administrator opens the ZSERVICE object first, and the variables of ZSERVICE are displayed in the list box.

The **Heap Information** area of the list box displays 143 heaps in this multi-heap object.

- 5 To view the object status codes, the administrator uses the **Sort...** option to sort using the ZSVCCSTA variable.
- 6 The administrator uses the **Go to...** option from the **Heap** menu to view the object status codes, starting with the last heap in the object—heap 143.

(All higher status codes end up at the end of the object.)

- 7 The administrator chooses the << button in the **Heap Information** area to scroll backwards through the status values of the object.

The status codes indicate that a write error occurred during the EDM Client Connect Process for at least one resource that comprised a particular service.

Following the same procedure, the administrator then checks the ZRSCCSTA field status codes for the ZRSOURCE object.

These status codes indicate that no data for the resource could be found in the EDM Manager.

Based on this process, the EDM Administrator recommends that the resource be promoted to the EDM Manager to make sure that it resides in the EDM Manager.

The affected EDM Client then re-invokes the EDM Client Connect Process, and the icons are displayed, as expected, on the EDM Client desktop.

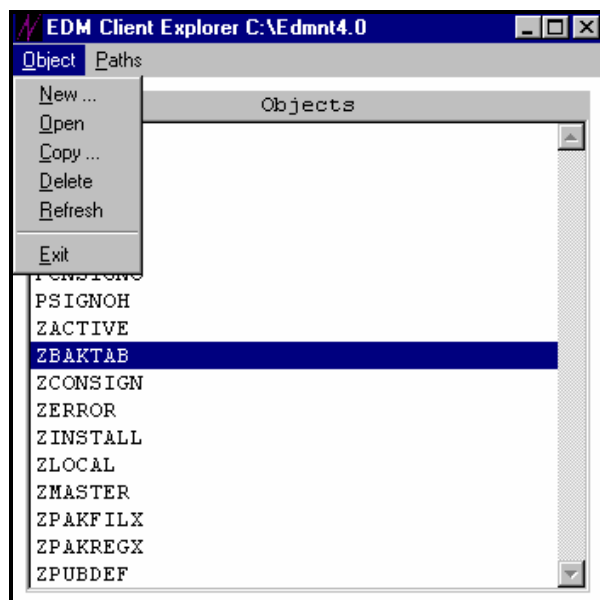
## EDM Client Explorer Object Menu Map Reference

---

### Object View Menu Map Reference

The **Object** menu and the **Paths** menu are displayed when you first launch the EDM Client Explorer.

### Object Menu Map



The **Object** menu provides commands that allow you to create new objects, display an object's variables, copy or delete an object, refresh a list of edited or new objects and exit the EDM Client Explorer.

**New:** Enables you to create a new object.

**Open:** Displays an object view window listing the variables associated with the selected object.

**Copy:** Allows you to copy an existing EDM object under a new object name. Displays a dialog box that allows you to enter the new object name.

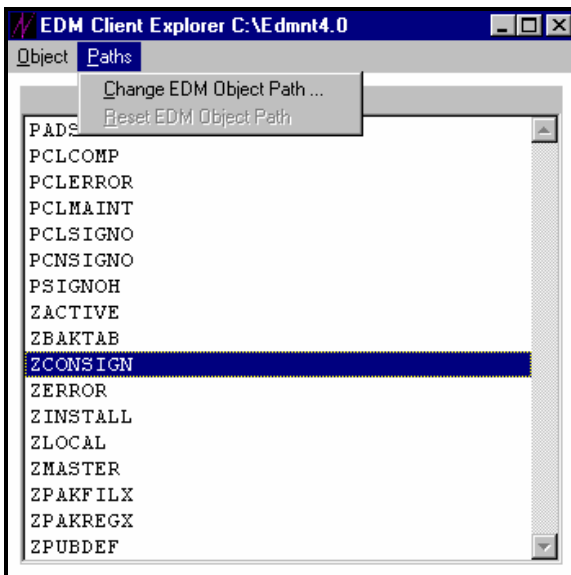
**Delete:** Deletes the selected object and the associated object variable. A dialog box allows you to confirm your selection before it is deleted.

**Refresh:** Refreshes the list of edited or new objects that have been saved before closing the object view window, or deleted from the object list box.

Refresh does not save edit changes. Changes are saved only by choosing **Save** from the **Object** menu of the object view window.

**Exit:** Exits the EDM Client Explorer, ending all object editing.

## Path Menu Map



The **Path** menu provides commands that allow you to change an EDM object path and then reset it.

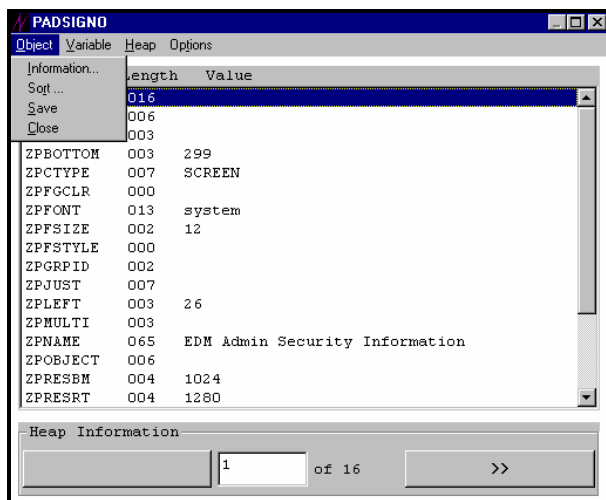
**Change EDM Object Path:** Changes the EDM object path (EDMLIB). When selected it prompts you for a new path.

**Reset EDM Object Path:** Resets the EDM object path to the previous value. This will not be an option until a change has been made to the EDM object path.

## Variable View Menu Reference

The Variable View Menu provides commands that allow you to manipulate variables.

## Object Menu Map



The **Object** menu of the Variable View Menu provides commands that allow you to display information about the object, sort the heaps of a multi-heap object, save an EDM object and close the Variable View Menu.

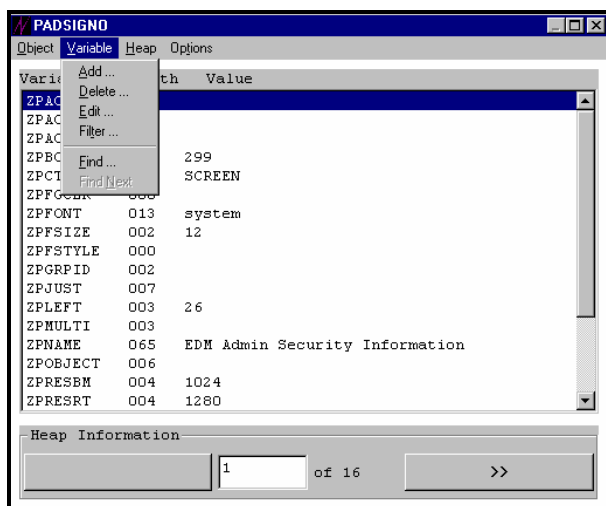
**Information:** Displays the Object Information Screen.

**Sort:** Sorts (rearranges) the heaps of a multi-heap EDM object by variable or by multiple variables.

**Save:** Saves an EDM Object.

**Close:** Closes the Variable View Menu.

## Variable Menu Map



The **Variable** menu provides commands that allow you to add, delete, edit and filter an EDM object variable and to search on a variable in a multi-heap object for a specified string.

**Add:** Adds a new variable to an EDM object.

**Delete:** Deletes a variable from an EDM object.

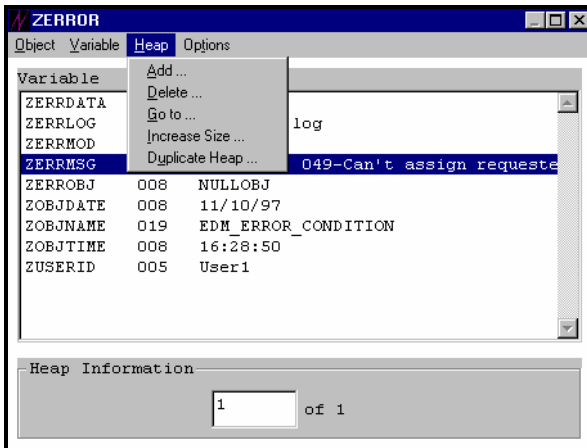
**Edit:** Allows you to change the value of an EDM object variable.

**Filter:** Displays a specific variable, or list of variables, that match a specified character string.

**Find** Searches on a variable in a multi-heap object for a specified string.

**Find Next:** Searches for additional occurrences (if any) of the string you specified in the **Find** command in the remaining heaps.

## Heap Menu Map



The **Heap** menu provides commands that allow you to add and delete heaps, and go to, increase the size of, and duplicate a specific heap within a variable.

**Add:** Appends a heap to the end of an EDM object..

**Delete:** Deletes a heap from an EDM object.

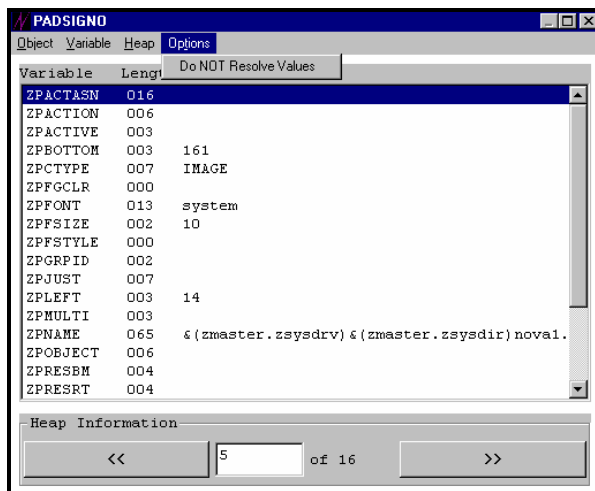
**Go to:** Finds a specific heap number of an object..

**Increase Size:** Increases the size of a heap by an amount that you specify.

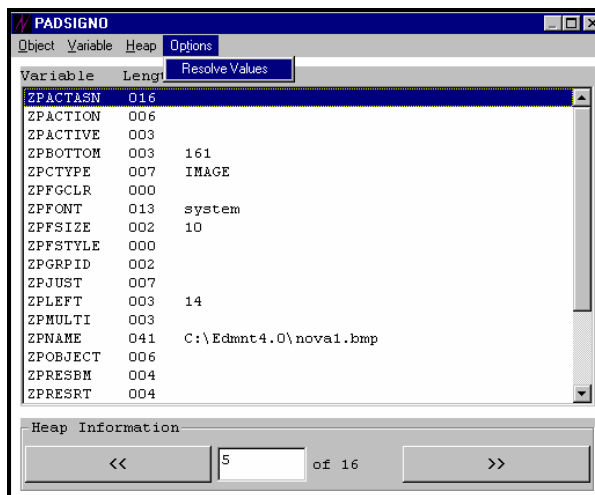
**Duplicate Heap:** Appends a heap to the end of an EDM object. Its variables are identical to the heap you selected to duplicate.

## Options Menu Map

If you use symbolic substitution to represent the value of a variable, the **Options** menu provides commands that allow you to toggle between resolved and unresolved values of a variable.



**Do NOT Resolve Values:** Displays the resolved values of symbolic substitution in the value field of the variables.



**Resolve Values:** Toggles back to the symbolic notation for the values.



# Using the EDM Screen Painter

This chapter explains how to create, edit, and sequence screen objects using the EDM Screen Painter. A screen can prompt the user to input information (for example, name and password). This chapter also explains how to use screens to invoke outside processing using EDM REXX methods, modify EDM objects, and execute programs.

## Before You Use the EDM Screen Painter

---

Before using the EDM Screen Painter, have a clear concept or sketch of the screens you want to create and the functions you want associated with them. Avoid creating screens that are too wordy or vague. The clearer the screen is, the less often a user will need to turn to the system administrator or a manual for help.

When designing more than one screen, make sure that the screens maintain a consistent appearance.

If you are designing multiple screens that interact with outside programs, you must consider the sequence in which the screens will be displayed, as well as the complexities of the manipulation and resolution of the variables.

**Warning:** Do not attempt to create or edit screen objects with the EDM Client Explorer.

Be careful not to corrupt screen object variables while viewing them with the EDM Client Explorer.

## Screens and EDM Object Resolution

Nearly every screen you create will access and set variables within other EDM managed data objects on the desktop. The screen (dialog box) is an ideal tool for obtaining information from a user, but in order for EDM to be able to use that information, it must be saved in an EDM object.

For example, it is very easy to create a screen that prompts the user to type a user name and password. But this information will be of little use unless it is saved in, or compared to the information saved in, ZMASTER.ZUSERID and ZMASTER.ZPWD objects.

EDM screen objects can display the values of variables in EDM objects. EDM screen objects can also display variable names in symbolic notation without resolving the object.

The EDM objects used in the screen display program are ZMASTER (the master data resolution object), the screen itself, and any object explicitly referenced within the screen.

The EDM screen facility uses reserved Z-named variables to implement logical navigation and pass information regarding user actions to subsequent programs (REXX methods). For more information see the section, “*EDM Screen Painter Variable Reference*” in this chapter. With REXX methods, you can create screens that add and create variables and objects.

## EDM Screen Objects at a Glance

Using the EDM Screen Painter (EDMPANEL.EXE), you can create and edit EDM screen objects. Like all EDM objects, screen objects have the .EDM extension, however, there are differences between the two types of objects. The following table compares and contrasts EDM screen objects and regular EDM objects.

Feature	EDM Screen Objects	Other EDM Objects
Has .EDM extension.	✓	✓
Can be viewed with EDM Client Explorer.	✓	✓
Can be edited with EDM Client Explorer.		✓
Can be viewed with EDM Screen Painter.	✓	
Can be edited with EDM Screen Painter.	✓	
Can be used in a REXX method.	✓	✓

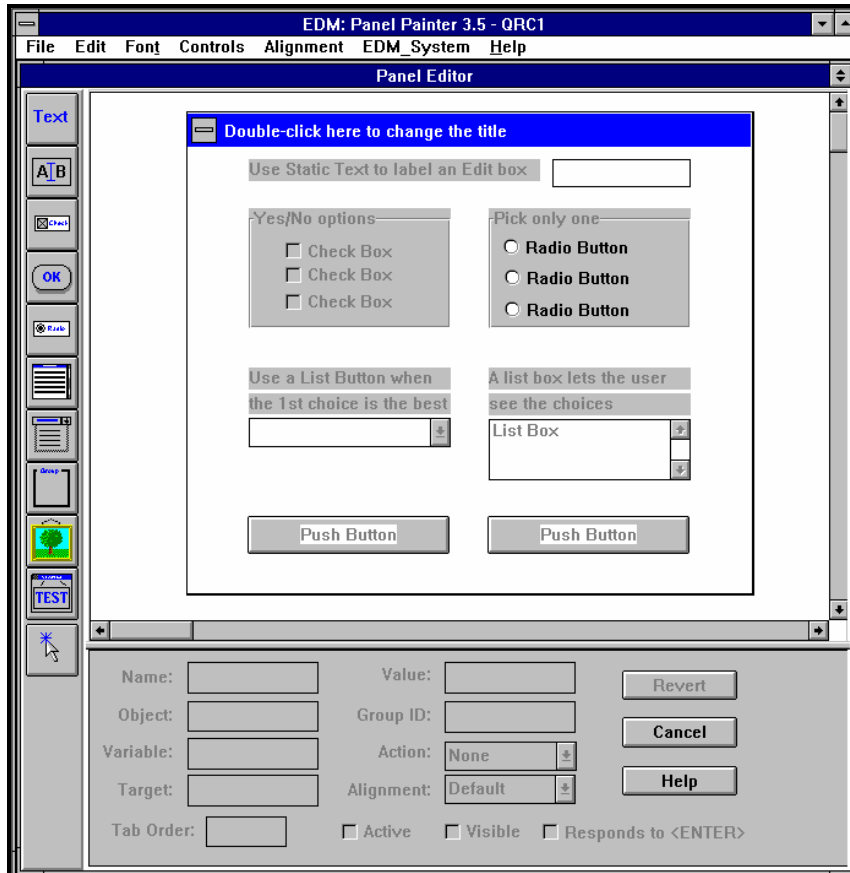
The EDM Client user version (EDMPNLGN.EXE and EDMPNLGR.EXE), included with the EDM Client for Windows NT, displays screen objects that were previously created by the EDM Screen Painter.

## The Screen Editor Screen

---

When you open the EDM Screen Painter, the following is displayed.








## The EDM Screen Painter Screen







## The Screen Edit Tools

The following table describes the available controls and tools on the Controls toolbar. Note that all of the items in the Controls toolbar are also available from the **Controls** menu.

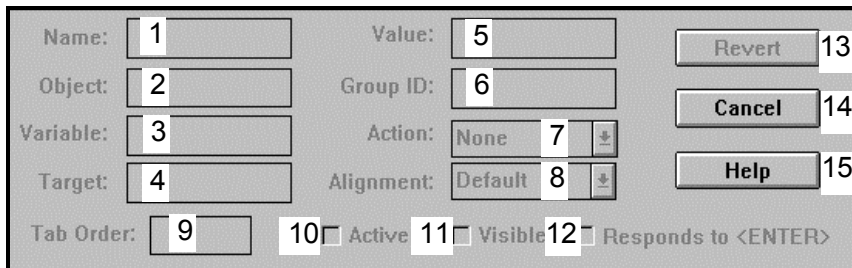
### EDM Screen Painter Controls Toolbar

Tool Name	Tool	Description
Static Text		Use this control to display labels, descriptions, and object variable values.
Edit		Use this control for inputting text and displaying object variable values.
Push Button		Use this control to perform any of the following actions: EXIT, CANCEL, SHOW, ENABLE, SCREEN, LAUNCH, PREVHP, NEXTHP
Check Box		Use this control for yes/no or on/off type items.
Radio Button		Use this control for a multiple choice item where only one option can be selected.
Group Box		Use this control to group and organize other controls.
List Box		Use this control to provide a scrollable list of choices.

Tool Name	Tool	Description
List Button		Use this control to display a drop down list of choices where only one can be selected.
Image		Use this control to display a bitmap image.
Test		Use this tool to execute the screen in real time mode.
Select		Use this tool to select and modify screens, controls, and properties.

## The Properties Window

A general description of the Properties Window follows. Certain controls require only a few properties. Details on assigning properties for each control are provided in the section, “*Screen Painter Reference*” in this chapter.



The Properties Window contains the following fields and buttons:

- Name:
- Value:
- Revert:
- Object:
- Group ID:
- Cancel:
- Variable:
- Action:
- Help:
- Target:
- Alignment:
- Tab Order:
- 10 Active
- 11 Visible
- 12 Responds to <ENTER>

## EDM Screen Painter Properties Window

Item	Property	Description
1	<b>Name</b>	The label displayed on the control. Also used for indicating a target name.
2	<b>Object</b>	The object where the text to be displayed is located, and from which the control reads and writes data.
3	<b>Variable</b>	The variable where the text to be displayed is located, and from which the control reads and writes data.
4	<b>Target</b>	The screen, file, or control that is affected by what is specified in the Action property.
5	<b>Value</b>	What you specify in the Value property depends on the control.
6	<b>Group ID</b>	A unique string that identifies radio buttons and other controls in a group.
7	<b>Action</b>	Specifies what the control will do when selected.
8	<b>Alignment</b>	Justification for the text on the control. The default (Left) is recommended for most controls. Center is recommended for push buttons.
9	<b>Tab Order</b>	Specifies the order in which the control is selected when the TAB key is pressed.
10	<b>Active</b>	Determines whether control will be initially enabled (selectable) or disabled (grayed out). (Default: active.)
11	<b>Visible</b>	Determines whether the control will be initially displayed or hidden. (Default: visible.)
12	<b>Responds to &lt;Enter&gt;</b>	Allows the user to choose the push button by pressing Enter on the keyboard. Available for only one control on each screen.
13	<b>Revert</b>	Restores property values for the control to the original values when the control was selected. <b>Note: Revert</b> in the <b>File</b> menu restores the screen to its last saved version.
14	<b>Cancel</b>	Closes the editing session without saving changes.

Item	Property	Description
15	Help	Invokes on-line help (not available yet).

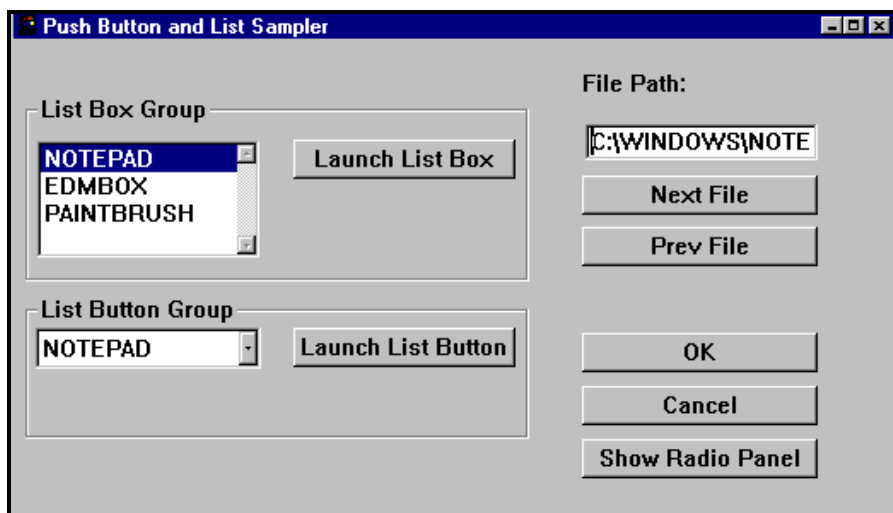
## Screen Creation Tutorial

This section walks you through the creation of a desktop object and two screens. These sample screens utilize all of the available Screen Editor controls and reference an object you create. To get input from a user, there are eight different types of controls you can include in a screen. However, chances are you will never design a screen that utilizes all eight.

If you will be designing numerous screens for your enterprise, you may want to refer to a GUI design handbook such as Weinschenk and Yeo, *Guidelines for Enterprise-Wide GUI Design*, John Wiley and Sons, Inc., 1995.

### Push Button and List Sampler

Figure 1 - Push Button and List Sampler Window



The Push Button and List Sampler window demonstrates the use of push button and lists, and the actions they perform. It also contains a static text box, an edit box, and two group boxes.

### Selecting a File

This window provides you with several different ways to select one of three files (Paintbrush, Notepad, and an EDM message). To select the file to launch you can:

- 1 Select from the list box.
- 2 Select from the list button.
- 3 Scroll through the files displayed in the text box using the **Next File** and **Prev File** buttons.
- 4 Type the file path in the text box.
- 5 When you design a screen for your site, you will most likely have only one way to select a file to launch. The sample screen provides several options for demonstration purposes only.

## Launching a File

This screen also provides several ways to launch the selected file. You can launch a selected file by:

- 1 Double-clicking a selection in the list box.
- 2 Choosing the **Launch Listbox** button.
- 3 Choosing the **Launch Drop Down** button.

**Note:** You can launch a file by double-clicking on a selection in a list box, but not in a list button.

## Common Push Buttons

Nearly every screen you design will probably have at least two push buttons: **OK** and **Cancel**. An **OK** button saves changes and closes the screen. A **Cancel** button closes the screen without saving changes.

**Tip:** As a rule of thumb, do not place more than six push buttons on a screen. More than six buttons will clutter the screen and confuse the user. Also, keep push buttons the same size and keep them aligned.

Before placing any buttons you must first begin an editing session. The rest of this tutorial will walk you through the steps, and explanations, for creating the sample screens and object mentioned earlier.

## Build the Sample Object

To help you better understand how EDM Screen objects interact with other EDM objects, most of the controls in the Push Button and List Sampler window reference an object that you are about to build. This object, EDTEST.EDM, has only two variables, RUNNAME and RUNFILE.

RUNNAME specifies the name of a file that a user can launch. RUNFILE specifies the full path for the file. The user will be able to choose from three different files, therefore this object will contain three instances (heaps). You can add more instances if you so desire.

The easiest way to build this object is by using the EDM Client Explorer. If you are unfamiliar with the EDM Client Explorer, see Chapter 4, *"The EDM Client Explorer"* in this book.

### ➤ To build the object:

Launch the EDM Client Explorer from the EDM Administrator folder.

- 1 From the **Object** menu, Choose **New...** In the dialog box, type EDTEST and then press ENTER.
- 2 To create Heap 1, choose **Add...** from the **Variable** menu. In the dialog box type RUNNAME and then press ENTER.
- 3 In the next dialog box, type NOTEPAD in the **Value** field and then press ENTER. In doing so, you are giving the variable **RUNNAME** the value of **NOTEPAD**.
- 4 From the **Variable** menu, choose **Add...**  
In the dialog box, type RUNFILE and then press ENTER.
- 5 In the next dialog box type C:\WINDOWS\notepad.exe in the **Value** field and then press ENTER.

- 6 To create Heap 2, choose **Add...** from the **Heap** menu, and then choose **RUNNAME** by double-clicking on it.
- 7 In the dialog box type **EDMBOX** and then press ENTER. In doing so, you are assigning the value of **EDMBOX** to the **RUNNAME** variable of this instance (heap).
- 8 Choose **RUNFILE** by double-clicking on it, and in the **Value** field type  
C:\FOLDERNAME\EDMBOX.EXE, where "foldername" is the name of a folder where you want to put edmbox.exe.
- 9 To create Heap 3, choose **Add...** from the **Heap** menu.
- 10 Choose **RUNNAME** by double-clicking on it, and in the **Value** field type PAINTBRUSH.
- 11 Choose **RUNFILE** by double-clicking on it, and in the **Value** field type C:\WINDOWS\PBRUSH.EXE.
- 12 From the **Object** menu choose **Save**, and then exit the EDM Client Explorer.

You have now finished building the EDTEST object. It is described in table format below.

### The EDTEST Object

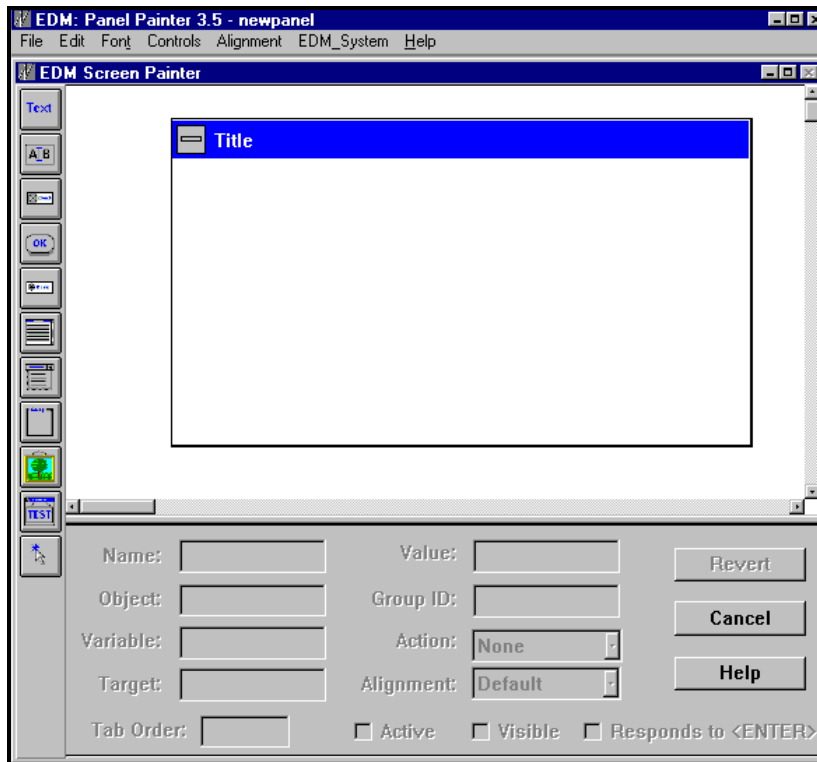
Heap Number	Variables	Value
Heap 1	RUNFILE	C:\WINDOWS\notepad.exe
	RUNNAME	NOTEPAD
Heap 2	RUNFILE	C:\FOLDERNAME\EDMBOX.EXE
	RUNNAME	EDMBOX
Heap 3	RUNFILE	C:\WINDOWS\PBRUSH.EXE
	RUNNAME	PAINTBRUSH

### Open the EDM Screen Painter

Open the EDM Screen Painter by choosing **EDM Screen Painter** in the EDM Administrator folder on your desktop.

### Create a New Screen

Create a new screen by choosing **New** in the **File** menu. If you just opened the EDM Screen Painter, you have a new screen ready for editing. Note that you cannot have more than one screen open for editing.

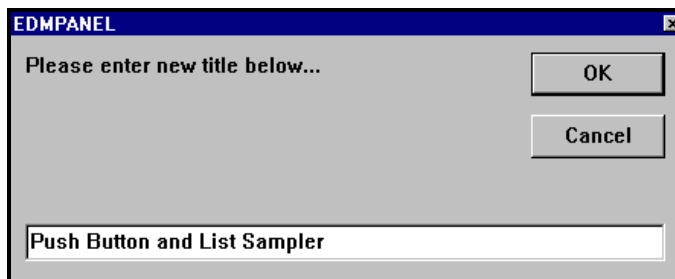


**Tip:** Maximize the Screen Painter Window and lower the Properties Window.

**Tip:** A good habit to get into is to name and save the screen before adding controls.

## Assign the Title

To assign a title to your screen, choose **Set Title** from the **Controls** menu. In the dialog box that is displayed, type **Push Button and List Sampler**. Then choose **OK** or press **ENTER**. **Push Button and List Sampler** will appear in the top blue line of the new screen name.




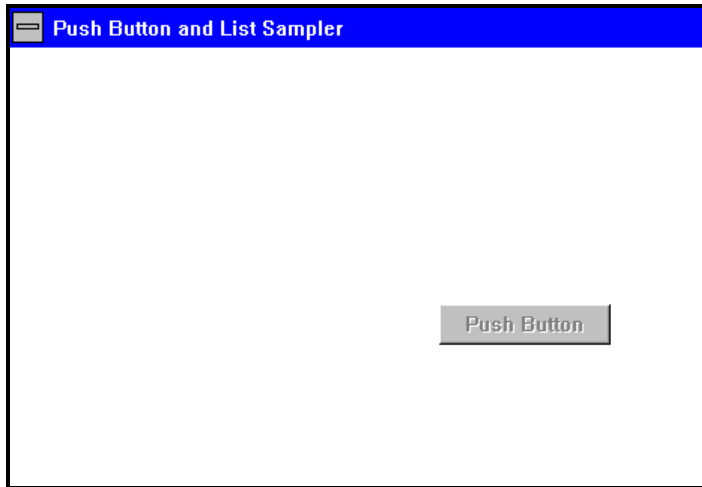
**Tip:** Save your work regularly to prevent hours of work from being accidentally destroyed. To save your screen, from the **File** menu, choose **Save** (or **Save As**). In the dialog box that displays, name your screen **A1SAMPLE**.

## Add the OK Button

You will probably want to test your screen several times during an editing session to verify that the screen functions the way you intended. Add a push button to close the screen early in the editing process, so that when you finish testing you can easily close the screen and return to your editing session.



To add a button, select the **Push Button** tool  from the toolbar, or choose **Push Button** from the **Controls** menu. The cursor is displayed as a push button. Then, click and drag a rectangle of the desired size on the screen. A push button, the size of the rectangle you created is displayed.



The push button you just created is named **Push Button**, and it has no function. To change its name and to give it a function (designate an action) you must edit its properties.

### Assign the OK Button Properties

To assign properties to the button you just created (or to edit the properties of any control) click on the **Push Button**. When a control is selected, sizing handles are displayed on its corners as shown below. Now you can now edit the Properties fields.



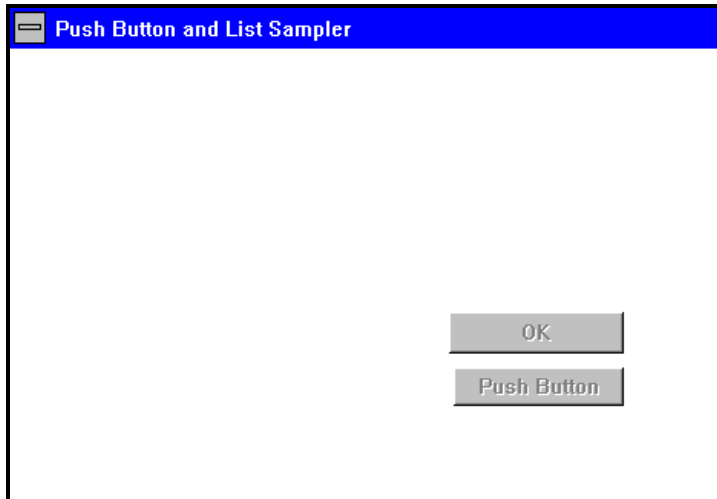
Enter the values for the **OK** button properties as indicated in the table below. The values are updated when you click outside the Properties Window.

Push	<input type="text" value="OK"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text"/>	Group ID:	<input type="text"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text"/>	Action:	<input type="text" value="EXIT"/>	<input type="button" value="Help"/>
Target:	<input type="text"/>	Alignment:	<input type="text" value="Center"/>	
Tab Order:	<input type="text" value="1"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice that the **Action** property is set to **Exit**, not **Cancel**. The action **Exit** causes the screen to close and save changes to the screen and related objects. The action **Cancel** causes the screen to close without saving any changes.

### Add the Cancel Button

Add another push button underneath the **OK** button.



Don't worry if the button you create is slightly smaller or larger than the **OK** button and don't worry if the two are not aligned. This tutorial will address size and alignment issues later.

### Assign the Cancel Button Properties

Select the new push button you just added and assign properties as indicated below.

Push: <input type="text" value="Cancel"/>	Value: <input type="text"/>	<input type="button" value="Revert"/>
Object: <input type="text"/>	Group ID: <input type="text"/>	<input type="button" value="Cancel"/>
Variable: <input type="text"/>	Action: <input type="text" value="CANCEL"/>	<input type="button" value="Help"/>
Target: <input type="text"/>	Alignment: <input type="text" value="Center"/>	
Tab Order: <input type="text" value="2"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible
	<input type="checkbox"/> Responds to <ENTER>	

### Buttons that Launch Other Screens

Notice the **Show Radio Panel** button in the lower right of the Push Button and List Sampler screen (in Figure 1 earlier in this chapter). This button closes the Push Button and List Sampler screen and launches the Radio Button Sampler screen. To have a push button launch another screen, set the **Action** property to **Screen**. The Screen Editor knows which screen to launch because you specify that screen's name in the **Target** property field.

### Add Show Radio Panel Button

Use the Push Button tool from the tool bar to create a new push button. Place it below the **Cancel** button. Because you have not created the Radio Button and Check Box Sampler screen yet, you cannot specify it as a target. In the **Push** field, type Show Radio Panel. We'll assign the remaining properties to this button after creating the Radio Button and Check Box Sampler screen.

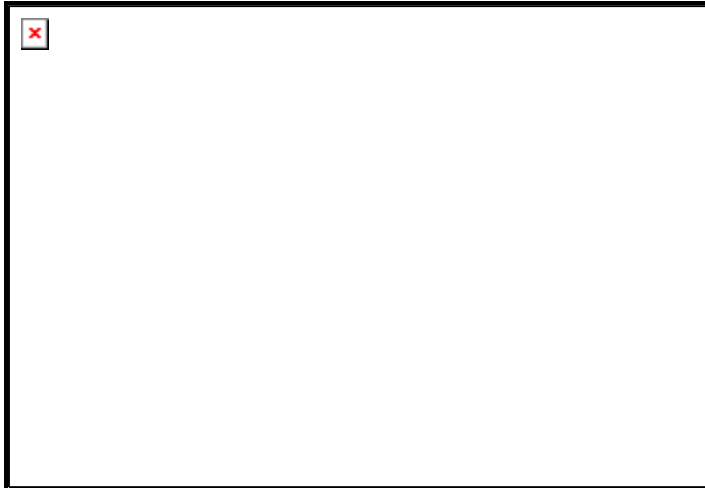
There are four more buttons that we need to place on this screen: **Launch List Box**, **Launch List Button**, **Next File** and **Prev File**. Before we do so, we'll create the controls they reference, namely, a list box, a list button (drop down list) and an edit (text) box.

**Tip: List Boxes Vs. List Buttons:**

Use a list box if you want the user to see all of the available options. Use a list button (drop down list) if most users will select the first item, or if you want to save space on the screen.

**Add the List Box**

To insert a list box, select the **List Box** tool from the toolbar, or choose **List Box** from the **Controls** menu. The cursor is displayed as a list box. Then, click and drag a rectangle of the desired size on the screen. Make the list box big enough to display between three and eight items. A list box the size of the rectangle you created is displayed.

**Assign the List Box Properties**

Select the list box with your pointing device and enter the values as indicated in the table below.

List Box	List Box	Value:	LAUNCH	Revert
Object:	EDTEST	Group ID:		Cancel
Variable:	RUNNAME	Action:	PAGEHP	Help
Target:	RUNFILE	Alignment:	Default	
Tab Order:	4	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

**Explanation of Properties**

Double-click on **List Box** to display a list. The only needed properties are **Object** and **Variable**. This list box references the local object that you built earlier, EDTEST.EDM, and displays the variable RUNNAME.

The variables in the object contain the information necessary to launch a file. The variable RUNNAME contains the name of the file to launch (for example, NOTEPAD). The variable RUNFILE contains the full path of the file to launch (for example C:\WINDOWS\notepad.exe). There is an instance for each file (for each item on the list). When you select an item from the list, the instance that contains that item becomes the currently active instance.

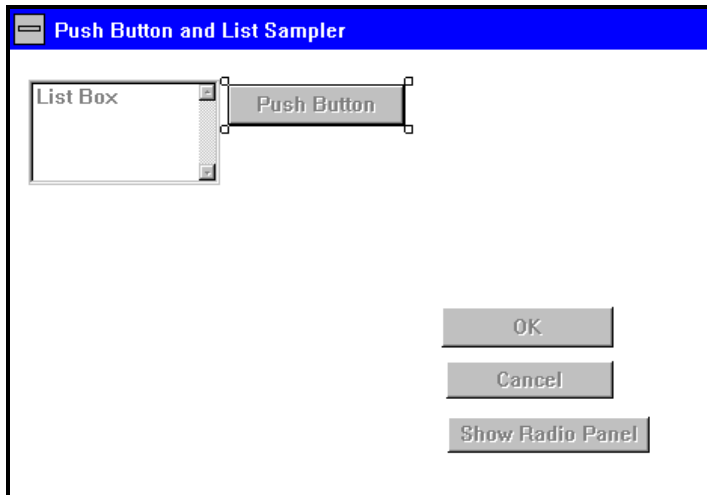
Specifying LAUNCH in the **Value** field enables a list box to launch a file if an item is double-clicked. When you specify LAUNCH, you must also specify what is to be launched in the **Target** field.

## Add the Launch Listbox Button

In addition to double-clicking in a list box, you can launch the item selected in a list box with a push button. For a push button to launch a file, its **Action** property must be set to LAUNCH. In this case the **Target** property is set to List Box. Hence, if the button is chosen, it performs the action LAUNCH on the target, the selected item in the list box.



To add this button, select the **Push Button** tool from the toolbar, or choose **Push Button** from the **Controls** menu. The cursor is displayed as a push button. Then, click and drag a rectangle of the desired size on the screen. A push button, the size of the rectangle you created is displayed.



## Assign the Launch Listbox Button Properties

Select the push button with your pointing device and enter the values as indicated in the table below. Note that the full value of the **Push** field should be Launch List Box.

Push:	Launch List Bo	Value:		Revert
Object:	EDTEST	Group ID:		Cancel
Variable:	RUNFILE	Action:	LAUNCH	Help
Target:	List Box	Alignment:	Center	
Tab Order:	5	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

A list button is the control of choice when:

- You have many options to choose from. Use a list instead of cluttering a screen with more than six radio buttons. The data or options are likely to change over time. If you use radio buttons or push buttons, you will have to modify your screens every time the options change. If you use lists, you won't have to modify your screens as options change. This is because the choices in a list are contained in a separate EDM object.

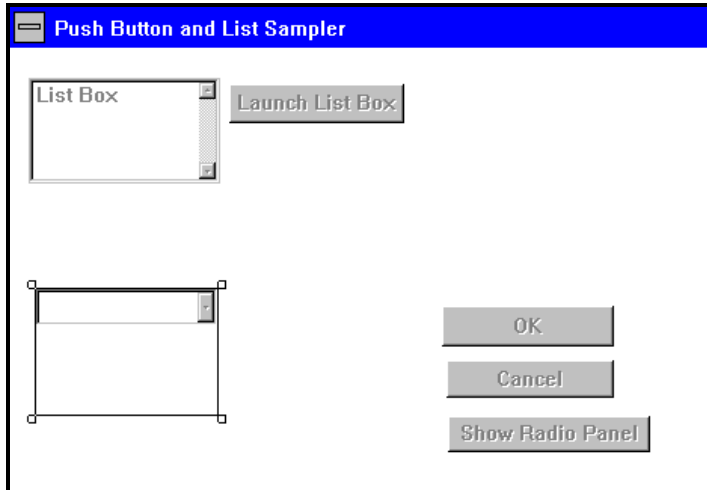
In the Push Button and List Sampler screen, the user selects and launches a file. The choices, Notepad, Paint Brush, and EDMBOX are contained in the local object EDTEST.EDM. If more options are added to the object, or if the options change, the list in the screen is updated automatically.

## Add the List Button

Unlike list boxes, list buttons cannot launch a file if the user double-clicks on an item. A list button can, however, update static and edit text boxes with the **PAGEHP** action.



Select the **List Button** tool from the toolbar, or choose **List Button** from the **Controls** menu. The cursor is displayed as a list button. Then, click and drag a rectangle of the desired size on the screen. Draw the rectangle long enough to accommodate the text it will display. A list button, the size of the rectangle you created is displayed. Note that its name will not be displayed until you close the file and reopen it.



## Assign the List Button Properties

Select the list button with your pointing device and enter the values as indicated in the table below.

List	List Button	Value:		Revert
Object:	EDTEST	Group ID:		Cancel
Variable:	RUNNAME	Action:	PAGEHP	Help
Target:	RUNFILE	Alignment:	Default	
Tab Order:	6	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

## Add the Launch List Button

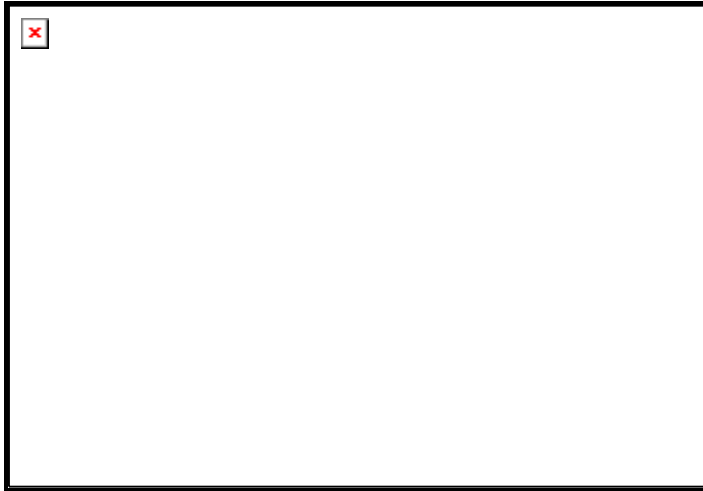
Unlike a list box, a list button cannot be specified as the target of a LAUNCH action. Instead, the object that the list button references is specified.

If the user selects an item from the list button and then chooses the **Launch List Button**, the selected item in the list button is launched. However, if the user selects an item from the list button and then selects a different item from the list box, the item in the list box is launched. This is because this button launches the file in the currently selected instance in the EDTEST.EDM object.

If the screens you design do not offer an alternative to the list button for selecting from a particular object, then a LAUNCH push button will launch what ever is selected in the list button.



To add this button, select the **Push Button** tool from the toolbar, or choose **Push Button** from the **Controls** menu. The cursor is displayed as a push button. Then, click and drag a rectangle of the desired size on the screen. A push button the size of the rectangle you created is displayed.



### Assign the Launch List Button Properties

Select the new push button with your pointing device and enter the values as indicated in the table below. Note that the full **Push** value should be `Launch List Button`.

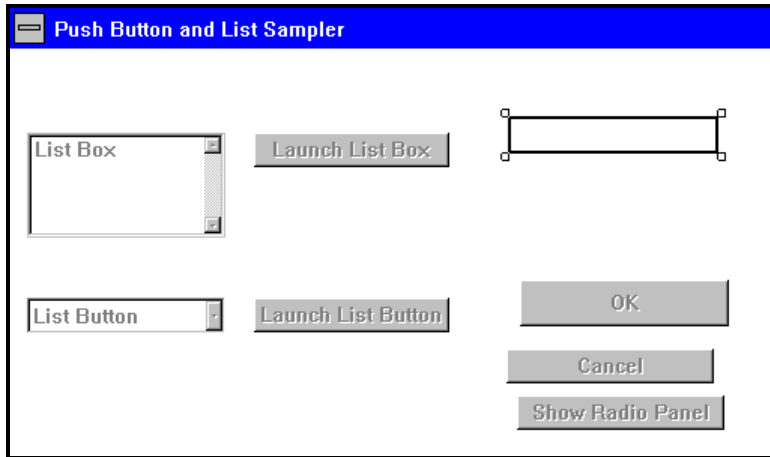
Push	<input type="text" value="Launch List Bu"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text" value="EDTEST"/>	Group ID:	<input type="text"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text" value="RUNFILE"/>	Action:	<input type="text" value="LAUNCH"/>	<input type="button" value="Help"/>
Target:	<input type="text"/>	Alignment:	<input type="text" value="Center"/>	
Tab Order:	<input type="text" value="7"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

### Add the Edit Box

The edit box in the Push Button And List Sampler window displays the full path of the file selected in the list box or list button. Because it is an edit box instead of a static text box, you can type a new path for any item in the lists.

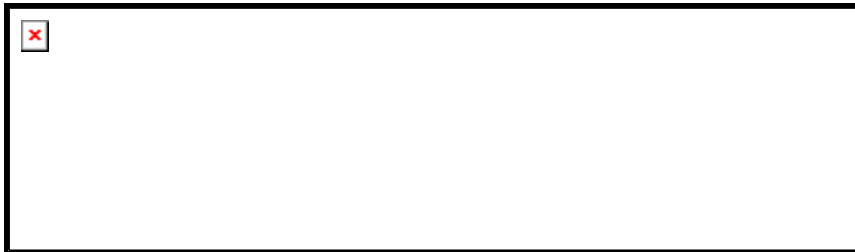


To add the edit box, select the **Edit** tool from the toolbar, or choose **Edit** from the **Controls** menu. The cursor is displayed as an edit box. Then, click and drag a rectangle of the desired size on the screen. Make sure it's big enough to accommodate the font and the text it is to display. An edit box the size of the rectangle you created is displayed.




## Assign the Edit Box Properties

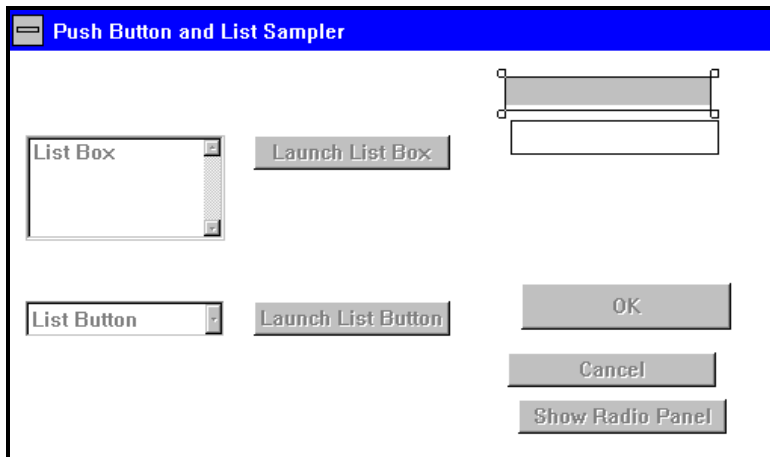
Select the edit box with your pointing device and enter the values as indicated in the table below.



## Label the Edit Box

**Tip:** Always label every control. Don't leave your users guessing what to type in an edit text box or what to choose from a list.


To add the label, select the **Static Text** tool  from the toolbar, or choose **Static Text** from the **Controls** menu. The cursor is displayed as a static text box. Then, click and drag a rectangle to contain the text on the screen. A static text box the size of the rectangle you created is displayed.



There is only one property you need to set for static text. In the **Static** field, type `File Path:`

## Add the Next File and Prev File Push Buttons

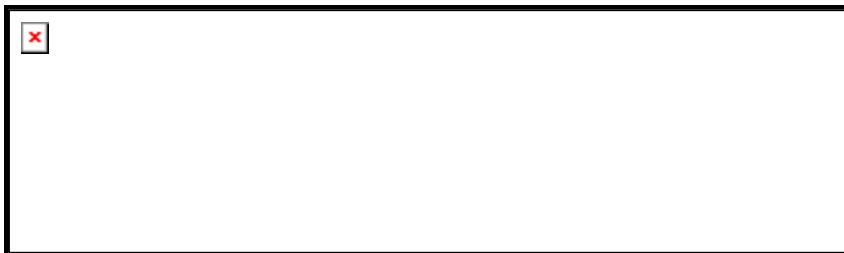
These two buttons allow you to scroll up or down through the instances so you can pick the file you want to launch. When you choose the **Next File** or **Prev File** button, the next or previous instance is displayed in the text box, depending on which button you pressed. A push button whose action is NEXTHP or PREVHP makes the next or previous instance in the object the currently selected instance.

To add these buttons, select the **Push Button** tool  from the toolbar, or choose **Push Button** from the **Controls** menu. The cursor is displayed as a push button. Then, click and drag a rectangle of the desired size on the screen. A push button the size of the rectangle you created is displayed. Repeat and create a second button underneath the first.



## Assign the Next File and Prev File Properties

Select the top push button with your pointing device and enter the values as indicated in the table below.



Now, select the lower push button with your pointing device and enter the values as indicated in the table below.

Push	<input type="text" value="Prev File"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text" value="EDTEST"/>	Group ID:	<input type="text"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text"/>	Action:	<input type="text" value="PREVHP"/>	<input type="button" value="Help"/>
Target:	<input type="text"/>	Alignment:	<input type="text" value="Center"/>	
Tab Order:	<input type="text" value="11"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Before we add the group boxes, let's align and re-size the existing buttons.



## Re-Size the Push Buttons

The buttons you created are probably not the same size.

**Tip:** Make all buttons in a given screen the same size as the largest button.

The first step in making controls the same size, is to select them. The control you select first is the one that the EDM Screen Painter uses as a standard for resizing the others. Use the **Show Radio Panel** as a standard because it contains the longest text.

To make the buttons the same size, select the **Show Radio Panel** with your mouse. Then hold down the SHIFT key and select the other buttons. From the menu bar, choose **Alignment**, and then choose **Align Same Width** from the submenu. The buttons are now the same width.

To make the buttons the same height, from the **Alignment** menu, choose **Align Same Height**. The buttons are now the same height.

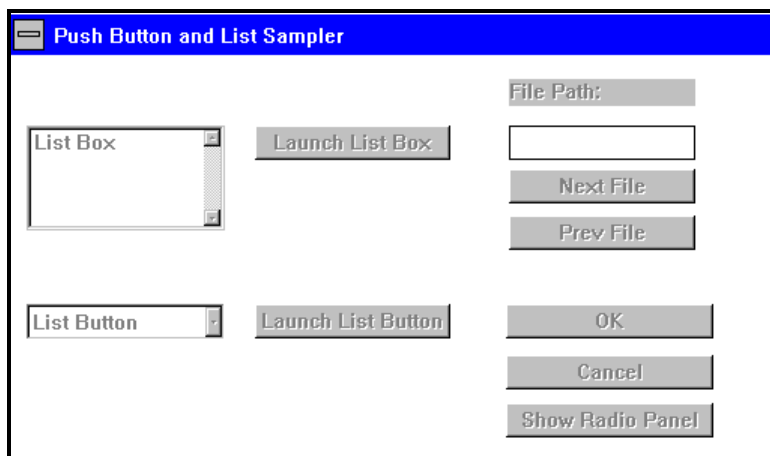
## Align the Right Side of the Screen

Give the screen a professional look by aligning all elements within the screen. Because the static text and the edit text box are not the same size as the buttons, we are going to left align the controls on the right side of the screen.

To align the controls, select one of the buttons you aligned earlier (for example **OK** or **Cancel**). Then, while pressing the SHIFT key, select the rest of the controls on the right side of the screen. From the **Alignment** menu, choose **Align Lefts**. The controls are now aligned.

To ensure that all controls in a group are evenly spaced vertically, select all the controls in that group (for example, **OK**, **Cancel** and **Show Radio Panel**) by clicking on them. Once selected, from the **Alignment** menu, choose **Align Even Vertical**. The controls are now spaced evenly vertically. The same option is available for spacing controls evenly horizontally. To do so select all controls in the group by clicking on them, select the **Alignment** menu, choose **Align Even Horizontal**.

Your panel should now look like this:



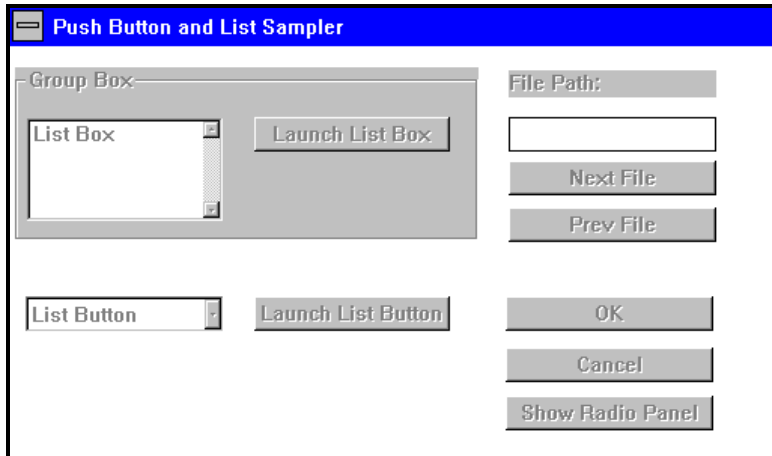
## Add the Group Boxes

A group box provides organization for your user. Without it, the screen may seem overwhelming. A group box is always the last item added to a screen.

**Note:** Once you add a group box, the controls beneath it are difficult to select. You will most likely have to move or delete the group box to modify or align the controls beneath it.



To add a group box, select the **Group Box** tool from the toolbar, or choose **Group Box** from the **Controls** menu. The cursor is displayed as a frame. Click and drag the Group Box and so it surrounds the **List Box** and **Launch List Box** buttons in the Group Box as shown below.



Repeat the above process and create a second group box that surrounds the **List Button** and **Launch Button**.

### Assign the Group Box Properties

The only property a group box needs is **Group**, its label. If you will be enabling and disabling or hiding and showing the controls in a group box, then you should also specify a **Group ID** for the group box and its controls.

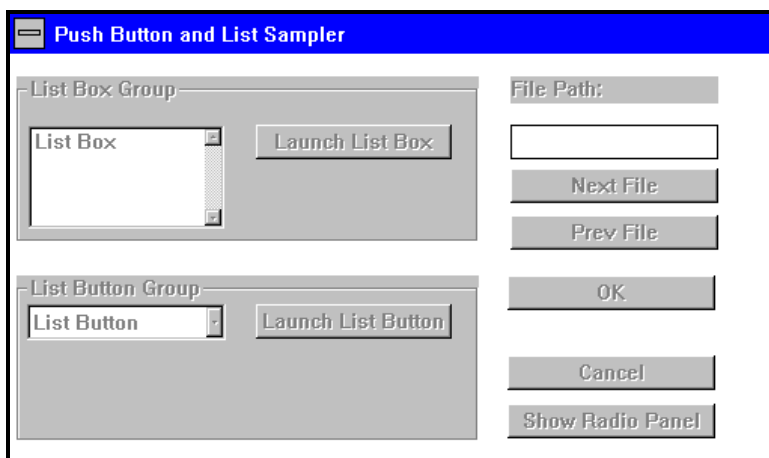
Select the top group box with your pointing device and type List Box Group in the **Group** field.

Select the lower group box and type List Button Group in the **Group** field.

### Align the Group Boxes


Select both group boxes by holding down the SHIFT key and selecting both Group Boxes. If your group boxes are not the same size, align their sizes the way we did with the buttons on the right side of the screen earlier. Then, from the menu bar, choose **Alignment**, and then choose **Align Lefts** from the submenu.

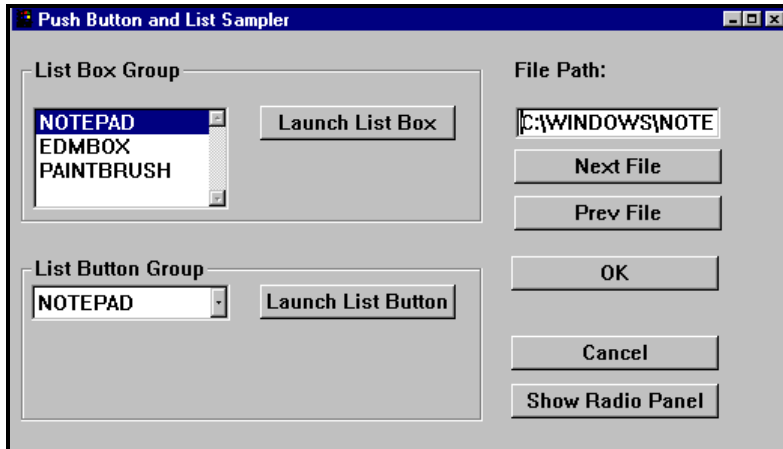
Everything in your screen should now be aligned and should look like the following screen.



## Test the Screen

Test your screen to make sure everything works properly. (The **Show Radio Panel** push button is the only control that is not functional yet.)

Choose the **TEST** tool  from the tool bar, or choose **Test** from the **EDM\_System** menu by clicking on Test Tool. A new window opens and displays a fully functional screen as it will be displayed on the user's screen.



If the screen displays off center on your screen, select and move the screen in the editing window. If any of the controls do not work properly, go back and fix them now.

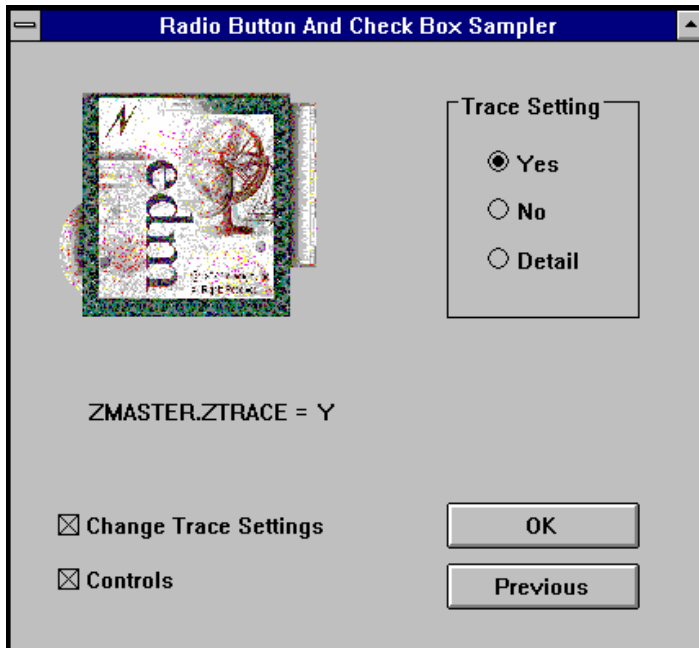
Users can now press either **Esc** or **Cancel** to close a screen.

Save your work by choosing **Save** from the **File** menu. Now that you have saved your work, you can create the next screen.

**Note:** When the cancel push button is pressed, the ZNAMESEL variable is set to cancel. In the REXX version a REXX script will be called and ZNAMESEL is not set.

## Radio Button and Check Box Sampler

### Radio Button and Check Box Sampler Screen



The Radio Button and Check Box Sampler Screen demonstrates the use of radio buttons, check boxes, and the actions they perform. This screen also contains two push buttons, two static text fields, and a group box.

This screen shows you the current value of the trace setting in the ZMASTER object and allows you to change the value with the radio buttons. The valid values are Y (yes), N (no), and D (detail). By using radio buttons, instead of a text box, you are preventing the user from specifying an invalid value.

**Note:** This screen actually writes to the ZTRACE variable in the ZMASTER object. If you change the value while experimenting with the screen, be sure to change it back to its original setting.

**Tip:** Use radio buttons when the user must choose one and only one option from a list of up to six options. If there are more than six options, use a list box or a list button.

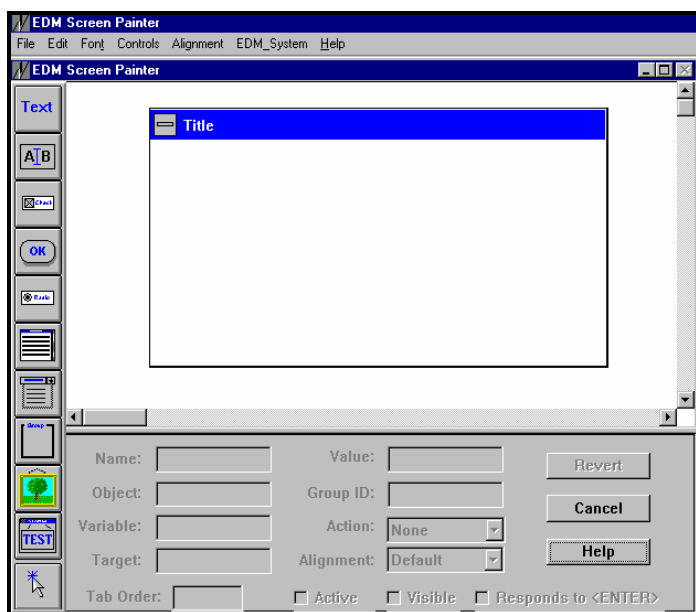
The check boxes demonstrate two new actions, SHOW and ENABLE. SHOW toggles controls between visible and invisible. Hide controls when most users won't need them, but when a few will. Users will not experiment with controls that are hidden. However, these controls will still be available to the few users who do need them.

ENABLE toggles controls between being active and inactive (grayed out). An inactive control signals the user that something must be done for the desired control to become active. In this case, the **OK** and **Previous** buttons are not active until the **Controls** check box is checked. The user has no choice but to check the **Controls** check box to continue.

Before adding controls, you must first create a new screen.

### Create a New Screen

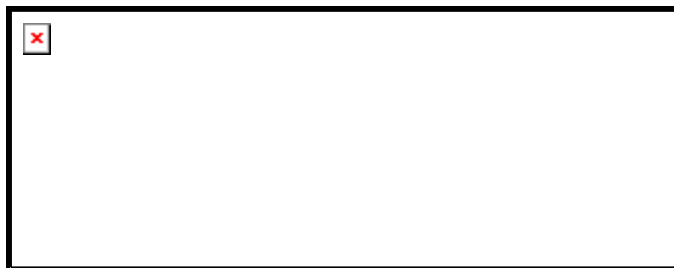
From the **File** menu, choose **New**. The current session closes and a new screen displays in the Screen Painter window. Maximize the window and shrink the properties window to your convenience.



## Assign the Title

To assign a title to your screen, choose **Set Title** from the **Controls** menu, or double-click the title bar of the screen. In the dialog box that is displayed, type Radio Button And Check Box Sampler.


Then choose **OK** or press ENTER.

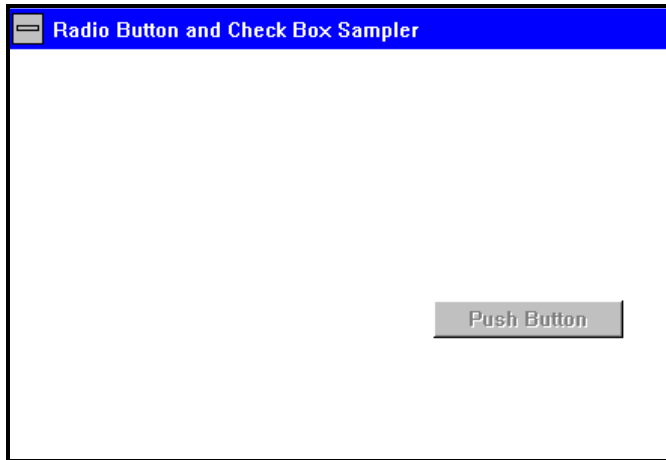


## Save the Screen

Save your work regularly to prevent hours of work from being accidentally destroyed. To save your screen, from the **File** menu, choose **Save As....** In the dialog box that appears type A2SAMPLE to name your screen.

## Create an OK button

To create this button, select the **Push Button** tool  from the toolbar, or choose **Push Button** from the **Controls** menu. The cursor is displayed as a push button. Then, click and drag a rectangle of the desired size on the screen. A push button the size of the rectangle you created is displayed.



The push button you just created is named **Push Button**, and it has no function. To change its name and to give it a function (designate an action) you must edit its properties.

### Assign the OK Button Properties

Select the push button with your pointing device and enter the values for the **OK** button properties as indicated in the table below.

Push	<input type="text" value="OK"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text"/>	Group ID:	<input type="text" value="controls"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text"/>	Action:	<input type="text" value="None"/>	<input type="button" value="Help"/>
Target:	<input type="text"/>	Alignment:	<input type="text" value="Default"/>	
Tab Order:	<input type="text" value="1"/>	<input type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice the value controls in the **Group ID** field. We will use this later.

### Add the Previous Button

A **Previous** button will allow you to return to the previous screen. To include it add another push button below the OK button.



## Assign the Previous Button Properties:

Select the push button with your pointing device and enter the values for the **Previous** button properties as indicated in the table below.


Push	<input type="text" value="Previous"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text"/>	Group ID:	<input type="text" value="controls"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text"/>	Action:	<input type="text" value="PANEL"/>	<input type="button" value="Help"/>
Target:	<input type="text" value="A1SAMPLE"/>	Alignment:	<input type="text" value="Center"/>	
Tab Order:	<input type="text" value="2"/>	<input type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input checked="" type="checkbox"/> Responds to <ENTER>

Notice the value controls in the **Group ID** field. We will use this later.

Also notice that the **Responds to Enter** check box is selected. With this check box selected, at runtime, you can press ENTER on your keyboard, instead of using your mouse, to choose this button.

## Add the ZMASTER.ZTRACE Description Text

This label displays static text. The value of this static text field does not change, regardless of what the user enters on the screen.


To add the label, select the **Static Text** tool  from the toolbar, or choose **Static Text** from the **Controls** Menu. The cursor is displayed as a static text box. Then, click and drag a rectangle to contain the text on the screen. A static text field the size of the rectangle you created is displayed.



There is only one property you need to specify in the properties box for static text. In the **Static** field, type ZMASTER.ZTRACE =.

## Add the ZMASTER.ZTRACE value label

This static text field displays the value of the ZTRACE variable in the ZMASTER object. Because it is static text, not edit text, a user cannot select it on the screen, nor can the user type in a new value.

To add the label, select the **Static Text** tool  from the toolbar, or choose **Static Text** from the **Controls** menu. The cursor is displayed as a static text box. Then, click and drag a rectangle to contain the text on the screen. A static text field the size of the rectangle you created is displayed.



### Assign the ZMASTER.ZTRACE Value Static Text Properties

Select the static text field with your pointing device and enter the values for the static text properties as indicated in the table below.


Static:	<input type="text"/>	Value:	<input type="text"/>	<input type="button" value="Revert"/>
Object:	<input type="text" value="ZMASTER"/>	Group ID:	<input type="text"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text" value="ZTRACE"/>	Action:	<input type="text" value="None"/>	<input type="button" value="Help"/>
Target:	<input type="text"/>	Alignment:	<input type="text" value="Default"/>	
Tab Order:	<input type="text" value="4"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice that values are specified in the **Object** and **Variable** fields but not in the **Static** field. This ensures that the static text displays the current value of the ZTRACE object in the ZMASTER variable.

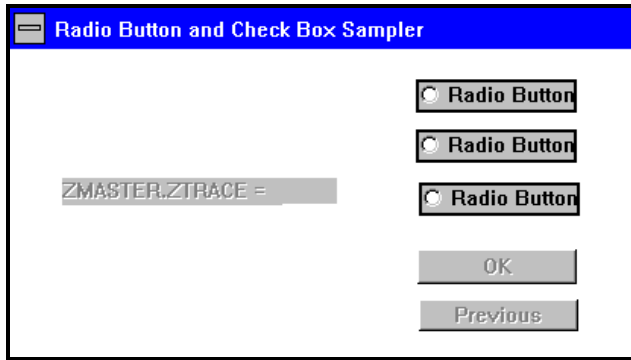
### Add the Radio Buttons

**Tip:** To ensure that only one radio button can be selected at a time, radio buttons are assigned a group identification in the **Group ID** field. When you select a radio button, any other radio button with the same group ID is deselected. When assigning a group ID, choose a value that makes sense to you.



To add a radio button, select the **Radio Button** tool  from the toolbar, or choose **Radio Button** from the **Controls** menu. The cursor is displayed as a radio button. Then, click and drag a rectangle on the screen. A standard sized rectangle is created to the radio button. Repeat and create a second and third button underneath the first.





## Assign the Radio Button Properties

Select the top radio button with your pointing device and enter the values as indicated in the table below.



Select the middle radio button with your pointing device and enter the values as indicated in the table below.

Radio	Yes	Value:	Y	Revert
Object:	ZMASTER	Group ID:	radio	Cancel
Variable:	ZTRACE	Action:	None	Help
Target:		Alignment:	Default	
Tab Order:	5	<input checked="" type="checkbox"/> Active	<input type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Now, select the bottom radio button with your pointing device and enter the values as indicated in the table below.

Radio	Detail	Value:	D	Revert
Object:	ZMASTER	Group ID:	radio	Cancel
Variable:	ZTRACE	Action:	None	Help
Target:		Alignment:	Default	
Tab Order:	7	<input checked="" type="checkbox"/> Active	<input type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice that in the above property windows the **Visible** check box is not checked. These radio buttons will not be displayed in the screen unless the **Change Trace Settings** check box (which we will define later) is checked.

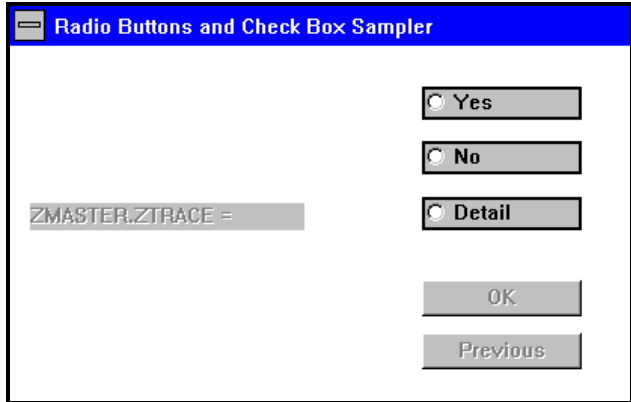
Before we add the group box, let's align the radio buttons.

## Align the Radio Buttons

**Tip:** Always left align radio buttons.

Select a radio button. The others will be aligned to it. Then, while holding the SHIFT key, select the other two. From the **Alignment** menu, choose **Align Lefts**. The radio buttons are now left aligned.

To ensure that all controls in a group are evenly spaced vertically, select all the controls in that group (for example **Yes**, **No** and **Detail**). Once selected, from the **Alignment** menu, choose **Align Even Vertical**. The controls are now vertically evenly spaced. The same option is available for evenly spacing controls horizontally; to do so select the **Alignment** menu, choose **Align Even Horizontal**.

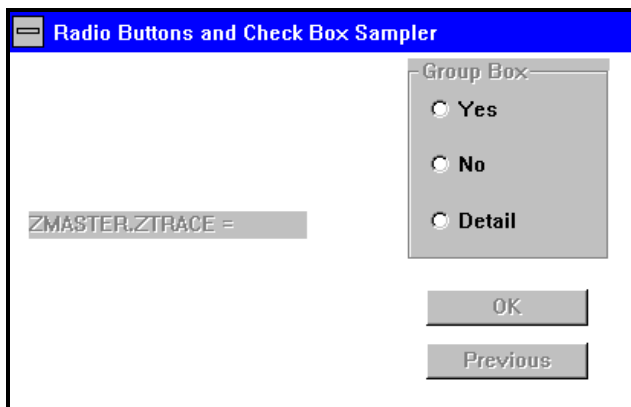


## Add the Group Box

**Tip:** Always add a group box around radio buttons.



To add a group box, select the **Group Box** tool from the toolbar, or choose **Group Box** from the **Controls** menu. The cursor is displayed as a frame. Click and drag a rectangle of the desired size on the screen. A frame the size of the rectangle you created is displayed.



## Assign the Group Box Properties


Generally, the only property a group box needs is **Group** (the text label for the group box). In this tutorial, however, you will also specify a **Group ID** so that the group box can be hidden or displayed along with the radio buttons.

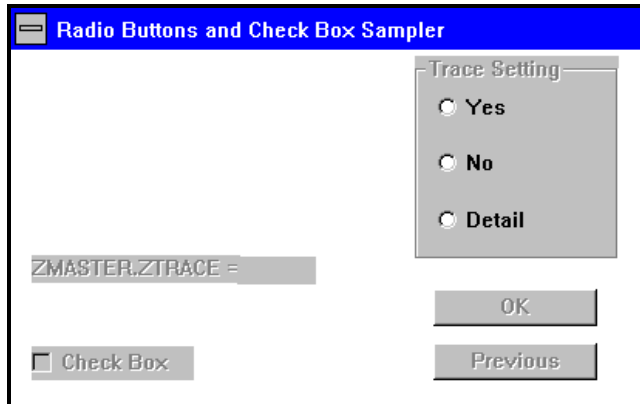
Select the group box with your pointing device and enter the values as indicated in the table below.

Group	Trace Setting	Value:		Revert
Object:		Group ID:	radio	Cancel
Variable:		Action:	None	Help
Target:		Alignment:	Default	
Tab Order:	8	<input checked="" type="checkbox"/> Active	<input type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

## Add the Change Trace Settings Check Box

As mentioned earlier, a check box is used for on/off or yes/no type items. In this case, the **Change Trace Settings** check box hides and displays the radio buttons and group box. This check box demonstrates the action SHOW.

To add a check box, select the **Check Box** tool  from the toolbar, or choose **Check Box** from the **Controls** menu. The cursor is displayed as a check box. Then, click and drag a rectangle of the desired size on the screen. The rectangle must be big enough to accommodate the check box label. A check box the size of the rectangle you created is displayed.



## Assign the Change Trace Settings Check Box Properties


Select the new check box with your pointing device and enter the values as indicated in the table below. Note that the full value for **Check** should be Change Trace Settings.

Check	Change Trace	Value:	N	Revert
Object:		Group ID:		Cancel
Variable:		Action:	SHOW	Help
Target:	radio	Alignment:	Default	
Tab Order:	9	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice the **Target** is the **Group ID** you specified for the radio buttons.

## Add the Controls Check Box

The **Controls** check box enables and disables the push buttons on this screen. This check box demonstrates the action ENABLE.

To add a check box, select the **Check Box** tool  from the toolbar, or choose **Check Box** from the **Controls** menu. The cursor is displayed as a check box. Then, click and drag a rectangle of the desired size on the screen. The rectangle must be big enough to accommodate the check box label. A check box the size of the rectangle you created is displayed.



## Assign the Controls Check Box Properties


Select the **Controls** check box with your pointing device and enter the values as indicated in the table below.

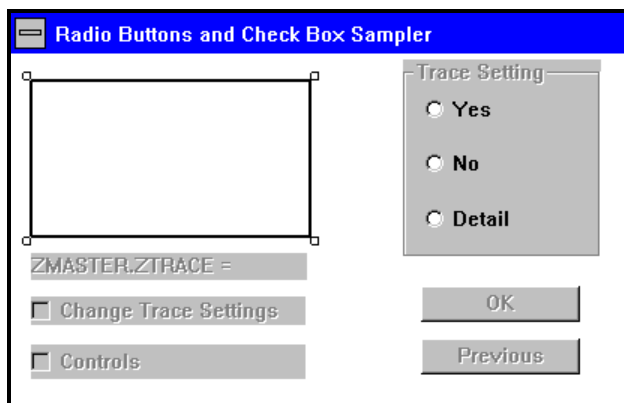
Check	<input type="text" value="Controls"/>	Value:	<input type="text" value="N"/>	<input type="button" value="Revert"/>
Object:	<input type="text"/>	Group ID:	<input type="text"/>	<input type="button" value="Cancel"/>
Variable:	<input type="text"/>	Action:	<input type="text" value="ENABLE"/>	<input type="button" value="Help"/>
Target:	<input type="text" value="controls"/>	Alignment:	<input type="text" value="Default"/>	
Tab Order:	<input type="text" value="10"/>	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input type="checkbox"/> Responds to <ENTER>

Notice the **Target** is the **Group ID** you specified for the push buttons.

## Add a Bitmap Image

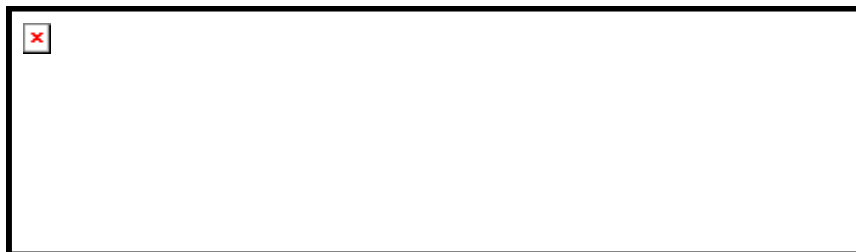
**Tip:** A graphic image can make a screen less intimidating. It can also give the screen your corporate look. The EDM Screen Painter allows you to add windows bitmap (.BMP) images to your screens. Images are automatically scaled to fit in the frame that you place on the screen. If the frame you place on the screen is significantly larger than the original image, the image may appear distorted.

To add a bitmap image, select the **Image** tool  from the toolbar, or choose **Image** from the **Controls** menu. The cursor is displayed as a paintbrush. Then, click and drag a rectangle proportional to the size of the image. An empty frame is displayed.



## Assign the Image Settings

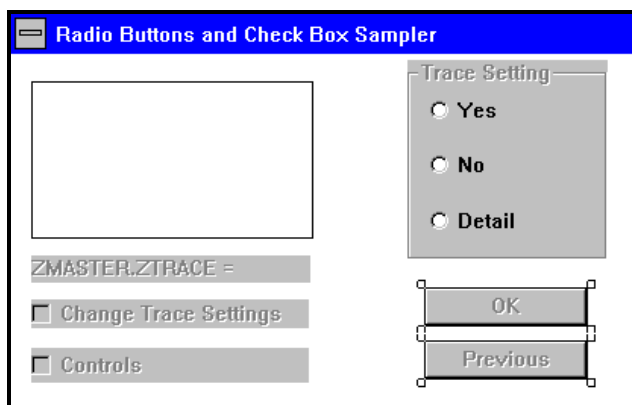
The only property you need to set for an image is the **Image** property. The image property contains the full path of the bitmap image, including the .BMP extension. In the **Image** field, type the full path of your logo, or use a windows bitmap. Windows bitmaps can be found in the Windows directory on your hard drive.



**Note:** You must update the **Image** property, and save the screen, if you move the source bitmap to a different directory.

## Re-size the Push Buttons

As mentioned earlier, the control you select first is the one to which the others are aligned. Select one of the push buttons. Then, while holding the SHIFT key, select the other. From the **Alignment** menu, choose **Align Same Width**. The buttons are now the same width.



To make the buttons the same height, from the **Alignment** menu choose **Align Same Height**. The buttons are now the same height.

## Align the Check Boxes to the Graphic Image

**Tip:** There is no need to make your check boxes the same height and width because check boxes are frame-less.

Select the image frame. The check boxes will be aligned to it. Then, while holding the SHIFT key, select the check boxes. From the **Alignment** menu, choose **Align Lefts**.



## Align the Push Buttons to the Group Box

Select the group box. Then, while pressing the SHIFT key, select the **OK** and **Previous** buttons. From the **Alignment** menu, choose **Align Lefts**.

## Save the Screen

From the **File** menu choose **Save**.

You are now ready to return to the Push Button and List Sampler screen and set the properties for the **Show Radio Panel** push button.

## Assign the Show Radio Panel Button Properties

From the **File** menu, choose **Open**. In the dialog box type A1SAMPLE and then choose **Open** or press ENTER. The Push Button and List Sampler screen is displayed.

Select the **Show Radio Panel** push button with your pointing device and enter the values as indicated in the table below. Note that the full value for **Push** should be Show Radio Panel.

Push	Show Radio Pa	Value:		Revert
Object:		Group ID:		Cancel
Variable:		Action:	PANEL	Help
Target:	A2SAMPLE	Alignment:	Center	
Tab Order:	3	<input checked="" type="checkbox"/> Active	<input checked="" type="checkbox"/> Visible	<input checked="" type="checkbox"/> Responds to <ENTER>

Notice that the **Responds to <ENTER>** check box is selected. With this check box selected, at runtime, you can press ENTER on your keyboard, instead of using your mouse, to choose this button. When you test the screens, you can press ENTER on your keyboard to toggle back and forth between the two screens.

From the **File** menu, choose **Save**. You are now ready to test the screens you just created.

## Test the Screens

Test your screens to make sure the controls work properly.



Choose the **TEST** tool from the tool bar, or choose **Test** from the **EDM\_System** menu. A new window opens and displays a fully functional screen as it will be displayed on the user's screen.

Choose the **Previous** and **Show Radio Panel** buttons several times. Feel free to reposition the screens in the screen edit window. You can have the screens appear in the center of the screen, side by side, on opposite ends of the screen, or any way you want.

**Tip:** Have screens display in the center of the screen. If you move screens in the screen edit window, be sure to save afterwards. A screen is displayed on the screen relative to where it is positioned in the screen edit window.

The screen you have just created does not include a **Cancel** push button. Not to worry, your users need only press the **Esc** key to achieve the same effect.

You are now ready to create screens for your users. See the “*Screen Painter Reference*” in this chapter for any details you need for creating your screens.

### Controlling Screen Sequencing

Two techniques allow you to control the sequence in which screens are displayed. In this section, and the examples that follow, we will refer to these two approaches as Technique 1 and Technique 2.

Technique 1 – the simpler technique – utilizes the EDM screen engine, EDMPNLGN.EXE, and controls the sequencing of screens using a command button whose action is PANEL. This technique does not invoke outside processing.

Technique 2 – the more complex technique – utilizes the EDM screen engine, EDMPNLGR.EXE, and controls screen sequencing using the ZPREXEC variable. This technique has the added capability of invoking external REXX methods. For more information on Z-named variables, see the “*EDM Screen Painter Variable Reference*” at the end of this chapter.

### EDM Screen Engines

Technique 1	Technique 2
EDMPNLGN.EXE where: <b>N</b> indicates that the screenengine does not call a REXX method.	EDMPNLGR.EXE where: <b>R</b> indicates that the screenengine calls a REXX method.

## Screen Sequencing Technique 1

### To Create Sequenced Screens Using Technique 1:

- 1 Make a push button on the screen whose action is PANEL.

Set its **Target** property to the name of the screen it is to invoke.

**Note:** You must create the screen to be invoked before you create the push button that launches it.

You have now finished sequencing screens without invoking REXX programming methods.

## Screen Sequencing Technique 2

More complex variable manipulation and evaluation can be accomplished by layering REXX programs between the display of screens.

Technique 2 utilizes the ZPREXEC variable, which automatically calls the REXX program that is specified for the variable.

Including a REXX program allows you to evaluate variables in addition to ZCMD, manipulate files, and perform other functions available to REXX programs.

This approach requires more memory than the non-REXX version of the screen program (technique 1) because the screen program and the REXX interpreter are co-resident at certain times during execution.

In technique 2, the REXX method has access to the screen and local object variables, and controls the program and screen flow by changing ZPCONT and ZPANEL.

### EDM Manager Selection Program

This sample program allows users to select an EDM Manager to which their EDM Client desktop will connect.

The first screen displays the default EDM Manager in the top half of the screen. A user can choose an alternate EDM Manager from the list box, or they can add a new Manager to the list of choices by choosing **Edit**.

When the user chooses **Edit**, the REXX method displays the EDM Manager Selection Editor screen. Information in this screen is written to an object that the REXX program creates, called MRGSEL.EDM. The user can define a new EDM Manager by choosing **Add**, and then typing over the “new manager” selection that is created.

You can add as many EDM Managers as you see fit. After the last EDM Manager has been entered, the user chooses **Close**. The REXX method displays the EDM Manager Selection Screen and the user selects the EDM Manager he or she wants from the list button.

When the user chooses **Activate**, the Screen Painter sets the EDM Manager connection information in the ZMASTER and ZADMIN objects. The user terminates the program by choosing **Close**.

To use this example in your enterprise, create the screens and REXX program as indicated on the pages that follow. Save them in your EDMLIB directory using the file names PMGRSEL.EDM and PMGRSEL2.EDM respectively. Then create a desktop icon named EDM MANAGER SELECTION that executes the following:

```
C:\EDMNT\EDMREXW MGRSEL.REX START
```

The EDM Manager Selection icon launches the REXX program. The REXX program launches the screens.



### EDM Manager Selection Screen

File name: PMGRSEL.EDM

**EDM Manager Selection**

Currently Selected Manager

Test Manager

TCP/IP Address: 204.7.83.59

Destination Socket: 1922

Administrator Logon ID: EDM\_MAST

Production Manager

New Manager Selection

Production Manager

TCP/IP Address: 1.1.1.99

Destination Socket: 1234

Administrator Logon ID: ADMIN

Select Edit Done

### EDM Manager Selection Editor Screen

File name: PMGRSEL2.EDM

**EDM Manager Selection Editor**

Manager Information

New Manager

TCP/IP Address: X.X.X.X

Destination Socket: 1234

Administrator Logon ID: ADMIN

Test Manager

New Manager

New Copy Delete Done

The table below lists the properties that were used to define the EDM Manager Selection screen. Note that all the controls in the screen are both **Active** and **Visible**, and the **Alignment** of all controls is set to **Default**.

### EDM Manager Selection Screen Control Properties

Control	Name	Object	Variable	Target	Action	Description
Group Box	Currently Selected Manager					Top group box
Static Text		ZMASTER	MGRNAME			EDM Manager name
Static Text	TCP/IP Address					Label
Static Text		ZMASTER	ZIPADDR			Actual TCP/IP address
Static Text	Destination Socket:					Label
Static Text		ZMASTER	ZDSTSOCK			Destination socket number
Static Text	Administrator Logon ID:					Label
Static Text		ZADMIN	ZUSERID			Logon Id
List Button	mgrsellb	MGRSEL	MGRNAME	MGRSOCK	PAGEHP	Selects the EDM Manager
Group Box	New Manager Selection					Bottom group box
Static Text		MGRSEL	MGRNAME			EDM Manager name
Static Text	TCP/IP Address:					Label
Static Text		MGRSEL	MGRIP			TCP/IP address
Static Text	Destination Socket:					Label
Static Text		MGRSEL	MGRSOCK			Destination socket number
Static Text	Administrator Logon ID:					Label
Static Text		MGRSEL	MGRADMID			Logon ID
Push Button	Select				EXIT	
Push Button	Edit				EXIT	
Push Button	Done				EXIT	

The table below lists the properties that were used to define the EDM Manager Selection Editor screen. Note that all the controls in the screen are both **Active** and **Visible**, and the **Alignment** of all controls is set to **Default**.

### EDM Manager Selection Editor Screen Control Properties

Control	Name	Object	Variable	Target	Action	Description
Group box	Manager Information					Group box
Edit		MGRSEL	MGRNAME			EDM Manager name
Static Text	TCP/IP Address:					Label
Edit		MGRSEL	MGRIP			EDM Manager TCP/IP address
Static Text	Destination Socket:					Label
Edit		MGRSEL	MGRSOCK			Destination socket

Control	Name	Object	Variable	Target	Action	Description
Static Text	Administrator Logon ID:					Label
Edit		MGRSEL	MGRADMID			Logon Id
List Box		MGRSEL	MGRNAME		PAGEHP	
Push Button	New				EXIT	
Push Button	Copy				EXIT	
Push Button	Delete				EXIT	
Push Button	Done				EXIT	

## MGRSEL.REX

```

/*****
/* MGRSEL.REX for PMGRSEL Panels */
/* This REXX program allows EDM Administrators to */
/* maintain a list of EDM Managers that can be */
/* connected to. If the parm passed to this */
/* program is "START", then the panel engine */
/* program EDMPNLGR.EXE is started to display the */
/* PMGRSEL panel. From then on, this program */
/* reacts to buttons pressed on the PMGRSEL and */
/* PMGRSEL2 panels. */
/*****

```

```

/* trace r */
call edmget 'zmaster'
parse arg parm
parm = translate(parm)

```

```

/*****
/* First time through: Display the first panel. */
/*****

```

```

if parm = "START" then
do
zmaster.zpanel = "PMGRSEL"
zmaster.zprexec = "MGRSEL.REX"
call FirstDisplay
end

```

```

/*****
/* Handle buttons pressed on the PMGRSEL panel. */
/*****

```

```

if zmaster.zpanel = "PMGRSEL" then
do
if zmaster.znamesel = "Done" then
do
zmaster.zpcont = 0

```

```

zmaster.znamesel = 0
zmaster.zpsel = 0
zmaster.zpanel = " "
zmaster.zprexec = " "
call edmset 'zmaster'
exit
end
if zmaster.znamesel = "Select" then
do
call edmget('mgrsel',zmaster.zpsel)
zmaster.zpheapno = zmaster.zpsel
zmaster.zipaddr = strip(mgrsel.mgrip)
zmaster.zdstsock = strip(mgrsel.mgrsock)
zmaster.mgrname = strip(mgrsel.mgrname)
call edmget('zadmin')
zadmin.zuserid = strip(mgrsel.mgradmid)
call edmset('zadmin')
call Redisplay
end
if zmaster.znamesel = "Edit" then
do
zmaster.zpanel = "PMGRSEL2"
zmaster.zpheapno = 0
call ReDisplay
end
end

```

```

/*****/
/* Handle buttons pressed on the PMGRSEL2 panel. */
/*****/

```

```

if zmaster.zpanel = "PMGRSEL2" then
do
if zmaster.znamesel = "New" then
do
call edmadd('mgrsel')
mgrsel.mgrname = "New Manager"
mgrsel.mgrip = x.x.x.x
mgrsel.mgrsock = 1234
mgrsel.mgradmid = "ADMIN"
call edmset('mgrsel')
call edmsort('mgrsel',mgrname)
call Redisplay
end
if zmaster.znamesel = "Copy" then
do
call edmget('mgrsel',zmaster.zpsel)
copyname = strip(mgrsel.mgrname)
copyip = strip(mgrsel.mgrip)

```

```

copysock = strip(mgrsel.mgrsock)
copyaid  = strip(mgrsel.mgradmid)
call edmadd('mgrsel')
mgrsel.mgrname = "Copy of " || copyname
mgrsel.mgrip   = copyip
mgrsel.mgrsock = copysock
mgrsel.mgradmid = copyaid
call edmset('mgrsel')
call edmsort('mgrsel',mgrname)
call Redisplay
end
if zmaster.znamesel = "Delete" then
do
call edmdelheap('mgrsel',zmaster.zpsel)
call edmset('mgrsel')
call Redisplay
end
if zmaster.znamesel = "Done" then
do
zmaster.zpanel = "PMGRSEL"
call Redisplay
end
end
/*****/
/* No action found, sort object and redisplay current panel */
/*****/
call edmsort('mgrsel',mgrname)
call Redisplay
exit

/*****/
/* FirstDisplay subroutine: */
/* Set variables to display the first panel. */
/* Start panel engine program asynchronously (nowait) */
/* Exit this program. */
/*****/

FirstDisplay:
zmaster.zpcont = 1
zmaster.znamesel = ''
zmaster.zpsel = 0
call edmset 'zmaster'
"nowait " || zmaster.zsysdrv || zmaster.zsysdir || "EDMPNLGR"
exit
return

/* Redisplay subroutine: */
/* Set variables to redisplay a panel. */
/* Exit to the panel engine, */
Redisplay:
zmaster.zpcont = 1

```

```

zmaster.znames1 = ''
zmaster.zpsel   = 0
call edmsset 'zmaster'
exit
return

```

## Screen Painter Reference

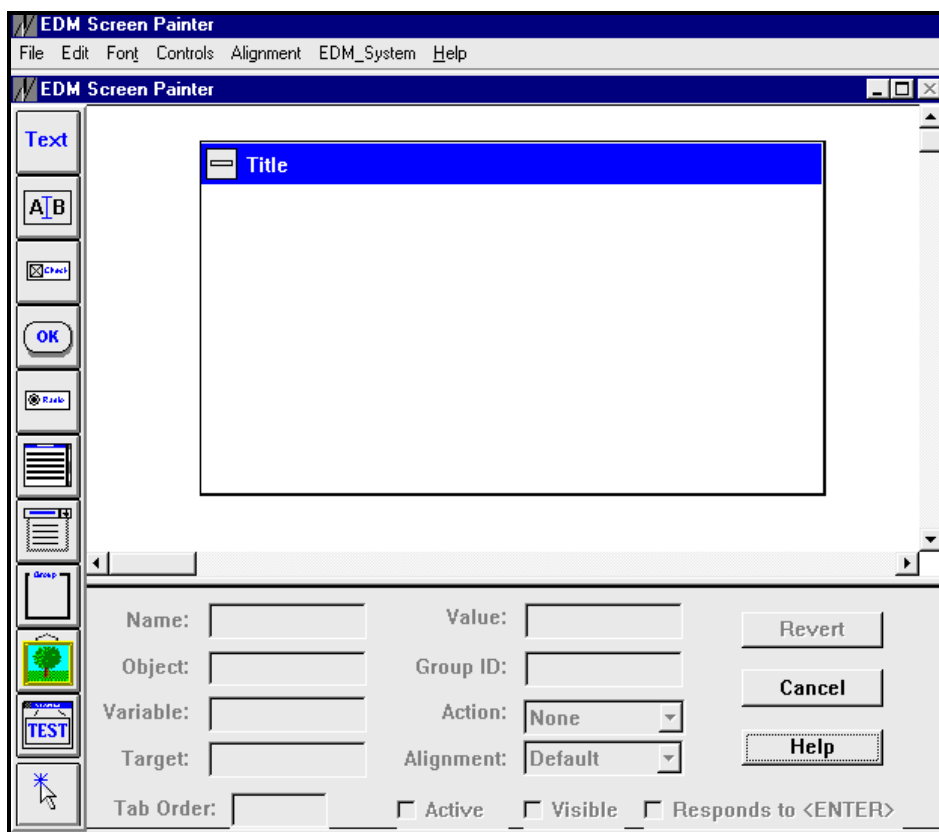
This section provides a reference on how to use each Screen Painter editing tool. This section also provides EDM reference information about the internal EDM variables that a REXX program can use to control screens.

## Opening the EDM Screen Painter

### To Launch the EDM Screen Painter:

- 1 From the EDM Administrator folder, choose **EDM Screen Painter**.

The EDM Screen Painter is displayed with a new screen ready for editing.

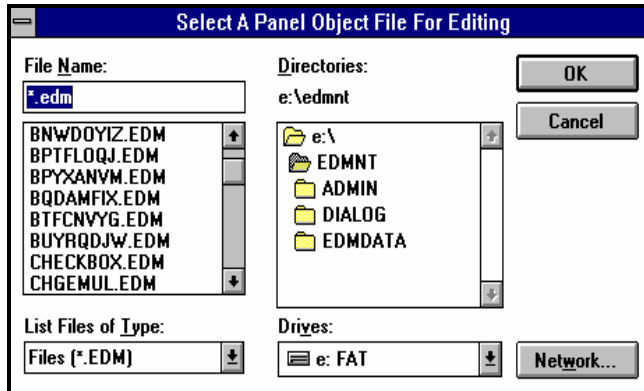


## Opening an EDM Screen Object

### To Open an EDM Screen Painter Object:

- 1 From the **File** menu, choose **Open**.

The Open Screen dialog box is displayed.



**Note:** EDM Screen Painter screens, like all EDM objects, are identified with the .EDM extension.

- 2 Select a screen and choose **OPEN** or press ENTER.

The screen you selected is displayed.

**Note:** The Version 4.x EDM Screen Painter, EDMPANEL, EDMPNLGN, and EDMPNLGR do not support screens created with the Version 3.x EDM Panel Manager.

## Opening a New Screen

### To Open a New Screen:

When you first start the EDM Screen Painter, a new blank screen entitled “Title” is displayed in the Screen Painter window.

- 1 To create a new screen, choose **New** from the **File** menu.

A new blank screen called “Title” is displayed:



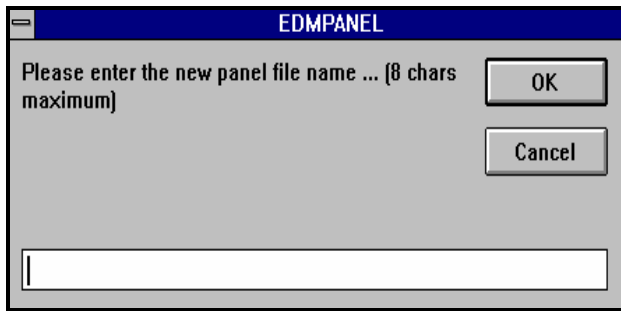
**Note:** If you choose New from the File menu, any screen you may have open for editing is closed.

## Saving a Screen

### To Save a Screen:

- 1 From the **File** menu, choose **Save**.

The following dialog box is displayed.



- 2 Type in the file name *without* the .EDM extension, and choose **OK**.

**Warning:** Do not use the name PNLTST when saving a screen. If you do, your screen will be overwritten the next time you use the EDM Screen Painter.

**Note:** The Screen Painter chooses the EDM directory for you. There is no need to specify the full file path.

## Closing Without Saving Changes

### To Close a Screen Without Saving Changes:

- 1 In the properties window, choose Cancel. Or, from the File menu, choose Close.

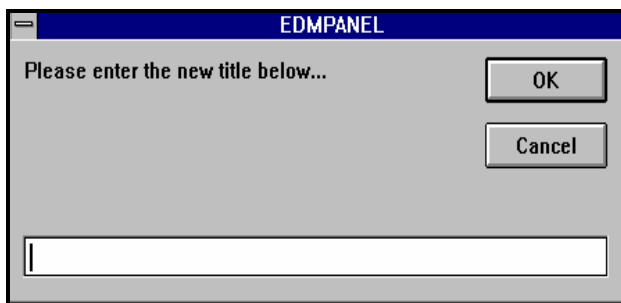
## Changing the Screen Title

By default, a new screen is labeled “Title” in the title bar.

### To Change the Title of a Screen:

- 1 Double-click the title bar of the screen, or choose **Set Title** from the **Controls** menu.

A dialog box is displayed.



- 2 Type the new title in the text box and choose **OK** or press ENTER.



## Inserting Static Text

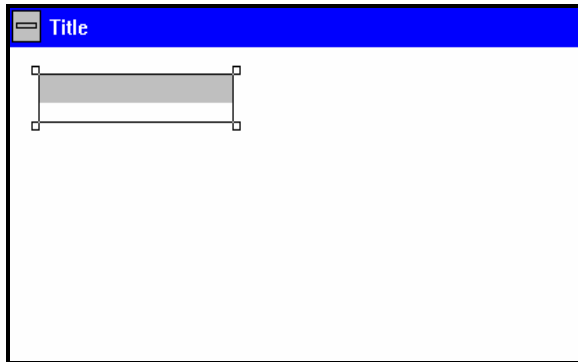
### To Insert Static Text:

- 1 Select the **Static Text** tool  from the toolbar, or choose **Static Text** from the **Controls** Menu.

The pointer is displayed as a static text field.

- 2 Drag a rectangle to contain the text on the screen.

A static text field the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.

**Note:** If the text you type does not fit in the static text field, you can enlarge the field with the **Select** tool.

- 3 Select the static text.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Static	The text to be displayed. Leave this blank if the text is coming from a variable in an object.	ZPNAME
Object	The object where the text to be displayed is located. (Optional)	ZPOBJECT
Variable	The variable where the text to be displayed is located. (Optional)	ZPVARNAM
Alignment	Justification for the text within the bounding rectangle (Left, Center, Right).	ZPJUST (L, C, R)
Visible	If this check box is checked, then the text is initially visible, if this check box is not checked, then the text is initially hidden. If this variable is not present, then the text will be visible.	ZPSHOW (Y, N)


## Changing Fonts

You can change the font and style of the text for each control on all supported platforms. However, the default system font is recommended for most applications.

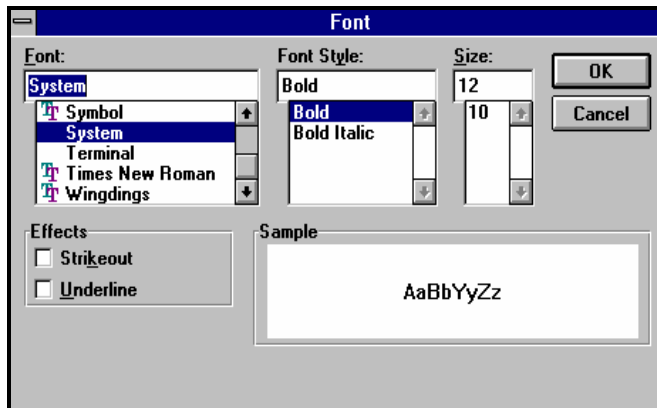
**Note:** The EDM Screen Painter displays screen controls in default system fonts if the operating system is unable to use the fonts you selected.

**Note:** You must change the font setting for each control that you want to display in a font other than the default system setting.

### To Change the Font of a Control:

- 1 Select the control with the **Select** tool .
- 2 Choose **Select...** from the **Font** menu.


A dialog box is displayed.



- 3 Select the desired settings and choose **OK** or press ENTER.

### Inserting an Edit Box

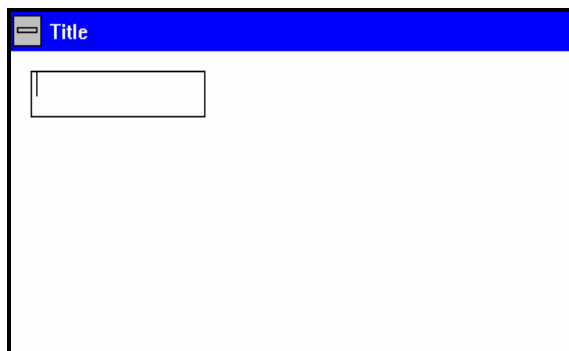
#### To Insert an Edit Box:

- 1 Select the **Edit** tool  from the toolbar, or choose **Edit** from the **Controls** menu.

The pointer is displayed as an edit box.

- 2 Click and drag a rectangle of the desired size on the screen.

An edit box the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.

- 3 Select the edit box.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Edit	The name of this control for reference in groups. (optional)	ZPNAME
Object	The object where the contents of the field are stored.	ZPOBJECT
Variable	The variable where the contents of the field are stored.	ZPVARNAM
Visible	If this check box is checked, then the edit box is initially visible, if this check box is not checked then the edit box is initially hidden. If this variable is not present, then edit box will be visible.	ZPSHOW (Y, N)
Active	If this check box is checked, then the edit box is initially enabled, if this check box is not checked, then the edit box is initially disabled (grayed out). If this variable is not present, then edit box will be enabled.	ZPACTIVE (Y, N)
Alignment	Justification for the text within the edit box (Left, Center, Right) on those platforms that natively support justification of edit box.	ZPJUST (L, C, R)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD

## Inserting a Push Button

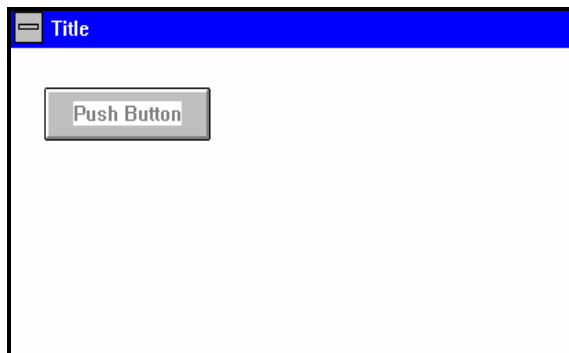
### To Insert a Push Button:

- 1 Select the **Push Button** tool  from the toolbar, or choose **Push Button** from the **Controls** menu.

The pointer is displayed as a push button.

- 2 Drag a rectangle of the desired size on the screen.

A push button the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.

- 3 Select the push button.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Push	The text label on the push button.	ZPNAME
Action	Specifies what the button will do when selected. The valid values are, EXIT, CANCEL, SHOW, ENABLE, SCREEN, LAUNCH, PREVHP, or NEXTHP. See the table below for a description of these actions.	ZPACTION
Target	This variable contains information to process the action above. This will be different for different actions.	ZPACTASN
Alignment	Justification for the text of the push button (Left, Center, Right) on those platforms that natively support push button text justification. <b>Center</b> is recommended.	ZPJUST (L, C, R)
Responds to <ENTER>	Only one push button in the screen can have this check box checked. If this check box is checked then the user can push this push button by pressing ENTER on the keyboard. This is generally checked for a push button whose action is EXIT.	ZPMULTI (Y, N)
Visible	If this check box is checked, then the push button is initially visible, if this check box is not checked, then the push button is initially hidden. If this variable is not present, then the push button will be shown.	ZPSHOW (Y, N)
Active	If this check box is checked, then the push button is initially enabled, if this check box is not checked then the push button is initially disabled (grayed out). If this variable is not present, then the push button will be enabled.	ZPACTIVE (Y, N)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD

The table below describes the actions that you can attribute to a push button.

### Actions for Push Buttons

Action	Description
EXIT	Saves the contents of the screen, and closes the screen.
CANCEL	Closes the screen without saving changes.
SHOW	Causes the control specified in the <b>Target</b> field to be visible or invisible, depending on its current state. If the <b>Target</b> field contains a <b>Group ID</b> for a group of controls in the screen, then that group of controls is made invisible or visible.
ENABLE	Causes the control specified in the <b>Target</b> field to be enabled or disabled (grayed out), depending on its current state. If the <b>Target</b> field contains a <b>Group ID</b> for a group of controls in the screen, then that group of controls is enabled or disabled.
PANEL	Exits the current screen and displays the screen specified in the <b>Target</b> field. If the <b>Target</b> field is blank, then the <b>Object</b> and <b>Variable</b> fields specify the variable that contains the name of the screen to be displayed.  Advanced: The <b>Target</b> specifies the name of either a list box, or a list button, and <b>Object</b> and <b>Variable</b> point to a variable in a multi-instance object. The name of the screen to be displayed is based on the selection in the list box or list button.
LAUNCH	Launches the application specified in the <b>Target</b> field. If the <b>Target</b> field is blank, then the <b>Object</b> and <b>Variable</b> fields specify the variable that contains the name of the file to be launched.  Advanced: The <b>Target</b> field contains the name of either a list box, or a list button, and the <b>Object</b> and <b>Variable</b> fields point to a variable in a multi-instance object. The multi-instance object contains the name of the file to be launched.
PREVHP	Pages to the previous instance in a multi-instance object (if not on the first instance), and reloads all the controls that use that object. The object to be paged is specified in the <b>Object</b> field. The variable specified in the <b>Value</b> field is displayed.
NEXTHP	Pages to the next instance a multi-instance object ( if not on the last instance), and reloads all the controls that use that object. The object to be paged is specified in the <b>Object</b> field. The variable specified in the <b>Value</b> field is displayed.

## Inserting a Check Box

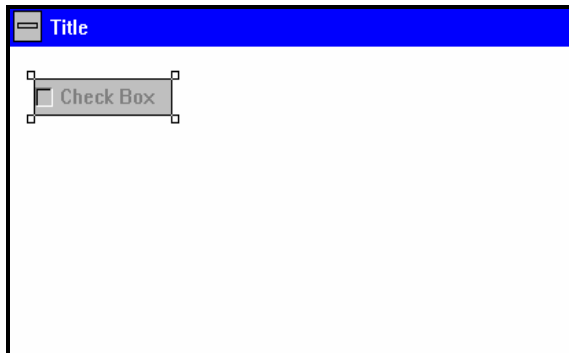
### To Insert a Check Box:

- 1 Select the **Check Box** tool  from the toolbar, or choose **Check Box** from the **Controls** menu.

The pointer is displayed as a check box.

- 2 Drag a rectangle of the desired size on the screen. The rectangle must be big enough to accommodate the check box label.

A check box the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.


- 3 Select the check box.
- 4 Assign properties as indicated in the table below.

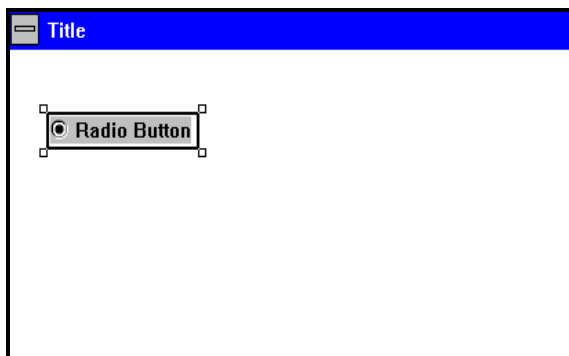
Property	Description	Internal Variable Name (Advanced)
Check	The label for the check box.	ZPNAME
Object	The object in which the result of the check box is stored.	ZPOBJECT
Variable	The variable in which the result of the check box is stored.	ZPVARNAM
Value	The default setting for this check box, checked (on) or unchecked(off), if the above variable doesn't exist. Valid values are Y or 1, indicating checked (on), or N or 0 indicating unchecked (off).	ZPVALUE (Y, 1, N, 0)
Action	If this property is set to SHOW or ENABLE, then the control specified in the Target field is displayed or enabled when the check box is checked (on), and hidden or disabled (grayed out) when the check box is unchecked (off). (Optional)	ZPACTION
Target	If the Action property is set to SHOW or ENABLE, then this property specifies the name (ZPNAME) of the control to show, hide, enable, or disable. (Optional)	ZPACTASN
Alignment	Justification for the check box (Left, Center, Right) on those platforms that support check box text justification.	ZPJUST (L, C, R)
Visible	If this check box is checked, then check box is initially visible, if this check box is not checked then the check box is initially hidden. If this variable is not present, then the check box will be shown.	ZPSHOW (Y, N)
Active	If this check box is checked, then the check box is initially enabled, if this check box is not checked, then the check box is initially disabled (grayed out). If this variable is not present, then the check box will be enabled.	ZPACTIVE (Y, N)

Property	Description	Internal Variable Name (Advanced)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD

## Inserting a Radio Button

### ➤ To Insert a Radio Button:

- 1 Select the **Radio Button** tool  from the toolbar, or choose **Radio Button** from the **Controls** menu.  
The pointer is displayed as a radio button.
- 2 Drag a rectangle on the screen. A rectangle large enough to contain the radio button will be created.



The pointer is displayed as the select tool.

- 3 Select the radio button.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Radio	The label for the radio button.	ZPNAME
Object	The object in which the result of the group or cluster of radio buttons is stored.	ZPOBJECT
Variable	The variable in which the result of the group or cluster of radio buttons is stored.	ZPVARNAM
Value	The value to be stored in the above variable when this radio button is selected.	ZPVALUE
Group ID	A unique string identifying all radio buttons in the group.	ZPGRPID
Action	If this property is set to SHOW or ENABLE, then the control specified in the Target field is made visible or enabled when the radio button is selected, and hidden or disabled (grayed out) when another radio button in the group is selected. (Optional)	ZPACTION
Target	If the Action field is set to SHOW or ENABLE, then this property specifies the name (ZPNAME) of the control to display, hide, enable, or disable. (Optional)	ZPACTASN
Alignment	Justification for the radio button (Left, Center, Right) on those platforms that natively support radio button justification.	ZPJJUST (L, C, R)

Property	Description	Internal Variable Name (Advanced)
Visible	If this check box is checked, then the radio button is initially visible, if this check box is not checked then the radio button is initially hidden. If this variable is not present, then the radio button will be shown.	ZPSHOW (Y, N)
Active	If this check box is checked, then the radio button is initially enabled, if this check box is not checked then the radio button is initially disabled (grayed out). If this variable is not present, then the radio button will be enabled.	ZPACTIVE (Y, N)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD

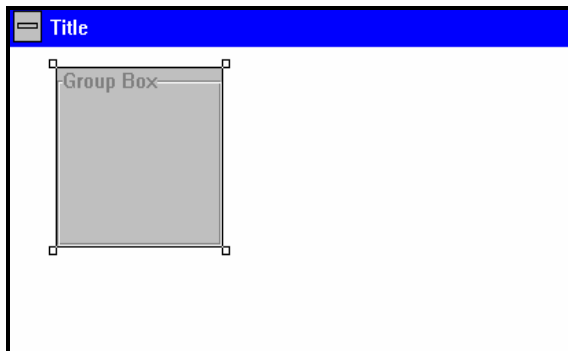
## Inserting a Group Box

### ➤ To Insert a Group Box:

- 1 Select the **Group Box** tool  from the toolbar, or choose **Group Box** from the **Controls** menu.

The pointer is displayed as a group box.

- 2 Drag a rectangle of the desired size on the screen.



A group box the size of the rectangle you created is displayed. The pointer is displayed as the select tool.

- 3 Select the group box.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Group	The text label for the group box.	ZPNAME
Alignment	Justification of the label for the group box (Left, Center, Right ).	ZPJUST (L, C, R,)
Visible	If this check box is checked, then the group box is initially visible, if this check box is not checked, then the group box is initially hidden. If this variable is not present, then group box will be visible.	ZPSHOW (Y, N)

## Inserting a List Box

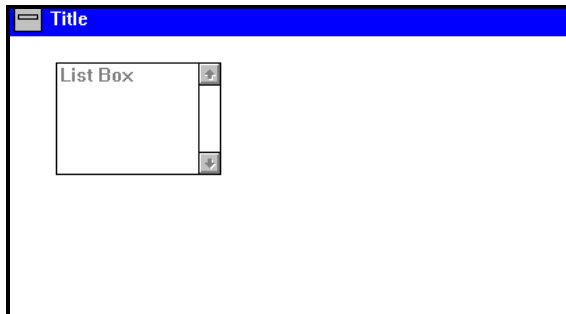
### ➤ To Insert a List Box:

- 1 Select the **List Box** tool  from the toolbar, or choose **List Box** from the **Controls** menu.

The pointer is displayed as a list box.

- 2 Drag a rectangle of the desired size on the screen.

A list box the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.

- 3 Select the list box.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
List Box	The name of the control for specifying it as a target. (Optional)	ZPNAME
Object	The multi-instance object from which the list is built.	ZPOBJECT
Variable	The multi-instance variable from which the list is built.	ZPVARNAM
Action	If this property is set to PAGEHP, then selecting an item from the list box pages to the corresponding instance in the object specified in the Object field, and redraws any other controls that get their data from that object. (Optional)	ZPACTION (PAGEHP)
Value	Specifies the action to be taken when the user double-clicks on a selection in the list box. Valid values are PANEL or LAUNCH. (Optional)	ZPVALUE (PANEL, LAUNCH)
Target	If either LAUNCH or PANEL is specified in the Value field, this field specifies the name of the variable in the object specified in the Object field that contains the name of the screen to display or the file (.EXE, .EXT, .REX, .BAT) to launch. (Optional)	ZPACTASN
Visible	If this check box is checked, then the list box is initially visible. If this check box is not checked, then the list box is initially hidden. If this variable is not present, then the list box will be visible.	ZPSHOW (Y, N)
Active	If this check box is checked, then the list box is initially enabled. If this check box is not checked, then the list box is initially disabled (grayed out). If this variable is not present, then the list box will be enabled.	ZPACTIVE (Y, N)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD



## Inserting a List button

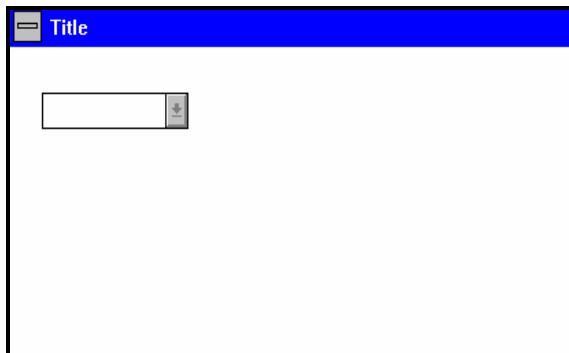
### ➤ To Insert a List Button:

- 1 Select the **List button** tool  from the toolbar, or choose **List Button** from the **Controls** menu.

The pointer is displayed as a list button.

- 2 Drag a rectangle of the desired size on the screen. Draw the rectangle long enough to include the text it will display.

A list button the size of the rectangle you created is displayed.




The pointer is displayed as the select tool.

- 3 Select the list button.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
List	The name of the control for specifying it as a target. (Optional)	ZPNAME
Object	The multi-instance object from which the list is built.	ZPOBJECT
Variable	The multi-instance variable from which the list is built.	ZPVARNAM
Action	If this property is set to PAGEHP, then selecting an item from the list pages to the corresponding instance in the object specified in the Object field, and redraws any other controls that get their data from that object. (Optional)	ZPACTION (PAGEHP)
Visible	If this check box is checked, then the list button is initially visible. If this check box is not checked, then the list button is initially hidden. If this variable is not present, then the list button will be visible.	ZPSHOW (Y, N)
Active	If this check box is checked, then the list button is initially enabled, If this check box is not checked, then the list button is initially disabled (grayed out). If this variable is not present, then the list button will be enabled.	ZPACTIVE (Y, N)
Tab Order	Specifies the order in which the control is selected when the TAB key is pressed. The default value is based on the order in which you added the control to the screen.	ZPTABORD

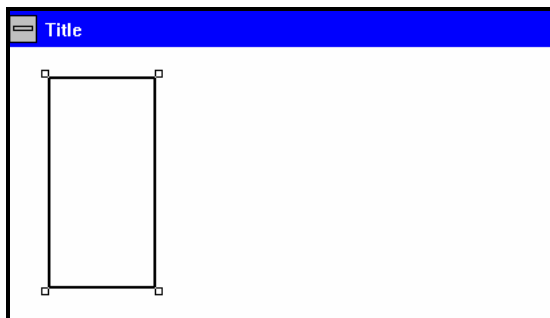
## Inserting a Bitmap Image

### ➤ To Insert a Bitmap Image:

- 1 Select the **Image** tool  from the toolbar, or choose **Image** from the **Controls** menu. The pointer is displayed as a paint brush.
- 2 Drag a rectangle to contain the image on the screen.

**Note:** The image is scaled to fill the entire frame.

A frame the size of the rectangle you created is displayed.



The pointer is displayed as the select tool.

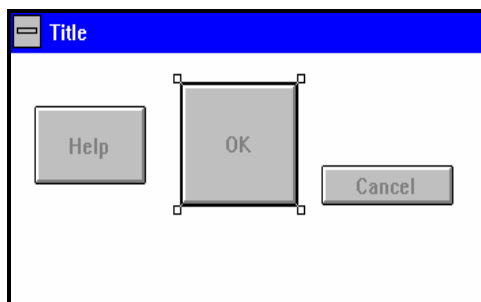
- 3 Select the frame.
- 4 Assign properties as indicated in the table below.

Property	Description	Internal Variable Name (Advanced)
Image	The full path of the bitmap image to be displayed, including the .BMP extension.	ZPNAME
Visible	If this check box is checked, then the image is initially visible. If this check box is not checked, then the image is initially hidden. If this variable is not present, then the image will be visible.	ZPSHOW (Y, N)

## Selecting Controls

### ➤ To Select a Single Control:

- 1 Using your mouse or pointing device, click on a control.

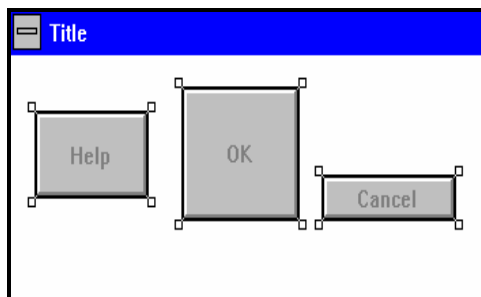


Notice the sizing handles on a selected control.

**Note:** You can drag the sizing handles to resize the control.

### ➤ To Select Multiple Controls:

- 1 Using your mouse or pointing device, click on a single control.
- 2 While holding down the SHIFT key, click on any other control you want to select.



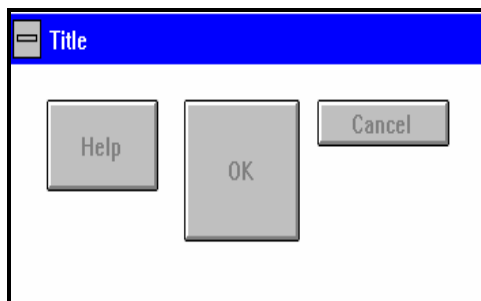
Notice the sizing handles on all three controls.

**Note:** All selected controls move together. To move only one control you must first deselect the others.

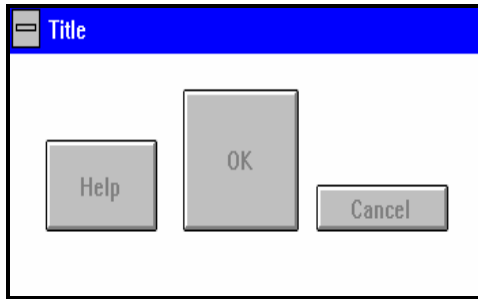
## Horizontally Aligning Controls

The EDM Screen Painter allows you to line up controls horizontally, so that the:


Top borders of the controls are aligned.

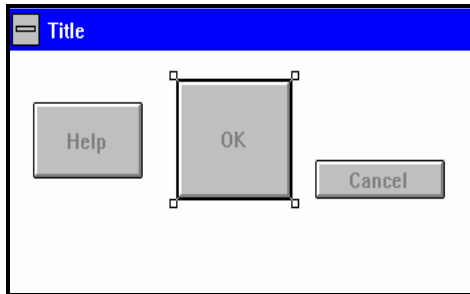


Bottom borders of the controls are aligned.



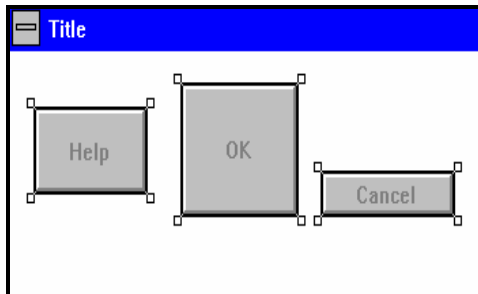
### ➤ To Horizontally Align Controls:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Select a control to be used as a reference. This control does not change position. You will align the other controls to the reference control.



In the screen above, notice that only the **OK** button is selected.

- 3 Select the remaining controls you want to align by holding down the **SHIFT** key, and then clicking on them with your mouse.




In the screen above, notice that all three controls are selected. The **Help** and **Cancel** buttons will be aligned to the **OK** button because it was selected first.

- 4 From the **Alignment** menu, choose **Align Tops** or **Align Bottoms**

The selected controls are now aligned by their top or bottom borders.

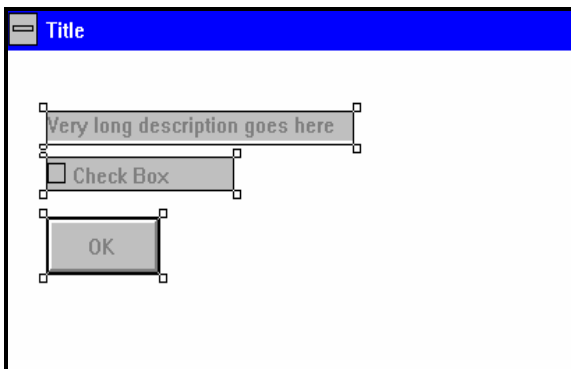
## ➤ To Evenly Space Controls Horizontally:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Press the SHIFT key and select all controls you want to evenly space horizontally.
- 3 From the **Alignment** menu, choose **Align Even Horizontal**.

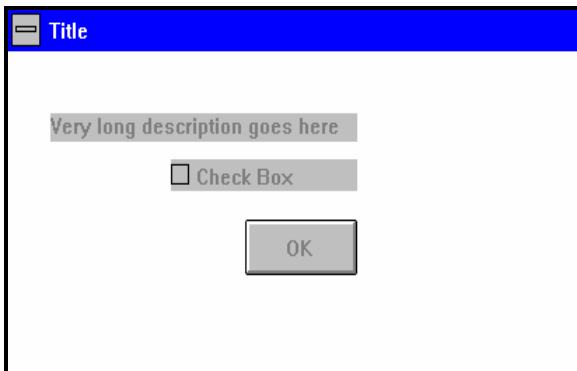
## Vertically Aligning Controls

The EDM Screen Painter allows you to line up controls vertically, so that the:

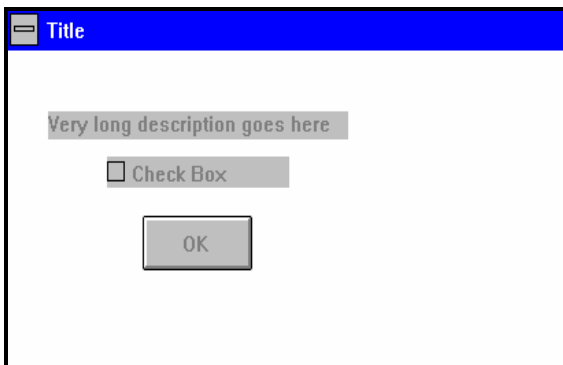
Left borders of the controls are aligned.




Right borders of the controls are aligned.

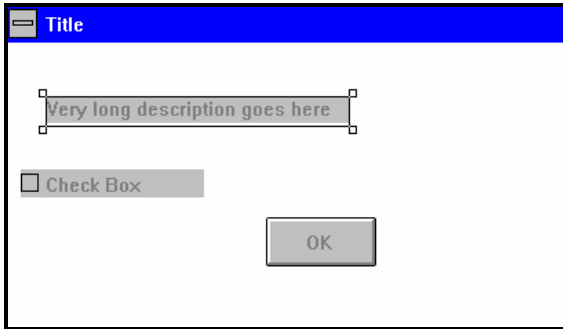


Centers of the controls are aligned.



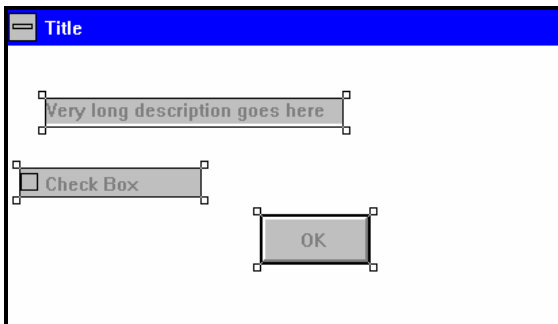
### ➤ To Vertically Align Controls:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Select a control to be used as a reference. This control does not change position. You will align the other controls to the reference control.



In the screen above, notice that only the static text is selected.


- 3 Select the remaining controls you want to align by holding down the **SHIFT** key, and then clicking on them with your mouse.



In the screen above, notice that all three controls are selected. The push button and the check box will be aligned to the static text, because it was selected first.


- 4 From the **Alignment** menu, choose **Align Lefts**, **Align Rights**, or **Align Centers**.

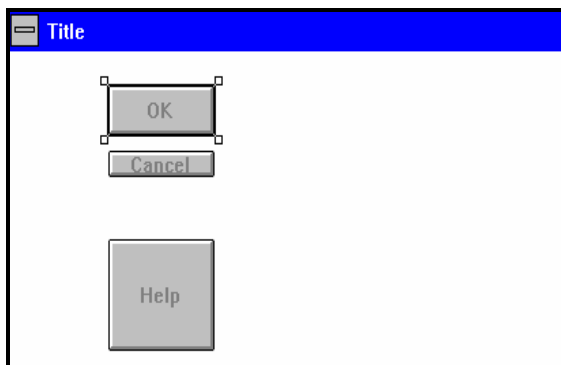
### ➤ To evenly space controls vertically:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Press the **SHIFT** key and select all controls you want to evenly space vertically.
- 3 From the **Alignment** menu, choose **Align Even Vertical**.

## Making Controls the Same Height

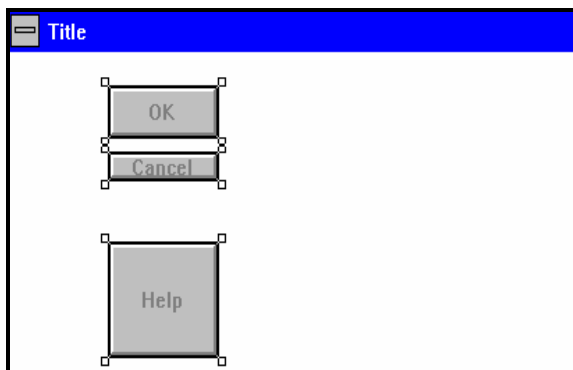
### ➤ To Make Controls the Same Height:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Select a control to be used as a reference. This control does not change size. You will make the other controls the same height as this control.

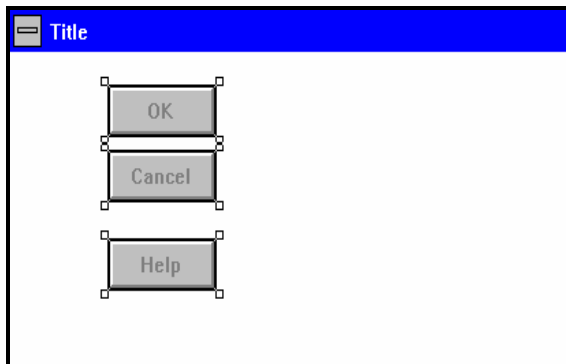


In the screen above, notice that only the **OK** button is selected.

- 3 Select the remaining controls you want to resize by holding down the SHIFT key, and then clicking on them with your mouse.




In the screen above, notice that all three controls are selected. The **Help** and **Cancel** buttons will be resized to the height of the **OK** button, because it was selected first. From the **Alignment** menu, choose **Align Same Height**.

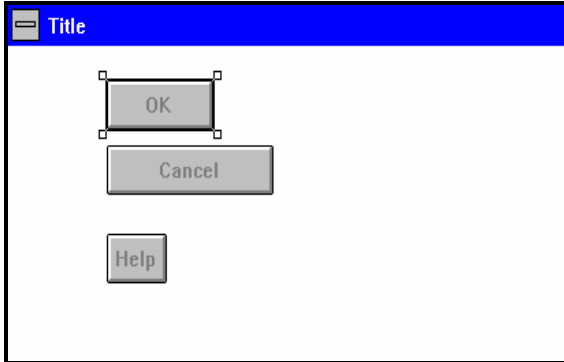


## Making Controls the Same Width

---

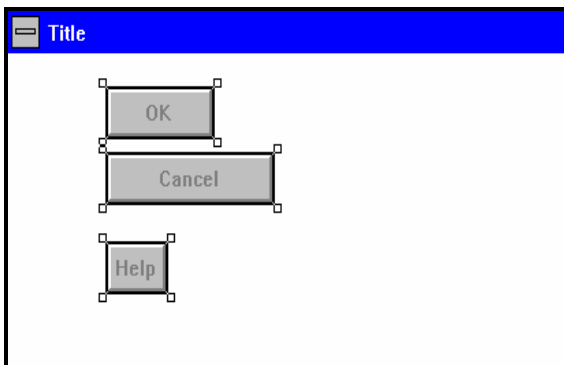
### ➤ To Make Controls the Same Width:

- 1 Select the **Select** tool  from the toolbar, or choose **Select** from the **Controls** menu.
- 2 Select a control to be used as a reference. This control does not change size. You will make the other controls the same width as this control.



In the screen above, notice that only the **OK** button is selected.

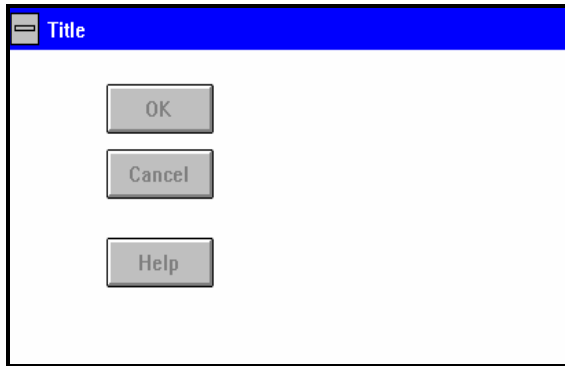
- 3 Select the remaining controls you want to resize by holding down the SHIFT key, and then selecting them with your mouse.



In the screen above, notice that all three controls are selected. The **Help** and **Cancel** buttons will be resized to the width of the **OK** button because it was selected first.

- 4 From the **Alignment** menu, choose **Align Same Width**.






## Testing a Screen

### ➤ To Test a Screen (View a Screen in Real Time Mode):

- 1 If you are testing a screen that uses REXX processing, choose **Use REXX for Test** from the **EDM\_System** menu. This utilizes the REXX Screen Engine, EDMPNLGR.EXE, at run time.

If you are testing a screen that does not use REXX processing, use the default setting (**Use REXX for Test** is not checked). This uses the non-REXX screen engine, EDMPNLGN.EXE, at run time.

- 2 Choose the **TEST** tool  from the tool bar, or choose **Test** from the **EDM\_System** menu.

A new window opens and displays a fully functional screen as it will be displayed on the user's screen.

**Note:** When a screen is displayed in real time mode it is fully operational. If you enter any values in the screen, your desktop EDM objects may be altered.

**Note:** Advanced

When editing with the EDM Screen Painter, screen contents are stored in a temporary **screen called** PNLTST.EDM. When you choose **Test**, the screen engine executes PNLTST.EDM, not the filename assigned to the screen.

## Displaying Variable Names Without Resolving Objects

When you type a variable name in the **Name** field of the properties window using symbolic substitution (for example, &(ZMASTER.ZUSERID)), the screen engine resolves the variable during runtime. You may also have noticed that when you type a variable name in an edit box, using symbolic substitution, the variable is resolved the next time the screen is called.

The EDM Screen Painter provides you with a way to display variable names without resolving the object.

### ➤ To Display Variable Names Without Resolving Symbolic Substitution for the Screen:

- 1 Specify the symbolics in the **Name** field of the controls that you do not want resolved.

**Note:** EDM always resolves any data specified in the **Object** and **Variable** fields in the properties window.

- 2 From the **EDM\_System** menu, choose **Do Not Resolve Screen**.

## EDM Screen Painter Variable Reference

---

The list of Z-named variables that can be set for an individual screen are detailed below. You will find these variables quite useful when you write REXX methods that manipulate and sequence EDM screen objects.

### EDM Screen Painter Z-Named Variable Reference

Variable Name	Value Options / Description
ZNAMESEL	The name of the button that the user pressed to exit the screen.
ZPANEL	Specifies the next screen object to be displayed. Define and set this variable in ZMASTER.
ZPCONT	0 = End 1 = Continue to screen currently specified in ZSCREEN When ZPCONT is set to 1, the screen display program displays the next screen (specified in ZSCREEN) instead of exiting.
ZPHEAPNO	For use with multi-instance objects. Define and set this variable in ZMASTER if you want an instance other than instance 0 to be the default selected instance.  For example, add and set ZPHEAPNO to 2 if you want the third item in a list box to be selected when the screen is displayed.
ZPREXEC	Specifies the REXX method to be executed. Define and set this variable in ZMASTER.
ZPSAVE	0 = Cancel 1 = Exit Specifies how the user closed the screen.
ZPSEL	The relative instance number of the selected item in a list box or list button when ENTER was pressed.

## Internal Variables

---

The EDM Screen Painter uses XVT native controls to display a screen. The object that drives the screen is a multi-instance object, which contains:

- One instance to describe the window.
- One instance for each control on the screen.

One or more instances to describe any EDM objects the screen will access.

Each instance in an EDM screen object must have a ZPCTYPE variable. The following table lists the valid values for ZPCTYPE.

### Valid ZPCTYPE Values

Value	Description
CKBOX	This instance describes a check box.
EDIT	This instance describes a text input field.
GRPBOX	This instance describes a labeled box that can surround other controls.
IMAGE	This instance describes a bitmap image.
LISTBOX	This instance describes a list of items in a scrollable box.
LISTBUT	This instance describes a drop-down list of choices.
PBUTTON	This instance describes a push button.

Value	Description
RBUTTON	This instance describes a radio button.
PANEL	This instance describes the window for the screen.
TEXT	This instance describes a static text field.

## Panel

For a PANEL instance, the following variables are significant.

Variable	Description
ZPTOP, ZPLEFT, ZPBOTTOM, ZPRIGHT	The bounding rectangle for the screen window (in pixels).
ZPNAME	The window title for the screen.
ZPFONT, ZPFSIZE, ZPFSTYLE	The default font, font size, and font style (for example, bold or italic) for the screen.
ZPBKCLR	The Background color for the screen. The valid values are: BLACK, BLUE, CYAN, DARK GRAY, GRAY, GREEN, LIGHT GRAY, MAGENTA, RED, WHITE, and YELLOW.
ZPFGCLR	The default text color for all controls in the screen. This variable can be overridden if it is set in an individual control's instance. The valid values are: BLACK, BLUE, CYAN, DARK GRAY, GRAY, GREEN, LIGHT GRAY, MAGENTA, RED, WHITE, and YELLOW.

Warning: Do not change ZPTOP, ZPLEFT, ZPBOTTOM, or ZPRIGHT by any means other than dragging the screen in the Workspace, or by moving its sizing handles.

## Troubleshooting

There are three log files you can access for diagnosing any problems you may encounter with EDM screen objects:

- NEWPANEL.LOG
- *panelname*.LOG

PNLTST.LOG

The NEWPANEL log is generally less than ten lines in length, and is written every time you launch the screen engine. Check this log if a screen does not display.

Once the screen engine knows which screen is going to be loaded, all log information is written to *screenname*.LOG. (*screenname* indicates the filename of the screen.) Check this log if the screen displays but does not function properly.

PNLTST.LOG is the log file for the screen you are testing with the TEST feature of the EDM Screen Painter.

If your screens call EDM REXX methods, you may want to check the REXX log, EDMREXXW.LOG.



# Using EDM Extended Batch Commands

This chapter provides you with the information you need to understand and use the EDM Extended Batch facility. This chapter includes information on using various commands to control Extended Batch processing, defining and controlling the Extended Batch user window, running Extended Batch files, guidelines on syntax and usage, and considerations for developing Extended Batch procedures using symbolic substitution.

For your convenience, a quick reference table is provided with brief definitions of the batch commands contained in the main reference. The main reference guide contains complete Extended Batch definitions with parameters, syntax, and usage examples.

## Platform Availability

The EDM Extended Batch Commands are available on the following platforms:

Platform Availability				
DOS	<input checked="" type="checkbox"/>		NCR MP/RAS	<input checked="" type="checkbox"/>
Windows	<input checked="" type="checkbox"/>		Sun OS	<input checked="" type="checkbox"/>
Windows NT	<input checked="" type="checkbox"/>		Unix AIX	<input checked="" type="checkbox"/>
Windows 95	<input checked="" type="checkbox"/>		Unix HP-UX	<input checked="" type="checkbox"/>
OS/2	<input checked="" type="checkbox"/>		Unix Solaris	<input checked="" type="checkbox"/>
Macintosh	<input checked="" type="checkbox"/>		UnixWare	<input checked="" type="checkbox"/>
NetWare	<input checked="" type="checkbox"/>			

The EDM Extended Batch commands function identically in all supported platforms unless otherwise noted.

## The EDM Extended Batch Commands at a Glance

---

This facility is essentially the DOS command language, or the OS/2 command language, with extensions for accessing and manipulating EDM objects.

EDM commands can be stored in a batch file: an unformatted text file with an .EXT extension that contains both environment-specific and EDM-specific instructions. Commands in an Extended Batch program are processed sequentially.

.EXT files can include any of the command statements available in the executing environment's conventional batch language, such as .BAT files in DOS and .CMD files in OS/2. EDM Extended Batch is available in DOS, Windows, Windows NT, Windows 95, OS/2, Macintosh, and Unix; the language is not case-sensitive in any environment.

The EDM Extended Batch Commands can utilize EDM Client variable substitution. With variable substitution, you can reference EDM objects, variables, and methods.

Extended Batch programs can invoke other EDM facilities, such as EDM REXX methods for sophisticated logic, and EDM screens for user interaction.

You can develop platform-independent systems management applications from procedural applets to interactive applications, such as a help-desk system, by combining the powerful features of EDM's Extended Batch, Screen Painter, REXX language extensions, and variable substitution.

## The Extended Batch Graphical User Interface

The EDM Extended Batch facility provides several options for communicating process information to the user during process execution:

- **USER (OR TASK) WINDOWS**

(Not available in EDMCONS or EDMMNTR.) The Extended Batch facility provides a user window (the task window) that is created and customized by the EDM developer. For example, you can set the size and position of the user window (`SET_USER_WINDOW`), and dynamically change text displayed in the user window (`WRITE_USER_WINDOW`).

- **USER BUTTONS**

(Not available in EDMCONS or EDMMNTR.) Buttons can be included in the user window. They allow a user to optionally abort an Extended Batch procedure. The user window buttons include **Abort**, **OK** and **Cancel**. **Abort** is displayed in the user window by default.

- **ECHO WINDOWS**

You can call attention to specific information, or prompt a user for input, before continuing execution by placing the information or prompt in its own window. To interact with the user in this manner, Extended Batch provides `ECHO` and `ASK` functions.

- **BATCH COMPLETION INDICATOR**

(Not available in EDMCONS or EDMMNTR.) By using the `SETIND` and `STARTIND` command, you can set and display progress indicators, or histograms, during the course of EDM execution. This command is available for Extended Batch processing in addition to the text information provided in the user and echo windows.

## Defining the User Window

A user window can be defined to be as large as the coordinate space allows. The coordinate system for the Extended Batch user window is based on pixels. The number of pixels available depends on the display device's resolution. For a typical VGA configuration, the coordinate space is 640x480. The origin, position (0,0), for the user window is the upper left corner of the screen.

Because the user window is actually the task window, all other display objects, text, echo boxes, etc., are defined within the coordinate space of the current user window. Thus, regardless of the size or position of a user window on a screen, the origin for placing display objects in the user window is always (0,0). Care should be taken to define the location of display objects within the display area of the user window, as fields defined outside of the window's coordinate space are not visible.

By default, Extended Batch does not display the user window. To display the user window use the `DISPLAY_USER_WINDOW` command.

## Controlling the User Window

Many of the appearance characteristics of the user window can be changed at run time, including background color and displayed text. Extended Batch commands are also provided to dynamically delete and recreate the user window. You can have the user window recreated during execution when you want to change static attributes, such as the user window's size or position, or the position of user buttons.

The size, position, and title can be set for each of these buttons. In order to have the buttons display as desired, their attributes should be set prior to the DISPLAY\_USER\_WINDOW command. While the program is running, it is possible to delete the user window, redefine buttons and redisplay the window.

When an Extended Batch procedure is executed from a graphical environment, the user window is displayed with only the **Abort** button visible.

Choosing **Abort** pauses the Extended Batch procedure, and the **OK** and **Cancel** buttons are displayed. Choosing **OK** will terminate the Extended Batch procedure, and **Cancel** will resume Extended Batch processing and hide the **Cancel** and **OK** buttons.

## Using Symbolic Notation

With EDM Extended Batch, you can use symbolic notation that enables the variables contained within an object class to be referenced during Extended Batch execution.

The symbolic reference in an .EXT file is replaced by the variable value that it finds in a local object. Symbolic reference to an EDM object from an .EXT file is made using the following syntax:

**Syntax**        &(OBJECT.VARIABLE.INSTANCE)

The notation, &(Object.Variable.Instance), is dynamically substituted with the contents or current value of the referenced class instance. If the Variable name is unique, the symbolic notation can be abbreviated further:

**Syntax**        &VARIABLE

If substitution fails for any reason, the expression remains unchanged.

### Parameters

Parameter Name	Explanation	Required or Optional
object	A local object such as ZMASTER, ZSERVICE and ZRSOURCE, or a valid local object class name.	Required
variable	A specific variable of the class.	Required
instance	The specific instance number (also referred to as a "heap") of a multi-dimension object. If not included, the current instance is used in the resolution.	Optional

The essential power of a symbolic reference within a batch procedure is its ability to construct highly generalized and reusable batch processing routines that can act on, or respond to, EDM object variables.

A very simple example of Extended Batch processing would be a batch file that copies files from a *source drive:\pathname* to a *target drive:\pathname*. The source and target *drive:\pathname* values are contained in local EDM object variables.

A batch file that accomplishes the task can be written using a number of techniques, but several advantages are realized by using an EDM Extended Batch file.

The .EXT file, relying on the class variables to be resolved and substituted, can be written without regard for the specific source or target. Therefore, the contents of the local object class variables determine any copy operation in EDM Extended Batch.

The appropriate local objects can be delivered to the EDM Client during the EDM Client Connect Process. Thus, the source and target of the copy operation can be controlled for a client by manager processing.

The .EXT file below can accomplish the above objectives:

### Example

```
@ECHO OFF
XCOPY & (HICLIENT.SPATH.0) *.* & (ELRIOX.TPATH.0)
```

In the example above, the files in the directory indicated by the SPATH variable, which is contained in the HICLIENT object, is copied to the directory indicated by the TPATH variable in the ELRIOX object.

This .EXT file continues to perform the file copy operation independent of changes to the source and target directories. The source and target directories can be changed by manipulating the objects delivered to the client, or by using the EDM Screen Painter to develop and present a panel tied to the objects.

In the example below, the COMMAND object is a variable, VAR1. This variable can contain a fully qualified DOS, Windows, OS/2, Macintosh, or Unix command so that different users, or the same user at different times, can execute tailored commands or programs.

### Example

```
& (COMMAND.VAR1)
```

The above example illustrates how even the command flow can be altered by the contents of a local object. This single-line .EXT file can theoretically execute any command.

Also notice that an .EXT file, executed in a Windows environment by EDMCONSW, can execute Windows modules. An .EXT file, executed in OS/2, can execute OS/2 modules. This can not be accomplished using DOS .BAT files, which run only in a DOS window, and, therefore, cannot call specific Windows or OS/2 modules and routines.

EDM Extended Batch capitalizes on the use of variable substitution to implement logical navigation. Whenever possible, you should use variable substitution to create fully qualified command flows. Doing so makes your Extended Batch file more portable across platforms and users.

## Command Line Modifiers

Command line modifiers, listed below, enhance your control over the desktop during the execution of the Extended Batch file. Keep in mind that modifiers can be grouped together but must be placed before the command to be executed.



## EDM Extended Batch Command Line Modifiers

Parameter Name	Explanation
HIDE and SHOW	HIDE prohibits the display of the startup and execution of a program, or implementation of a command. HIDE is the default for DOS commands run under Windows. If you wish to see the commands as they execute, use the SHOW modifier.
INBACK	For Macintosh only. Same as HIDE above.
TRAP and RELEASE	These two modifiers control what the user can or cannot do during the execution of a command line. Use the TRAP modifier to disable any mouse or keyboard input. The default is RELEASE.
WAIT and NOWAIT	These modifiers instruct the batch executor either to WAIT for the completion of a command line or to immediately process the next statement (NOWAIT). The default is WAIT.
FULLSCR	For OS/2 only. This modifier executes a program or command line in a full screen window, rather than the default window size.
MINIMIZE	This modifier executes a program or command in a minimized window.

### Example

```
NOWAIT SHOW ERASE C:\MYDIR\RESUME.DOC
```

Executed in OS/2, the statement in the above example tells the batch executor to show the user that the OS/2 ERASE command (delete file) is being processed, and to return control to the user (or process the next instruction) without waiting for its completion.

## Running Extended Batch Files

Extended Batch programs are executed by an EDM-provided driver. Extended Batch drivers are included in EDM Client for DOS, Windows, Windows NT, Windows 95, OS/2, Macintosh, and Unix. EDM Extended Batch is not case sensitive in any environment.

If the directory containing the EDM Client executables is included in a path statement, the EDM Extended Batch driver, as well as all other EDM Client functions, can be run without regard for the current directory setting. Alternatively, in Windows or OS/2, a Program Information File (PIF) can specify the directory containing the EDM executable modules.

All status messages for a given invocation of an EDM Extended Batch driver are logged into an ASCII log file. The log file is created with a filename matching the filename of the .EXT file executed and a file extension of .LOG (for example: MYBATCH.EXT will have an associated log file named MYBATCH.LOG). The log is placed in the same directory in which the .EXT file resides. A new log file is written for each run of an .EXT file. If the contents of the log need to be retained, then *filename*.LOG must be copied to an alternate file between invocations.

## EDMCONS

EDMCONS is the Extended Batch driver for DOS.

**Note:** Extended Batch GUI (graphical user interface) commands are not available in EDMCONS.

You can type EDMCONS from the DOS command prompt, followed by the Extended Batch file name, or use any valid DOS standard program invocation technique to run the program.

Status information about EDMCONS execution is logged in the EDMCONS.LOG file. Log files created during DOS Extended Batch execution are not assigned a name matching the Extended Batch file.

## Example

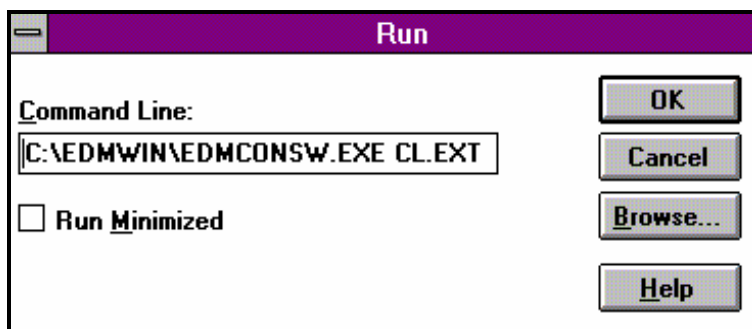
```
EDMCONS C:\EDMDOS\CMDFILE.EXT
```

**Note:** If executing EDMPNLnn.EXE (the panel display utility) from within an .EXT file, note that the companion EDMPNLnn.FRL file must be present in the current working directory. This means that the .EXT file must explicitly change its directory to the EDMSYS library before executing the panel program.

## EDMCONSW

EDMCONSW is the Extended Batch driver for the Windows environments and OS/2.

In Windows, Windows NT, and Windows 95, an Extended Batch program is run from the **Run** dialog box. To do so, launch the **Run** dialog box, then type the path of EDMCONSW.EXE, followed by the path and name of the Extended Batch program. Choose **OK**. (The following example shows an example command line batch execution within Windows 3.5.1.)



In OS/2, an Extended Batch program can be run from the OS/2 command prompt. You can provide a parameter specifying the target .EXT file when entering your command. If no argument is provided on the command line or in the PIF, an input file selection dialog box is displayed.

## EDMDOS.PIF

To optimize the performance of MS-DOS commands and programs in Windows, Windows NT, or Windows 95 Extended Batch programs, specify EDMDOS.PIF on the command line, followed by the command. To close the DOS window and return control to the Extended Batch program when the DOS command terminates, use the /C parameter.

## Example

```
EDMDOS.PIF /C DOSBATCH.BAT
```

## EDMCONSX

EDMCONSX is the Extended Batch driver for Macintosh and Unix platforms that support a graphical interface (AIX, HP-UX, NCR MP/RAS, Sun Solaris).

There are two ways to execute an EDM Extended Batch program in Macintosh OS:

- Choose the Extended Batch Program from its folder.
  - Use the EDM-supplied Launcher utility to emulate a command line interface. You must use Launcher if you want to enter any parameters.

➤ **To Execute an Extended Batch by Using Launcher:**

- 1 Choose Launcher from the EDM folder. A screen with a command prompt is displayed.



- 2 Type `EDMCONSX` and press `ENTER`. A parameter prompt is displayed.

**Warning:** Do not type your Extended Batch program name on the same command line as `EDMCONSX`, as Launcher can accept only one command or parameter at a time. After typing `EDMCONSX` press `RETURN`. Then type your program name and press `RETURN`.

- 3 At the parameter prompt, type the Extended Batch program name and then press `ENTER`.
- 4 At the next parameter prompt, type the next parameter value, if any, and then press `ENTER`

Note: Press `ENTER` after typing *each* parameter.

- 5 After you have entered your last parameter value, at the next parameter prompt, press period (`.`) and then press `RETURN`. A synchronous prompt is displayed.
- 6 At the synchronous prompt, type `N` and then press `ENTER` to execute your Extended Batch program asynchronously. Choosing asynchronous execution ensures that Launcher will return control to the user by displaying another command prompt, immediately, without waiting for the Extended Batch program to complete execution.

If you type `Y` and press `ENTER`, Launcher will not return control to the user (display another command prompt) until execution of the Extended Batch program is complete.

A background prompt is displayed.

- 7 Type `N` and then press `ENTER` for foreground execution. Choosing foreground execution ensures that your Extended Batch program is executed before any other active application.

If you type `Y` and press `ENTER`, your Extended Batch program will share processing time with all other applications that are running.

The Extended Batch program you specified will execute.

If Launcher displays:

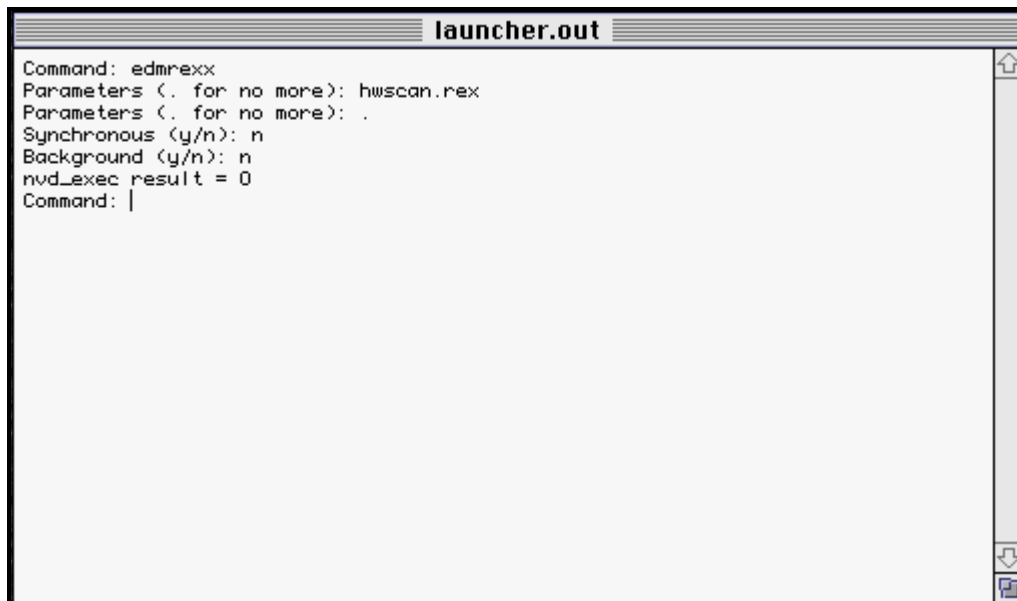
```
nvd_exec result = 0
```

the Extended Batch program executed successfully.

If Launcher displays:

```
nvd_exec result = 8
```

execution failed.



- 8 To exit Launcher, at the command prompt type `quit` and then press ENTER; or choose **Quit** from the **File** menu.
- 9 To execute an Extended Batch program on a Unix platform that supports a graphical interface (AIX, HP-UX, NCR MP/RAS, Sun Solaris), type `EDMCONSX` followed by the full path of the Extended Batch program, and then press ENTER.

## Example

```
/edmunix/admin/edmconsx /edmunix/admin/test1.ext
```

## EDMMNTR

When installing EDM on AIX, HP-UX, NCR MP/RAS, and Sun Solaris environments, client desktops can be installed with either EDMCONSW or EDMMNTR.

EDMMNTR is the Extended Batch driver for non-GUI Unix platforms (Sun OS, Unixware).

**Note:** Extended Batch GUI commands are not available in EDMMNTR.

- 1 To execute an Extended Batch program on a non-GUI Unix platform, type EDMMNTR followed by the full path of the Extended Batch program, and then press ENTER.

### Example

```
/edmunix/admin/edmmntr /edmunix/admin/test1.ext
```

## Extended Batch Usage Notes

- DOS, Windows, and OS/2 programs can be started from within an Extended Batch file by supplying the appropriate program name. If the specified program is a Windows or OS/2 program, an additional window appears. If the program is a DOS executable (.EXE) file, a DOS window is displayed. The DOS window closes when the DOS .EXE has finished execution.

**Note:** When using the EDMCONSW driver in Windows and OS/2, commands are "shelled" to a DOS or OS/2 window, respectively. This is impractical for many commands such as drive and directory changes, which are lost upon exit of the operating shell. This technique is effective, however, for permanent operations such as the native ERASE command.

- The ERRORLEVEL variable is not available to Windows .EXT files.
  - To retain a DOS window for inspection before it is closed to the Windows environment, a PAUSE statement can be inserted in the .EXT file containing the DOS commands.

## EDM Extended Batch Quick Reference

---

The following table contains a summary of the EDM Extended Batch commands. More detailed descriptions of the commands and their associated parameters are provided in the pages following the summary table.

### EDM Extended Batch Commands

Command Name	Description
ASK	Used to solicit a yes or no response from a user.
CALL	Executes a batch program from within another.
CHDIR or CD	(Novell) Changes working directory.
COPY	(Novell and Mac) Copies a file from a source to a destination.
DELAY	Pauses execution for a maximum of 60 seconds.
DEL_VAR	Removes a variable from an object.
DESTROY_IND **	Closes the Batch completion indicator.
DESTROY_USER_WINDOW **	Closes the user window.
DISABLE_MENU_EXIT **	(Macintosh only) Causes any Quit command from the File menu to be ignored.

Command Name	Description
DISPLAY_USER_WINDOW **	Displays the user window.
DROP_OBJECT	Removes an object from an in-storage list.
ECHO	Clears the current echo window text, then displays the new text specified.
EDM_ADD_INST	Adds an instance to a specified object.
EDM_COPY	Copies a single file to the location you specify.
EDM_DEL_INST	Removes an instance from an object.
EDM_DELETE	Deletes the file you specify.
EDM_ERASE	Deletes the file you specify.
EDM_MOVE	Moves a single file to the location you specify. Renames a single file.
EDM_SET_INST	Makes a specific instance active in an object.
ENABLE_MENU_EXIT **	(Macintosh only) Causes any Quit command from the File menu to terminate execution.
EXIST	Specifies a true condition if the specified file exists. Use with IF.
MKDIR or MD	(Novell) Creates a directory.
MOVE	(Macintosh only) Moves a file to another directory or file.
RELEASE	Enables mouse and keyboard input during execution.
REM	Indicates a remark.
RENAME	(Novell and Mac) Renames a file.
RMDIR	(Novell) Deletes a directory.
SET_ABORT_TITLE **	Allows you to set the text in the Abort button.
SET_ABORT_WINDOW **	Allows you to set the size and position of the Abort button.
SET_CANCEL_TITLE **	Allows you to set the text in the Cancel button.
SET_CANCEL_WINDOW **	Allows you to set the size and position of the Cancel button.
SET_CONTINUE_TITLE **	Allows you to set the text in the OK button.
SET_CONTINUE_WINDOW **	Allows you to set the size and position of the OK button.
SET_DECRYPT_ON	Prevents confidential variables, messages, and information from being written to log file.
SET_DECRYPT_OFF	Resumes writing to log file.
SET_ECHO_TITLE **	Sets the text in the echo window title bar.
SET_ECHO_WINDOW **	Sets the size and position of the echo window.
SET_IND_TITLE **	Sets the text in the completion indicator title bar.
SET_PROCESS_NAME **	(Windows NT and OS/2 only) Serializes the use of an Extended Batch file.
SET_USER_BACKGROUND	Changes the background color of the user window.
SET_USER_FOREGROUND	Changes the text color in the user window.
SET_USER_TITLE	Sets the text in the user window title bar.
SET_USER_WINDOW	Sets the size and position of the user window.
SET_VAR	Sets the value of a variable in a specified heap in an object.
SETIND **	Sets the percentage value in the completion indicator.
SINGLE_STEP_ON	Displays each command line before execution.
STARTIND **	Displays the completion indicator.
TRAP	Disables any mouse or keyboard input during execution.
WRITE_USER_WINDOW	Displays text in the user window.

\*\* Indicates not available in EDMCONS or EDMMNTR.

## EDM Extended Batch Command Reference

---

This section provides you with details for specifying the EDM Extended Batch commands. Each command is documented, including all the information required to use it, as follows:

- Command name.
- Syntax.
- Command description.
- Parameter name and explanation.
  - Usage examples.

### ASK

---

**Syntax** ASK Text\_String

**Description** The ASK command takes the question given in the Text\_String parameter and places it in a dialog box that includes **Yes** and **No** buttons. Execution of the Extended Batch procedure pauses until the user has responded to the dialog box. The user's response is placed in &ZRSP for interrogation by subsequent Extended Batch logic.

#### Parameters

Parameter Name	Explanation	Required or Optional
text_string	The text of the question to be asked. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

The ASK command is provided to easily solicit a **Yes** or **No** response from a user during Extended Batch execution. For more complex user interaction, use the EDM Screen Painter.

#### Example

```
ASK YOU HAVE MAIL IN YOUR IN-BOX; START THE MAIL PROGRAM?
```

If the user chooses **Yes**, &ZRSP is set to Yes; if **No** is chosen, &ZRSP is set to No.

### CALL

---

**Syntax** CALL File\_Name

**Description** The CALL command executes an EDM Extended Batch program from within another without causing the first to terminate.

## Parameters

Parameter Name	Explanation	Required or Optional
file_name	Specifies the name of the EDM Extended Batch program you want to execute. Specify a fully qualified path or use symbolic substitution.	Required

## Example

```
CALL MYOWN.EXT
```

## CHDIR or CD

---

**Syntax** CHDIR Directory

**Description** This command is available for Novell only. The CHDIR command changes the working directory.

## Parameters

Parameter Name	Explanation	Required or Optional
directory	Specifies the directory that will become the working directory.	Required

## Example

```
CHDIR Test1
```

## COPY

---

**Syntax** COPY (Source, Destination)

**Description** This command is available for Macintosh only. The COPY command will copy a file from a source, to a destination. The file may be renamed in the process. The original file will remain untouched.

## Parameters

Parameter Name	Explanation	Required or Optional
source	Specifies the name and location of the file you would like to copy. Specify a fully qualified path or use symbolic substitution.	Required
destination	Specifies the destination you would like to copy the file to. Specify a fully qualified path	Required



	or use symbolic substitution.	
--	-------------------------------	--

## Example

```
COPY (ZMASTER.EDM, Test1:ZMASTER.EDM)
```

## COPY

---

**Syntax** COPY File\_Name1 File\_Name2

**Description** This command is available for Novell only. The COPY command will create a copy of a file in the same directory. The original file will remain untouched.

### Parameters

Parameter Name	Explanation	Required or Optional
file_name1	Specifies the name of the file you would like to copy.	Required
file_name2	Specifies the name of the copy.	Required

## Example

```
COPY test1.txt test2.txt
```

## DELAY

---

**Syntax** DELAY (*S*)

**Description** The DELAY command pauses execution for *S* seconds.

### Parameters

Parameter Name	Explanation	Required or Optional
s	Specifies the number of seconds. Valid range of <i>S</i> is between 1 and 60. The default value is 1.	Optional

## Example

```
DELAY (30)
```

## DEL\_VAR

---

**Syntax** DEL\_VAR (Object\_Name,Variable)

**Description** The DEL\_VAR command deletes a specified variable from an object.

### Parameters

Parameter Name	Explanation	Required or Optional
object_name	A valid EDM object. <i>Object_Name</i> can be up to 8 characters in length.	Required
variable	A valid EDM variable. <i>Variable</i> can be up to 8 characters in length.	Required

### Example

```
DEL_VAR (MYOBJ,MYVAR)
```

## DESTROY\_IND

---

**Syntax** DESTROY\_IND

**Description** The DESTROY\_IND command closes the Batch completion indicator. The Batch completion indicator must be displayed before this command can be issued.

This command is not available in EDMCONS or EDMMNTR.

### Example

```
DESTROY_IND
```

## DESTROY\_USER\_WINDOW

---

**Syntax** DESTROY\_USER\_WINDOW

**Description** The DESTROY\_USER\_WINDOW command makes the user window invisible. DESTROY\_USER\_WINDOW preserves the contents of the echo and user windows. The Batch completion indicator is not preserved and will not be reopened with the DISPLAY\_USER\_WINDOW call.

The user window must be displayed before this command can be issued.

This command is not available in EDMCONS or EDMMNTR.

## Example

```
DESTROY_USER_WINDOW
```

---

## DISABLE\_MENU\_EXIT

**Syntax**           DISABLE\_MENU\_EXIT

**Description**      This command is available for Macintosh only. The DISABLE\_MENU\_EXIT command causes any Quit command from the File menu to be ignored.

## Example

```
DISABLE_MENU_EXIT
```

---

## DISPLAY\_USER\_WINDOW

**Syntax**           DISPLAY\_USER\_WINDOW [NOABORT]

**Description**      The DISPLAY\_USER\_WINDOW command displays the user window. Once displayed, the user window can be temporarily or permanently made invisible (destroyed). (See DESTROY\_USER\_WINDOW on page 263.) Attribute command calls for the user window, such as SET\_USER\_TITLE, have no effect while the user window is displayed.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
NOABORT	When specified, NOABORT suppresses the display of the user window ABORT buttons. If not specified, the default is to display the buttons.	Optional

## Example

```
DISPLAY_USER_WINDOW
```

---

## DROP\_OBJECT

**Syntax**           DROP\_OBJECT (*Object\_Name*)  
DROP\_OBJ (*Object\_Name*)

**Description**      The DROP\_OBJECT command removes an object from an in-storage list.

## Parameters

Parameter Name	Explanation	Required or Optional
object_name	A valid EDM object. <i>Object_Name</i> can be up to 8 characters in length.	Required

## Example

```
DROP_OBJECT (MYOBJ)
```

## ECHO

---

**Syntax**            *ECHO Text\_String*

**Description**        The ECHO command is issued multiple times during an Extended Batch process. Each successive ECHO command clears the current echo window text, then displays the new text specified.

## Parameters

Parameter Name	Explanation	Required or Optional
text_string	The text to be displayed in the echo window. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

## Example

```
ECHO THE SERVER IS NOT AVAILABLE. PLEASE TRY LATER.
```

The above example produces the following output:

Batch File Echo Display
THE SERVER IS NOT AVAILABLE. PLEASE TRY LATER.

## EDM\_ADD\_INST

---

**Syntax**            *EDM\_ADD\_INST (Object\_Name)*

**Description**        The EDM\_ADD\_INST command adds an instance to a specified object.

## Parameters

Parameter Name	Explanation	Required or Optional
object_name	A valid EDM object. <i>object_name</i> can be up to 8 characters in length.	Required

## Example

```
EDM_ADD_INST (MYOBJ)
```

## EDM\_COPY

---

**Syntax** EDM\_COPY (*Source, Destination*)

**Description** The EDM\_COPY command copies a single file to the location you specify. EDM\_COPY is faster and more efficient than shelling out to the native operating system and is especially useful for copying logs.

## Parameters

Parameter Name	Explanation	Required or Optional
source	Specifies the location and name of the file you want to copy. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required
destination	Specifies the location and name of the file to which you want to copy. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required

## Example

```
EDM_COPY (C:\EDMWIN\TEST.LOG, H:\CLIENT\TEST.LOG)
```

## EDM\_DEL\_INST

---

**Syntax** EDM\_DEL\_INST (*Object\_Name*)

**Description** The EDM\_DEL\_INST command removes the currently active instance from an object.

## Parameters

Parameter Name	Explanation	Required or Optional
object_name	A valid EDM object. <i>object_name</i> can be up to 8 characters in length.	Required

## Example

```
EDM_DEL_INST (MYOBJ)
```

## EDM\_DELETE

---

**Syntax** EDM\_DELETE (*File\_Name*)

**Description** The EDM\_DELETE command deletes a single file that you specify. EDM\_DELETE is faster and more efficient than shelling out to the native operating system and is especially useful for deleting logs.

There is no functional difference between EDM\_DELETE and EDM\_ERASE.

### Parameters

Parameter Name	Explanation	Required or Optional
file_name	Specifies the location and name of the file you want to delete. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required

## Example

```
EDM_DELETE (C:\EDMWIN\TEST.LOG)
```

## EDM\_ERASE

---

**Syntax** EDM\_ERASE (*File\_Name*)

**Description** The EDM\_ERASE command deletes a single file that you specify. EDM\_ERASE is faster and more efficient than shelling out to the native operating system and is especially useful for deleting logs.

There is no functional difference between EDM\_DELETE and EDM\_ERASE.

### Parameters

Parameter Name	Explanation	Required or Optional
file_name	Specifies the location and name of the file you want to delete. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required

## Example

```
EDM_ERASE (C:\EDMWIN\TEST.LOG)
```

## EDM\_MOVE

---

**Syntax** EDM\_MOVE (*Source, Destination*)

**Description** The EDM\_MOVE command moves a single file to the location you specify. This command can also be used to rename a file. *Source* and *Destination* must reside on the same drive. EDM\_MOVE is faster and more efficient than shelling out to the native operating system and is especially useful for moving and renaming logs.

### Parameters

Parameter Name	Explanation	Required or Optional
source	Specifies the location and name of the file you want to move. Also specifies the name if the file you want to rename. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required
destination	Specifies the new location or name of the file. Specify a full path or use symbolic substitution. Do not use wildcard characters.	Required

*Source* and *Destination* must reside on the same drive.

### Example

```
EDM_MOVE (H:\EDMWIN\USER1\CONNECT.LOG,H:\ADMIN\USER1CON.LOG)
```

The above example moves and renames the log. Note that both paths are on the same drive.

## EDM\_SET\_INST

---

**Syntax** EDM\_SET\_INST (*Object\_Name, Heap\_Number*)

**Description** The EDM\_SET\_INST command makes a specified instance of an object the currently active instance.

### Parameters

Parameter Name	Explanation	Required or Optional
object_name	A valid EDM object. <i>object_name</i> can be up to 8 characters in length.	Required
heap_number	Specifies the relative position of the heap or instance in the object. <i>heap_number</i> must be an integer	Required

Parameter Name	Explanation	Required or Optional
	between 0 and the number of heaps in <i>object_name</i> .	

## Example

```
EDM_SET_INST (MYOBJ, 4)
```

## ENABLE\_MENU\_EXIT

---

**Syntax**            ENABLE\_MENU\_EXIT

**Description**      This command is available for Macintosh only. The ENABLE\_MENU\_EXIT command causes any Quit command from the File menu to terminate execution of the EDM Extended Batch program.

## Example

```
ENABLE_MENU_EXIT
```

## ERASE or DEL

---

**Syntax**            ERASE *Directory*

**Description**      This command is available for Novell only. The ERASE command deletes the directory passed to it as a parameter.

## Parameters

Parameter Name	Explanation	Required or Optional
directory	Specifies the directory that will be deleted.	Required

## Example

```
ERASE Test1
```

## EXIST

---

**Syntax**            IF EXIST (*File\_Name*) *Command*

**Description**      The EXIST condition specifies a true condition if the specified file exists.



## Parameters

Parameter Name	Explanation	Required or Optional
file_name	Specifies the name of the file you want to test. Specify a fully qualified path.	Required
command	Specifies the command that should be executed if EXIST evaluates true.	Required

## Example

```
IF NOT EXIST (C:\AUTOEXEC.BAT) ECHO Can't find AUTOEXEC.BAT
```

## MKDIR or MD

---

**Syntax**      MKDIR *Directory*

**Description**      This command is available for Novell only. The MKDIR command will create a directory.

## Parameters

Parameter Name	Explanation	Required or Optional
directory	Specifies the name of the directory to be created.	Required

## Example

```
MKDIR Program_groups
```

## MOVE

---

**Syntax**      MOVE (File\_Name, Destination)

**Description**      This command is available for Macintosh only. The MOVE command will move a file to a new directory.

## Parameters

Parameter Name	Explanation	Required or Optional
file_name	Specifies the name of the file to be moved. Specify a full path or use symbolic substitution.	Required
destination	Specifies the directory the file is to be moved to. Specify a full path or use symbolic substitution.	Required

## Example

```
MOVE (edmaster.log &MACPATHS.DESKTOP)
```

## RELEASE

---

**Syntax** RELEASE

**Description** The RELEASE command enables mouse and keyboard input during execution. RELEASE is the default setting.

## Example

```
RELEASE
```

## REM

---

**Syntax** REM String  
or  
\* String

**Description** The REM command indicates a remark. All characters after REM or \* are ignored. The REM command is particularly useful for disabling commands.

## Parameters

Parameter Name	Explanation	Required or Optional
string	Can be any combination of characters.	Optional

## Example

```
*  
REM This batch program streamlines EDM.  
*
```

## RENAME

---

**Syntax** RENAME (OldFile\_Name, NewFile\_Name)

**Description** This command is available to Macintosh only. The RENAME command will change the name of any file. Symbolic notation may be used for either parameter.

## Parameters

Parameter Name	Explanation	Required or Optional
OldFile_Name	Specifies the file to be renamed.	Required
NewFile_Name	Specifies the new file name.	Required

## Example

```
RENAME ((&ZLIBDRV) (&ZLIBDIR) edmaster.log, &NEWLOGS.MASTERLOG)
```

## RENAME or REN

---

**Syntax**            RENAME File\_Name1 File\_Name2

**Description**      This command is available to Novell only. The RENAME command enables you to rename a file in a directory.

## Parameters

Parameter Name	Explanation	Required or Optional
File_Name1	Specifies the file to be renamed.	Required
File_Name2	Specifies the File_Name1's new file name.	Required

## Example

```
RENAME todo.txt tosee.txt
```

## RMDIR or RD

---

**Syntax**            RMDIR *Directory*

**Description**      This command is available to Novell only. The RMDIR command deletes a directory along with all its contents.

## Parameters

Parameter Name	Explanation	Required or Optional
Directory	Specifies the directory to be deleted.	Required

## Example

```
RMDIR EDMWIN
```

## SET\_ABORT\_TITLE

---

**Syntax**            `SET_ABORT_TITLE Text_String`

**Description**    The SET\_ABORT\_TITLE command allows you to set the text in the **Abort** button. In order to be effective, the SET\_ABORT\_TITLE call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_ABORT\_TITLE command is not initiated before the DISPLAY\_USER\_WINDOW call, the button is displayed with the default text, **Abort**.

While the program is running, it can delete the user window, change the button text, and redisplay the user window.

This command is not available in EDMCONS or EDMMNTR.

### Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed on the given button. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

### Example

```
SET_ABORT_TITLE STOP RUN
```

The above example changes the **Abort** button to read **Stop Run**.

## SET\_ABORT\_WINDOW

---

**Syntax**            `SET_ABORT_WINDOW (Left,Top,Right,Bottom)`

**Description**    The SET\_ABORT\_WINDOW command allows you to set the size and position of the **Abort** button. In order to be effective, the SET\_ABORT\_WINDOW call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_ABORT\_WINDOW command is not initiated before the DISPLAY\_USER\_WINDOW call, the **Abort** button is displayed in its default size and position.

While the program is running, it is possible to have it delete the user window, resize buttons, and redisplay the user window.

The **Abort** button cannot be displayed outside the user window.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the <b>Abort</b> button. The default is 90. The left border of the user window is considered position 0.	Required
Top	The pixel position for the top border of the <b>Abort</b> button. The default is 180. The top border of the user window is considered position 0.	Required
Right	The pixel position for the right border of the <b>Abort</b> button. The default is 180.	Required
Bottom	The pixel position for the bottom border of the <b>Abort</b> button. The default is 200.	Required

## Example

```
SET_ABORT_WINDOW (10,180,100,200)
```

The above example sets the size and position of the Abort window, which appears 10 pixels from the left of the user window, 180 pixels from the top, 100 pixels from the right, and 200 pixels from the bottom.

## SET\_CANCEL\_TITLE

---

### Syntax

```
SET_CANCEL_TITLE Text_String
```

### Description

The SET\_CANCEL\_TITLE command allows you to set the text in the **Cancel** button. The **Cancel** button resumes Extended Batch processing, and makes the **Cancel** and **OK** buttons invisible. In order to be effective, the SET\_CANCEL\_TITLE call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_CANCEL\_TITLE command is not initiated before the DISPLAY\_USER\_WINDOW call, the button is displayed with the default text, **Cancel**. The **Cancel** button is displayed only after the user chooses **Abort**.

While the program is running, it is possible to delete the user window, change the button text, and redisplay the user window.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed on the <b>Cancel</b> button. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

## Example

```
SET_CANCEL_TITLE Resume
```

The above example changes the **Cancel** button to read **Resume**.

## SET\_CANCEL\_WINDOW

---

**Syntax**            `SET_CANCEL_WINDOW (Left,Top,Right,Bottom)`

**Description**      The SET\_CANCEL\_WINDOW command allows you to set the size and position of the **Cancel** button. The **Cancel** button resumes Extended Batch processing, and makes the **Cancel** and **OK** buttons invisible. In order to be effective, the SET\_CANCEL\_WINDOW call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_CANCEL\_WINDOW command is not initiated before the DISPLAY\_USER\_WINDOW call, the **Cancel** button is displayed in its default size and position. The **Cancel** button is displayed only after the user chooses **Abort**.

While the program is running, it is possible to delete the user window, resize buttons, and redisplay the user window.

The **Cancel** button cannot be displayed outside the user window.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the <b>Cancel</b> button. The left border of the user window is considered position 0.	Required
Top	The pixel position for the top border of the <b>Cancel</b> button. The top border of the user window is considered position 0.	Required
Right	The pixel position for the right border of the <b>Cancel</b> button.	Required
Bottom	The pixel position for the bottom border of the <b>Cancel</b> button.	Required

## Example

```
SET_CANCEL_WINDOW (210,180,300,200)
```

The above example sets the default size and position values of the Cancel window, which appears 210 pixels from the left of the user window, 180 pixels from the top, 300 pixels from the right, and 200 pixels from the bottom.

## SET\_CONTINUE\_TITLE

---

**Syntax**            `SET_CONTINUE_TITLE Text_String`

**Description**    The SET\_CONTINUE\_TITLE command allows you to set the text in the **OK** button. Choosing the **OK** button will cause the Extended Batch program to terminate. In order to be effective, the SET\_CONTINUE\_TITLE call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_CONTINUE\_TITLE command is not initiated before the DISPLAY\_USER\_WINDOW call, the button is displayed with the default text, **OK**. The **OK** button is displayed only after the user chooses **Abort**.

While the program is running, it is possible to delete the user window, change button text, and redisplay the user window.

This command is not available in EDMCONS or EDMMNTR.

### Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed on the <b>OK</b> button. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

### Example

```
SET_CONTINUE_TITLE Exit
```

The above example changes the **Continue** button to read **Exit**.

## SET\_CONTINUE\_WINDOW

---

**Syntax**            `SET_CONTINUE_WINDOW (Left,Top,Right,Bottom)`

**Description**    The SET\_CONTINUE\_WINDOW command allows you to set the size and position of the **OK** button. Choosing the **OK** button will cause the Extended Batch program to abort. To be effective, the SET\_CONTINUE\_WINDOW call must precede the DISPLAY\_USER\_WINDOW command. If the SET\_CONTINUE\_WINDOW command is not initiated before the DISPLAY\_USER\_WINDOW call, the **OK** button is displayed in its default size and position. The **OK** button is displayed only after the user chooses **Abort**.

While the program is running, it is possible to delete the user window, resize buttons and redisplay the user window.

The **Continue** button cannot be displayed outside the borders of the user window.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the <b>OK</b> button. The <i>Left</i> border of the user window is considered position 0.	Required
Top	The pixel position for the top border of the <b>OK</b> button. The <i>Top</i> border of the user window is considered position 0.	Required
Right	The pixel position for the right border of the <b>OK</b> button	Required
Bottom	The pixel position for the bottom border of the <b>OK</b> button.	Required

### Example

```
SET_CONTINUE_WINDOW (310,180,400,200)
```

The example sets the default size and position of the Continue window. It appears 310 pixels from the left of the user window, 180 pixels from the top, 400 pixels from the right, and 200 pixels from the bottom.

## SET\_DECRYPT\_ON

---

**Syntax**            SET\_DECRYPT\_ON

**Description**     The SET\_DECRYPT\_ON command prevents confidential variables, messages, and information from being written to the log file. Once this command is issued, all information written to the log file is written as asterisks.

### Example

```
SET_DECRYPT_ON
```

## SET\_DECRYPT\_OFF

---

**Syntax**            SET\_DECRYPT\_OFF

**Description**     The SET\_DECRYPT\_OFF command resumes normal writing to the log file.

### Example

```
SET_DECRYPT_OFF
```

## SET\_ECHO\_TITLE

---

**Syntax**            SET\_ECHO\_TITLE *Text\_String*



**Description** If the SET\_ECHO\_TITLE command is not initiated before the ECHO text function call, the echo window is displayed with its default title bar, "Batch File Echo Display".

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed on the echo window title bar. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

## Example

```
SET_ECHO_TITLE ATTENTION AN ERROR HAS OCCURRED!
```

## SET\_ECHO\_WINDOW

---

**Syntax** SET\_ECHO\_WINDOW (*Left,Top,Right,Bottom*)

**Description** The SET\_ECHO\_WINDOW command sets the size and position of the echo window. An echo window is provided to display (and attract particular attention to) text or a message in a window separate from the user window during Extended Batch execution. If the SET\_ECHO\_WINDOW command is not initiated before an ECHO text function call, the echo window will be displayed using default size and position parameters. Once set and made visible, an echo window cannot be deleted or resized.

The echo window cannot be displayed outside the borders of the user window.

This command is not available in EDMCONS or EDMMNTR.

## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the echo window. The default is 10. The left border of the user window is considered position 0.	Required
Top	The pixel position for the top border of the echo window. The default is 25. The top border of the user window is considered position 0.	Required
Right	The pixel position for the right border of the echo window. The default is 400.	Required
Bottom	The pixel position for the bottom border of the echo window. The default is 80.	Required

## Example

```
SET_ECHO_WINDOW (10,25,400,65)
```

The above example sets the size and position of the echo window to be displayed for a subsequent ECHO text command. This echo window is displayed within the visible user window, which appears 10 pixels in from the left border of the user window and 25 pixels down from the top.

## SET\_FRONT\_WINDOW

---

**Syntax**            SET\_FRONT\_WINDOW

**Description**      The SET\_FRONT\_WINDOW command causes the window in which the EDM Extended Batch program is being executed to be the front most window.

This command is not available in EDMCONS or EDMMNTR.

## Example

```
SET_FRONT_WINDOW
```

## SET\_IND\_TITLE

---

**Syntax**            SET\_IND\_TITLE *Text\_String*

**Description**      The SET\_IND\_TITLE command allows you to set the text in the completion indicator title bar. In order to be effective, the SET\_IND\_TITLE command must precede the STARTIND command. If the completion indicator is displayed prior to the setting of the title bar text, the default title bar text is displayed, "Batch Completion Indicator."

## Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed in the completion indicator title bar. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed in the title bar.	Required

## Example

```
SET_IND_TITLE Percent Complete
```

## SET\_PROCESS\_NAME

---

**Syntax**            `SET_PROCESS_NAME (String)`

**Description**    This command is available for Windows 95, NT and OS/2 only. During program execution, the `SET_PROCESS_NAME` command prevents the execution of any other Extended Batch program on the same EDM Client that contains the process name *String*. All subsequent attempts to execute an Extended Batch program containing a process named *String* will be blocked, and no log files will be created to record the attempts. If you use this command, make `SET_PROCESS_NAME` the first line in the Extended Batch program.

### Parameters

Parameter Name	Explanation	Required or Optional
String	<i>String</i> can be up to 32 characters in length. The default value is CONNECT.	Optional

### Example

```
SET_PROCESS_NAME (connect)
```

The above example prohibits the execution of any Extended Batch file containing `SET_PROCESS_NAME (connect)` while the current Extended Batch program is processing.

## SET\_USER\_BACKGROUND

---

**Syntax**            `SET_USER_BACKGROUND Color`

**Description**    The `SET_USER_BACKGROUND` command allows you to set and alter the background color of the user window. The color can be changed during the execution of an Extended Batch procedure. Each successive `SET_USER_BACKGROUND` function call immediately changes the background color to the new specified color. The default window background color is white.

### Parameters

Parameter Name	Explanation	Required or Optional
Color	The background color for the user window. The available colors are white, red, green, blue, cyan, magenta, yellow, black, dkgray, gray, and ltgray.	Required

### Example

```
SET_USER_BACKGROUND RED
```

The above example changes the user window background to red.

## SET\_USER\_FOREGROUND

---

**Syntax**            `SET_USER_FOREGROUND Color`

**Description**     The SET\_USER\_FOREGROUND command allows you to set and alter the foreground (text) color of the user window. The color can be changed during the execution of an Extended Batch procedure. Each successive SET\_USER\_FOREGROUND function call immediately changes the foreground color to the new specified color. The default window foreground color is black.

### Parameters

Parameter Name	Explanation	Required or Optional
Color	The foreground (text) color for the user window. The available colors are white, red, green, blue, cyan, magenta, yellow, black, dkgray, gray, and ltgray.	Required

### Example

```
SET_USER_BACKGROUND BLUE
```

The above example changes the user window text to blue.

## SET\_USER\_TITLE

---

**Syntax**            `SET_USER_TITLE Text_String`

**Description**     The SET\_USER\_TITLE command allows you to set the text in the user window title bar. In order to be effective, the SET\_USER\_TITLE call must precede the DISPLAY\_USER\_WINDOW command. If the user window is displayed prior to the setting of the title bar text, the default title bar text is displayed, "EDM Extended Batch User Window."

During the course of the program, the user window can be deleted, the title reset, and the user window redisplayed with the new title.

### Parameters

Parameter Name	Explanation	Required or Optional
Text_String	The text to be displayed in the user window title bar. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed in the title bar.	Required

## Example

```
SET_USER_TITLE SIGNON PROCESS FOR &ZUSERID
```

In the above example, &ZUSERID is resolved before the title bar is set. If the value of &ZUSERID is USER001, the title bar is set to “SIGNON PROCESS FOR USER001.”

## SET\_USER\_WINDOW

---

**Syntax**            `SET_USER_WINDOW (Left,Top,Right,Bottom)`

**Description**     The SET\_USER\_WINDOW command allows you to set the size and position of the user window. If the SET\_USER\_WINDOW command is not initiated before the DISPLAY\_USER\_WINDOW call, the user window is displayed with its default size and position parameters. SET\_USER\_WINDOW has no effect if initiated after DISPLAY\_USER\_WINDOW.

The origin, position 0,0, for the user window is the upper-left corner of the screen. The horizontal size of the window is determined by subtracting the *Left* border setting from the *Right* border setting. Likewise, the vertical size of the user window is determined by subtracting the *Top* border setting from the *Bottom* border setting.

During the course of the program, the user window can be deleted, resized, and redisplayed.

### Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the user window. The default is 100.	Required
Top	The pixel position for the top border of the user window. The default is 100.	Required
Right	The pixel position for the right border of the user window. The default is 510.	Required
Bottom	The pixel position for the bottom border of the user window. The default is 310.	Required

## Example

```
SET_USER_WINDOW (50,50,550,300)
```

The above example sets the size and position for the user window with its upper-left corner 50 pixels from the left edge of the screen and 50 pixels from the top edge. The window is 500 pixels wide and 250 pixels high.

## SET\_VAR

---

**Syntax**            `SET_VAR (Object_Name,Heap_Number,Variable,Value)`

**Description** The SET\_VAR command sets the value of a variable within a specific heap of an object.

### Parameters

Parameter Name	Explanation	Required or Optional
Object_Name	A valid EDM object. <i>Object_Name</i> can be up to 8 characters in length..	Required
Heap_Number	Specifies the relative position of the heap or instance in the object. <i>Heap_Number</i> must be an integer between 0 and 65,536. The maximum value of <i>Heap_Number</i> is the size of <i>Object_Name</i> .	Required
Variable	A valid EDM variable. <i>Variable</i> can be up to 8 characters in length.	Required
Value	The data or value of the EDM variable. <i>Value</i> can be up to 255 characters in length.	Required

### Example

```
SET_VAR (ZMASTER, 0, ZBRC, 000)
```

## SETIND

---

**Syntax** SETIND *Percent*

**Description** The SETIND command allows you to set the value of the percentage that will be displayed in the batch completion indicator window. *PERCENT* incrementation should be provided frequently enough to indicate to the user that the procedure is running as expected. As there is no unequivocal method, in most cases, to determine the actual percent complete value, the indicator values should be a best guess or average percentage number.

In order to be effective, the STARTIND call must precede the SETIND command.

### Parameters

Parameter Name	Explanation	Required or Optional
Percent	The percentage of the Extended Batch procedure completed. The value of <i>PERCENT</i> must be an integer ranging from 0 to 100.	Required

### Examples

```
SETIND 25
SETIND 50
SETIND 99
```

## SINGLE\_STEP\_ON

---

**Syntax**            `SINGLE_STEP_ON`

**Description**    The `SINGLE_STEP_ON` command displays each command line before execution. In a GUI environment Extended Batch waits for the user to choose **OK**.

### Example

`SINGLE_STEP_ON`

## SINGLE\_STEP\_OFF

---

**Syntax**            `SINGLE_STEP_OFF`

**Description**    The `SINGLE_STEP_OFF` command stops the display of each command line before execution.

### Example

`SINGLE_STEP_OFF`

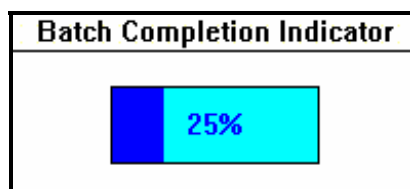
## STARTIND

---

**Syntax**            `STARTIND [ (Left,Top,Right,Bottom) ]`

**Description**    The `STARTIND` command sets the default size and position, and opens the Batch completion indicator. The indicator is initially set to 0%. Subsequent `SETIND` commands must be used to change the completion ratio.

Use the Batch completion indicator to give users continuous feedback during a long Extended Batch process, in addition to the text information you provide in the user and echo windows:



This command can be used with either all four parameters or with no parameters at all. If parameters are used, all four must be entered. If the `STARTIND` command is initiated without setting parameters, the indicator window is displayed in the default size and position. The Batch completion indicator cannot be displayed outside the borders of the user window.

This command is not available in `EDMCONS` or `EDMMNTR`.

## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the left border of the indicator. The default is 100. The left border of the user window is considered position 0.	Optional
Top	The pixel position for the top border of the indicator. The default is 50. The top border of the user window is considered position 0.	Optional
Right	The pixel position for the right border of the indicator. The default is 400.	Optional
Bottom	The pixel position for the bottom border of the indicator. The default is 150.	Optional

## Example

```
STARTIND (100,45,400,145)
```

The above example sets the default size and position values for the Batch completion indicator. The left of the indicator is 100 pixels from the left of the user window, the top is 45 pixels from the top of the user window, the right is 400, and the bottom is 145.

## TRAP

---

**Syntax**            TRAP

**Description**      The TRAP command disables any mouse or keyboard input during execution.

## Example

```
TRAP
```

## WRITE\_USER\_WINDOW

---

**Syntax**            WRITE\_USER\_WINDOW (*Left,Top,Text\_String*)

**Description**      The WRITE\_USER\_WINDOW command allows you to display text in the user window. Text can be changed during the execution of an Extended Batch procedure. Each successive WRITE\_USER\_WINDOW function call clears any existing text in the user window before adding the new text.

This command is not available in EDMCONS or EDMMNTR.



## Parameters

Parameter Name	Explanation	Required or Optional
Left	The pixel position for the beginning of the text string. The <i>Left</i> border of the user window is considered position 0.	Required
Top	The pixel position for the top of the text string. The <i>Top</i> border of the user window is considered position 0.	Required
Text_String	The text to be displayed in the user window. Do not place quotes around literals. Object variables included in the text string are resolved before being displayed.	Required

## Example

```
WRITE_USER_WINDOW (25,50,SIGNON PROCESS IN PROGRESS)
```

## Using Installed EDM Extended Batch Programs

---

When you install the EDM Client and the EDM Administrator on your computer, a number of EDM Extended Batch files are automatically installed, some on the EDM Administrator and some on the EDM Client. These Extended Batch files execute various EDM processes and utilities such as the EDM Client Connect Process, **and the EDM System Explorer.**

**Warning:** Before executing any EDM Extended Batch files, consult the following table, "*Installed EDM Extended Batch Files,*" for a description of each file. We strongly recommend not executing an .EXT file unless you know the intended result.

### Installed EDM Extended Batch Files

Extended Batch File	Description	Available on EDM Client	Available on EDM Administrator
ADMIN.EXT	A residual file from the EDM Administrator installation process. (This file should not be executed.)		
BROWSER.EXT	<b><u>Runs the EDM System Explorer</u></b> , which also can be run by clicking the EDM System Explorer in the EDM Administrator folder.		
CONNECT.EXT	Runs the EDM Client Connect Process, which also can be run by clicking the EDM Client Connect icon in the EDM Desktop program group.		
HWINFOW.EXT	Creates the ZCONFIG object that is used by the EDM Client Connect Process.		



# The EDM Dialog Command Reference

This chapter provides you with the information you need to understand and use the EDM Dialog scripting commands for LU2 access, including guidelines on syntax, usage, and variable definition.

For your convenience, a quick reference table is provided with brief definitions of the dialog commands. Each dialog definition includes usage examples, and a platform checklist to let you know if the command you need will be available on your operating system.

## Platform Availability

---

The EDM Dialog scripting commands are available on the following platforms:

Platform Availability				
DOS	<input checked="" type="checkbox"/>		NCR MP/RAS	<input type="checkbox"/>
Windows	<input checked="" type="checkbox"/>		Sun OS	<input type="checkbox"/>
Windows NT	<input checked="" type="checkbox"/>		Unix AIX	<input type="checkbox"/>
Windows 95	<input checked="" type="checkbox"/>		Unix HP-UX	<input type="checkbox"/>
OS/2	<input checked="" type="checkbox"/>		Unix Solaris	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>		UnixWare	<input type="checkbox"/>
NetWare	<input type="checkbox"/>			

EDM Dialog commands work the same on the supported platforms checked in the above table, except where noted in this chapter.

## EDM Dialog Command Facility at a Glance

---

EDM Dialog Manager is a high-level language application programming interface (HLLAPI) scripting facility that provides a simple means of automating standard key sequences, such as sign-on procedures, logon, and logoff. Dialogs are used for LU2 access only.

A dialog command file (.DIA) is a standard ASCII text file that contains EDM dialog scripting commands.

EDM dialogs are used for communicating with an existing host-based application, sending commands or key sequences and waiting for the appropriate host responses, and prompting the user for data such as password or logon ID. Scripts developed with the dialog commands can be used to accomplish repetitive tasks from the desktop.

The EDM dialog commands are visible to the local EDM objects on the desktop and can use object variables in place of hard coding values in scripts. For example, a script could be developed to sign a user onto TSO, and pass the userid and password from the ZMASTER object. The password entered during an EDM connection is encrypted on the desktop by ZMASTER.

## Understanding Dialog Syntax

---

You must follow certain conventions when creating dialog files. This section provides you with the details for creating dialog files and specifying dialog commands and variables.

There is no limit to the number of command lines you can create as long as the dialog file does not exceed 64K in length. The maximum length for a single line is 255 bytes.

- Any line beginning with an asterisk (\*) or a semicolon (;) is treated as a comment, and has no effect on the dialog.
- Commas cause a one-second delay. Do not use periods.
- The command can start in any column, provided that only blanks precede it. Blank lines are ignored. Command lines are translated to uppercase.
- Label lines identify positions in the dialog that can be referenced by the IF, ON, GOTO, and GOSUB commands. The label must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (\*) can follow the label on the label line. Commands can not be placed on a label line.

In the following examples, the first three lines are valid labels, and the last two lines are invalid labels:

### Examples

```
* valid labels:
:Time-is-up
:L1
:Do_PCMF
* invalid labels:
:This-is-too-long
:Has space
```

- A maximum of 100 labels can be used in any one dialog.
- Each command must be entirely contained within a single line. No continuation lines are accepted and only one command per line is permitted.

Some commands can be abbreviated. The minimum abbreviation of each command is underlined in the syntax statement of each page.

## Dialog Variables

---

If a command line contains a word that begins with a valid variable prefix, then that word is a variable. Variables are replaced with user responses or other data, depending on the variable type. Dialog supports the following four variable types.

- Prompted variables (variable prefixes: & or \$).
- Common variables (variable prefix: #).
- Environment variables (variable prefix: %).

Predefined variables (variable prefix: @).

The command `SEND 'AT TD' &PHONE-NUMBER ENTER` prompts you for a phone number prior to the execution of the dialog.

If a variable prefix occurs within a literal, then it is treated as any other character and not as a variable identifier. The `SEND '$DA' ENTER` command would send the string \$DA, because it is enclosed within single quotes.

If you want to use any variable types other than a prompted variable, you must include the following dialog command:

```
OPTION LANGLEVEL (2)
```

## Prompted Variables

Prompted variables are identified by a preceding & or \$. You are prompted to respond with the values for all prompted variable names prior to the actual dialog execution. (Prompted variables are resolved before execution.)

The variable name is the character string that immediately follows the & or \$ symbol; it is terminated by a blank, comma, parenthesis, or single quote. This variable name is used immediately after “Please enter” in the prompt. The name should be meaningful, but cannot exceed 32 characters, and must be free of any embedded termination characters. The prompt displays the variable name in uppercase.

Type the response and press ENTER. If you press ENTER without entering a response, the variable is null and does not appear in the command.

The &variable name is not secured, appearing on the screen as you enter it. The \$variable name is secured; as you enter the value for a \$variable, the input does not display on the screen. This is useful for inputting passwords and other sensitive types of information.

The same variable name can be used in several places within a DIALOG script. You are prompted only once for each unique variable name, and your response is substituted for each occurrence of the variable name within the script.

## Common Variables (DOS Only)

Common variables are used when the sharing of information across the entire EDM system is required. In DOS this is equivalent to environment variables. Common variables are allocated when they are the target of an assignment command (for example, SET, CALC, INPUT).

### Example

```
INPUT 'ENTER YOUR NAME: ' NAME
INPUT 'ENTER YOUR PASSWORD: ' PASSWORD HIDDEN
SEND HOME #NAME TAB #PASSWORD ENTER
```

**Note:** In assignment commands, the receiving variable must omit the variable prefix that it ordinarily uses.

## Environment Variables (DOS only)

Environment variables are replaced by the text associated with the corresponding DOS environment variable, which is established by the DOS SET command. If the DOS environment contains no matching variable, a null string replaces the variable.

When using DIALOG to dial a modem, a common requirement is to dial 9 first to obtain an outside line. If the DOS command SET DIALPREFIX=9 were in the executing machine's AUTOEXEC.BAT file, then the dialog command SEND 'AT TD' %DIALPREFIX &PHONE-NUMBER ENTER could be used.

The same dialog could then be used on machines with direct phone lines, since those machines would omit the SET DIALPREFIX command from their AUTOEXEC.BAT files.

## Predefined Variables (DOS Only)

Predefined variables begin with the @ symbol and are classified as simple or complex. Simple predefined variables allow you to access the date, time and various other items. Complex predefined variables are actually functions that require sub-parameters; however, they are used the same way as simple predefined variables.

In the following example, the first line is a simple predefined variable, and the second line is a complex predefined variable:

### Example

```
REM THE CURRENT DATE IS @DATE
REM THE MONTH OF THE YEAR IS @SUBSTR(@DATE,1,2)
```

For a complete description of all predefined variables, see “*Predefined Variables*” in this chapter.

## Variable Usage

Variable types are generally interchangeable, and variables can be used anywhere. There are three special rules concerning the use of variables:

When using the SET, CALC and INPUT commands, the target variable can only be a prompted variable or a common variable. If the target variable already exists, it is replaced and retains its type. If it does not exist, a common variable is created.

Within expressions, the contents of all referenced variables must be numeric or valid expressions.

Complex predefined variables are not to be used as parameters in other complex predefined variables.

In the following examples, the first example contains a valid usage, and the second example contains an invalid usage:

### Examples

```
* Valid usage:
@SCREEN(@ROW,@COL,10)
* Invalid Usage:
@SCREEN(@SUBSTR(#FIELD,1,3),@COL,10)
```

## Dialog Command Quick Reference

The following table contains a summary of the EDM Dialog commands. More detailed descriptions of the commands are provided in the pages following the quick reference table.

### EDM Dialog Commands

Command Name	Short Description
BREAK	Enables interrupts in the dialog with the CTRL+BREAK keys.
CALC	Evaluates the expression, and assigns results to a variable.
CLOCK	Halts the execution of a dialog for a specified interval, or until a specified time and day.
DELAY	Delays the dialog up to 255 seconds where it is used in the dialog file.
DEVICE	Specifies the communications device driver to be used. (See list of device drivers in this section.)
DIALOG	Sets the error level testable by a DOS .BAT procedure, or presents an error message on the console display.
DLGBOX	Allows for definition and execution of a Windows or OS/2 dialog box.
ELSE	Specifies an alternate command sequence when a preceding IF THEN command evaluates false.
END	Causes the dialog to terminate with a DOS error level of 0.
ENDIF	Terminates the preceding IF THEN command.
EXECUTE	Executes the specified program or batch file.
EXECUTE(P)	Instructs DIALOG to pass device and line parameters to the program being executed.
FLICKER	Sets the flicker check time to a specified number of seconds.
GOSUB	Causes an immediate branch to the specified position identified by a label in the dialog file.
GOTO	Causes the dialog to execute from the given label.
HOST	Defines the host session short name for multiple host session communication facilities.
IF	Allows different actions to be taken depending on what is found on the screen. There are three forms of the IF command: IF...THEN, IF...GOTO, and IF...QUIT.
INPUT	Prompts the user for a value that is to be assigned to a variable.
KEYDELAY	Sets a constant delay between keystrokes.
LINE	Permits the communications parameters to be specified within a dialog, and overrides any parameters in the configuration file. (Keywords used with this command are listed in this section.)
MANUAL	Initiates full-screen interactive manual keyboard, or terminal emulation mode.
NOBREAK	Prevents CTRL+BREAK key combination from interrupting the dialog.
OFF	Turns off a previously issued ON command.
OIA	Allows the DIALOG to define up to a 33-byte message that appears in the operator information area.
ON BREAK GOTO	The CTRL+BREAK key causes the control flow of the dialog to move to the command immediately following the label.
ON EVENTN	Allows a specific action to be taken whenever the presence of text is found on the screen.
ON GOTO	On presence of a given condition, causes the control flow of the dialog to move to the command immediately following the label.
ON PROGN GOTO	Allows specific exits from the manual keyboard mode whenever you press certain keys.
ON RUNOUT GOTO	Indicates where execution should continue if too many GOTOs occur within an IF or GOTO command.
ON TIMEOUT GOTO	Used in conjunction with any WAITFOR or SEND command.
OPTION	Establishes characteristics for dialog execution with regard to language level.
PAUSE	Causes the rest of the command line after PAUSE to be displayed followed by "Press Any Key" prompt.
PRINT	Sends screen images or literals to either the printer or a disk file.
PROMPT	Allows the DIALOG to prompt the operator for a one-character response.
QUIT	Terminates the dialog.

Command Name	Short Description
REMARK	Causes the rest of the command line after the REMARK command to be displayed.
RETURN	Returns execution of the statement after the last active GOSUB.
RUN OR RUN(P)	Allows the DIALOG to execute a program so that the result code from that program can be interrogated by the IF ERRORLEVEL command.
RUNOUT	Limits the number of GOTO or GOSUB jumps that can be executed by a dialog.
SCREEN	Sends the current screen to either the print file or the printer.
SEND	Sends keystrokes to the communications device.
SET	Concatenates the values of the specified items and assigns the results to a variable.
TIMEOUT	The time in seconds of a WAITFOR condition.
WAITFOR	Causes the dialog to wait until the specified text appears on the screen.

## EDM Dialog Command Reference

---

This section provides you with details for specifying the EDM dialog commands. Each command is documented, including all the information required to use it, as follows:

- Command name.
- Command description.
- Platform availability.
- Syntax.

Syntax or usage examples.

## BREAK

---

**Syntax**            BREAK

**Description**        The BREAK command allows the user to interrupt the dialog with the CTRL+BREAK key. This can be useful when a dialog pauses too long for a condition. BREAK is the default. This command can be repeated in the dialog, and it applies to all commands below it until a NOBREAK is encountered or the end of the dialog is reached.

**Availability**        DOS, WIN, NT, WIN95, OS/2

### Example

```
BREAK
```

## CALC

---

**Syntax**            CALC Variable = Expression

**Description**        The CALC command evaluates *Expression*, and assigns the resulting numeric value to *Variable*.



**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
Variable	A valid Dialog Common or Prompted variable. If the variable existed previously, the original contents are lost. The variable prefix (for example: #) must be omitted from the target <i>Variable</i> name.
Expression	A valid Dialog expression. The equal sign must be preceded and followed by at least one space. For complete details on expression syntax, see " <i>Numeric Expressions</i> " on page 239.

## Example

```
CALC SQUARE = #NBR * #NBR
```

# CLOCK

---

**Syntax** CLOCK(+HH:MM)  
or  
CLOCK(hh:mm [am/pm] [Day])

**Description** The CLOCK command halts the execution of a dialog until a specific time of day and day of the week is found, or until a specified time interval has elapsed.

When a dialog is in a CLOCK wait, it displays the time at which it will proceed and the current time. It also displays a prompt telling the user to press ESC to skip the wait and proceed, or to press CTRL+BREAK to quit the dialog.

The command can be repeated in the dialog and causes the specified wait whenever it occurs. DIALOG uses the clock and date in the desktop, not the EDM Manager nor the EDM Stager, to determine the time of day and day of week. It will start at the wrong time if the PC clock is not correct. Use the DOS TIME command to determine what time is set on the PC clock and to correct it if necessary.

**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
+HH:MM	The specified time interval, in hours and minutes, EDM Dialog Manager will wait before resuming execution of the dialog. The time specified is absolute and will not be affected by an incorrectly set PC clock.
hh:mm	The time at which the dialog will resume execution. Both 12 and 24 hour formats are accepted. Make sure the time of day and date is set correctly when using the CLOCK(hh:mm Day) format.
am/pm	Specify this option if you are using 12 hh:mm format.

Parameter Name	Explanation
Day	Valid <i>Day</i> designations are as follows: SUN, MON, TUE, WED, THU, FRI, SAT. If day is not specified, same day is assumed.

### Example 1

CLOCK (7:30pm)  
CLOCK (19:30)

In the above example, the dialog waits until 7:30 at night before proceeding. Note that either 12 or 24 hour format is acceptable.

### Example 2

CLOCK (7:30pm SAT)

In the above example, the dialog waits until Saturday night at 7:30 before proceeding. Note that if the dialog is started on Saturday, before the CLOCK time, the dialog runs that day at the time specified. If the dialog is started after the CLOCK time, the dialog waits until the next Saturday at 7:30pm.

### Example 3

CLOCK (+0:30)

In this example, the dialog waits exactly 30 minutes from the current time before proceeding.

## DELAY

---

**Syntax**            DELAY(*sss.nn*)

**Description**      The DELAY command delays the dialog for a specified amount of seconds, which can be useful in allowing for a full response from the host. DELAY commands can occur anywhere in the dialog, and are applied immediately to their location in the dialog.

A comma placed in a dialog causes a one-second pause when it is encountered.

**Availability**      DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
sss.nn	sss specifies seconds and <i>nn</i> specifies hundredths of seconds. The maximum delay is 255 seconds.

### Example

DELAY (123.45)

## DEVICE

---

**Syntax**            DEVICE(*Device\_Name*)

**Description**      The DEVICE command specifies the communications device driver to be used for a particular DIALOG execution. The DEVICE command can occur, only once, anywhere in the dialog.

The DEVICE command is ignored when the STARTUP.PROC option has been specified on the DOS command line.

The underlined characters in the parameters table represent the acceptable abbreviations.

**Availability**      DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
<u>3270</u> PC	IBM 3270PC/AT and Workstation.
<u>52A</u>	AST 5250 Emulator.
<u>5250</u>	IBM 5250 Emulator.
<u>AS</u> 400	IBM's PC-Support AS/400.
<u>AST</u>	AST 3274 remote emulator (BSC or SNA).
<u>BANYAN</u>	Banyan Vines gateway.
<u>CMG</u>	OS/2 Communications Manager.
DFT	cfSOFTWARE's resident DFT processor.
EIC	EICON.
<u>ENTRY</u> or <u>HLLAPI</u>	IBM PC 3270 Emulation Version 1 (HLLAPI) IRMA+.
<u>EXTW</u>	Attachmate Extra! for Windows.
FORTE	Forte 3278 emulator.
IBM	IBM 3278 emulation card (CUT mode).
IBW	Personal Communications Version 2 and above.
IDEA	IDEA 3278 emulator.
IRMA	IRMA/IRMA2 3278 emulator.
IRW	IRMA Workstation for Windows.
ITT	ITT Courier 3278 emulator.
LEVEL-3 or API	IBM PC 3270 Emulation Version 3.0 (API).
MICROPLUS	Micro Plus 3278 emulator.
PATHWAY	ICOT/Pathway 3274 remote emulator (BSC or SNA).
PCOX	PCOX/CXI any card.
PROTOCOL	Any protocol converter with IBM 3101 support.
RUM	RUMBA.
TYMNET78	Any TYMNET hook-up using the TYMNET78 protocol.
VT100	Any protocol converter with DEC VT100 support.
XIRCOM	Xircom Pocket 3270 Adapter.

## Example

```
DEV (&ZDEVTP)
```

## DLGBOX

---

**Syntax**

```
DLGBOX NEW
DLGBOX LINE $n$ ="String"
DLGBOX BUTTON $n$ ="Text"
DLGBOX EXECUTE
```

**Description** The DLGBOX command allows for the definition and execution of a Windows or OS/2 dialog box. A dialog box can have up to six lines of text and up to six buttons.

After the dialog box is executed, the result is returned in the same way a PROMPT command returns a value. A one through six is returned for buttons 1 through 6, and an X is returned if you pressed ESC.

**Availability** WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
NEW	Clears all lines and buttons.
LINE $n$ ="String"	Defines a line of text in the dialog box. $n$ specifies line number and is an integer between 1 and 6. The text in <i>String</i> may be up to 40 characters long and may be a quoted literal or a variable. A line can be cleared by setting <i>String</i> equal to "" .
BUTTON $n$ ="Text"	Defines the button text. $n$ specifies the button number and is an integer between 1 and 6. <i>Text</i> may be up to 12 characters long. A button can be cleared by setting <i>Text</i> equal to "" .
EXECUTE	Displays the dialog box.

## Example

```
DLGBOX NEW
DLGBOX LINE1="AN ERROR HAS OCCURRED."
DLGBOX LINE2="DO YOU WISH TO CONTINUE?"
DLGBOX BUTTON1="CONTINUE"
DLGBOX BUTTON2="CANCEL"
DLGBOX EXECUTE
IF {1} GOTO CONTINUE
IF {2} GOTO CANCEL
IF {X} GOTO ESCAPE
```

## ELSE

---

**Syntax**            *ELSE Command*

**Description**      The ELSE command specifies an alternate command sequence to be used if the preceding IF...THEN command evaluates false.

**Availability**      DOS

### Parameters

Parameter Name	Explanation
Command	Any valid Dialog command.

## Example

```
IF VAR #ANSWER EQ 'Y' THEN
    REM YOUR ANSWER IS YES
ELSE
    REM YOUR ANSWER IS NO
ENDIF
```

## END

---

**Syntax**            END

**Description**      The END command causes the dialog to terminate with a DOS ERRORLEVEL of 0 (OK). The END command can occur multiple times and is accepted when encountered within the dialog. END is not required as the last command. If the dialog reaches the end of the dialog command file, then END is assumed.

**Availability**      DOS, WIN, NT, WIN95, OS/2

## Example

END

## ENDIF

---

**Syntax**            ENDIF

**Description**      The ENDIF command terminates the preceding IF THEN command.

**Availability**     DOS

## Example

```
IF VAR #ANSWER EQ 'Y' THEN
    REM YOUR ANSWER IS YES
ELSE
    REM YOUR ANSWER IS NO
ENDIF
```

## EXECUTE EXECUTE(p)

---

**Syntax**            EXECUTE 'Program [*Parameters*]'  
or  
EXECUTE(P) 'Program [*Parameters*]'

**Description**      The EXECUTE command executes a specified program, batch file, or DOS command, passing it the included parameters as if they were specified on the DOS command line. This command invokes a new copy of DOS and passes to this DOS the command given within the single quotes. This allows you to pass any string of characters that is valid at the DOS command prompt. Any number of EXEC commands may be used in a dialog.

The EXECUTE(P) format instructs EDM Dialog Manager to pass DEVICE and LINE parameters to the program being executed. If this format is used to invoke MAIN, the following DIALOG parameters are passed and take precedence over any corresponding parameter defined in the setup file:

DEVICE.

HOST.

All LINE parameters (Commport, Speed, Parity, Data Bits, Stop Bits).

**Availability**     DOS

## Parameters

Parameter Name	Explanation
Program [Parameters]	Any valid DOS command, .BAT file or program, and its parameters. <i>Program</i> and any <i>Parameters</i> must be enclosed within single quotes and appear as if typed at the command prompt. <i>Parameters</i> , if present, must be separated from <i>Program</i> by at least one space and can be entered as either an &variable or a \$variable. See SEND for a description of variables. If a parameter is entered as a variable, it must be outside the single quotes.

## FLICKER

---

**Syntax**            FLICKER(*ss.nn*)

**Description**        The FLICKER command sets the flicker check time to the specified amount of seconds. Flicker check time is the extra time taken to assure that a full response has been received from the host, rather than a flickering partial response. This command can be repeated in the dialog and applies to all subsequent SEND and WAITFOR commands until another FLICKER command or the end of the dialog is reached.

**Availability**        DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
ss.nn	ss specifies seconds and nn specifies hundredths of a second. The default flicker time is 00.50 seconds.

### Example

FLICK (1)

## GOSUB

---

**Syntax**            GOSUB *Label*

**Description**        The GOSUB command causes an immediate branch to the specified Label. Control is returned to the command immediately following the GOSUB when the first RETURN command is encountered. GOSUBs can be nested in this fashion up to sixteen levels.

**Availability**        DOS

## Parameters

Parameter Name	Explanation
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

## Example

```
GOSUB CALC-DIFF
REM DIFFERENCE IS #DIFFERENCE
END
.
.
.
:CALC-DIFF
CALC DIFFERENCE = #HIGH - #LOW
RETURN
```

## GOTO

---

**Syntax**            *GOTO Label*

**Description**      The GOTO command causes the dialog to execute from the given label. There is no limit on the number of GOTO commands that can be used in a dialog.

## Parameters

Parameter Name	Explanation
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

## Example

```
GOTO RETRY_DIAL
```

The above example causes the dialog to execute the next sequential command following the :RETRY\_DIAL label line.



# HOST

---

**Syntax** HOST(*X*)

**Description** The HOST command defines the host session short name for multiple host session communication facilities, such as DFT. Usually, the host session short name is a single letter. The device drivers LEVEL-3 (API), ENTRY (HLLAPI), DFT and PCOX use this parameter.

If a HOST session is specified in a dialog, it overrides any host session name configured by the program MAINCON.

**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
X	The host session short name. X can be a prompted variable.

## Example

HOST (&ZHOST)

# IF

---

**Syntax** IF *Condition* THEN  
                   Commands\_Executed\_If\_True  
 [ELSE  
                   Commands\_Executed\_If\_False]  
 ENDIF

or  
 IF *Condition* GOTO *Label*  
 or  
 IF *Condition* QUIT [(*Return\_Code*) '*Message*']

**Availability** DOS, WIN, NT, WIN95, OS/2

**Description** The IF command tests whether *Condition* is true or false, and executes the command (or commands) that follows provided that *Condition* was found to be true. In DOS only, should *Condition* evaluate false, an alternate command sequence can be specified with ELSE.

Note, the IF parameters are divided into two tables, DOS only parameters, and Dos, Windows, Windows NT, Windows 95, and OS/2 paramters.

**Parameters** (DOS Only)

Parameter Name	Explanation
Condition	<i>Condition</i> is tested by the IF command. Test evaluation is either true or false. Should the test be true, IF will execute <code>Commands_Executed_If_True</code> . <i>Condition</i> may be one of the following: {x}, ERRORLEVEL, NUM, EXIST, Text, or VARIABLE. Descriptions for EXIST, Text, and VARIABLE are provided on the next page.
{x}	Tests the user's response to a PROMPT command and is not case sensitive. If the key the user entered is the one specified by x, the test is true. The response, x, can be any single character.
ERRORLEVEL Comparison Result_Value	Tests the result code issued by a program executed with the RUN command. The DOS ERRORLEVEL is compared with the coded value using the specified relational operator. The IF ERRORLEVEL command can only be issued in a DIALOG after a RUN command.  DOS allows a result code to be returned by a program. This result code is testable in a DOS .BAT file via the DOS IF ERRORLEVEL command. Any value testable under the DOS IF ERRORLEVEL command can be tested by the DIALOG IF ERRORLEVEL. Only programs that return a result code produce valid codes to be tested by DIALOG's IF ERRORLEVEL command. Comparison can be one of the following two letter operators: EQ (equal to). NE (not equal). LT (less than). LE (less than or equal to). GT (greater than). GE (greater than or equal to). Result_Value can be any unsigned number from 0 to 255.
NUM Expression	Evaluates a numeric Expression. For a complete discussion on expression syntax see "Numeric Expressions" at the end of this chapter.
Commands_Executed_If_True	Commands can consist of any number of valid DIALOG commands, including more IF...THEN sequences.
ELSE	This command specifies an alternate command sequence should the preceding IF THEN evaluate false.
Commands_Executed_If_False	Commands can consist of any number of valid DIALOG commands, including more IF...THEN sequences.
ENDIF	This command terminates the preceding IF THEN command. Every IF THEN must be paired with an ENDIF command.

## Parameters (DOS, Windows, Windows NT, Windows 95, and OS/2)

Parameter Name	Explanation
Condition	<i>Condition</i> is tested by the IF command. Test evaluation is either true or false. Should the test be true, IF will execute the GOTO or QUIT. <i>Condition</i> may be one of the following: EXIST, Text, or VARIABLE.
EXIST <i>Filename</i>	Test for the existence of a file. If the specified file is found in the target directory, the test condition is true and execution continues accordingly. Filename must include full path and extension. If the path is not specified, the current directory is assumed. Wildcards (* and ?) may be used. Only non-hidden files can be tested for with this command. If the file exists with a length of zero, this test evaluates as false.
[(rr,cc[,ll])] <i>Text</i>	Tests for the presence of the <i>Text</i> anywhere on the screen. If <i>Text</i> is found the test condition is true and execution continues accordingly. This is useful in cases where the current state of the 3270 session is unknown.  <i>Text</i> must be enclosed within single quotes. No uppercase translation is performed.  Multiple <i>Text</i> strings can be tested. Each <i>Text</i> string must be enclosed within single quotes and separated from other <i>Text</i> strings with the OR operator. If any of the <i>Text</i> strings are encountered, the test is true. Any number of <i>Text</i> strings can be tested in a single IF statement. Using (rr,cc[,ll]) limits the test for the presence of <i>Text</i> to the specified portion of the screen. String length // is optional. If // is not specified, the search commences, starting at row rr, and column cc, and continues to the end of the screen.

Parameter Name	Explanation
<code>VARIABLE &amp;Variable_Name EQ Text</code>	Tests if <code>&amp;Variable_Name</code> (or <code>\$Variable</code> ) is equal to <code>Text</code> . If the value supplied for the prompted variable is equal to <code>Text</code> , the test condition is true. The maximum length of <code>Text</code> or <code>&amp;Variable_Name</code> is 32 characters. If a longer value is supplied, the <code>Text</code> or <code>&amp;Variable_Name</code> is truncated to 32 characters. LT, GT, NE, or LE may be used instead of EQ. For descriptions see ERRORLEVEL parameter on previous page.
<code>GOTO Label</code>	Causes the dialog to execute the next command following the <code>Label</code> line. <code>Label</code> specifies a valid DIALOG label. Labels identify positions in the dialog, and must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.
<code>QUIT</code>	Terminates the dialog immediately if test condition is true. The IF QUIT is processed in the same way as the standard QUIT command. For more details see "QUIT" later in this chapter.

### Example 1

```
:MENU
REM 1. Choice one
REM 2. Choice two
REM 3. Choice three
REM 4. Choice four
PROMPT Please enter your choice:
IF {1} GOTO got1
IF {2} GOTO got2
IF {3} GOTO got3
IF {4} GOTO got4
GOTO PROMPT
```

In the above example, a menu of choices is displayed by the REM command and a response is obtained via the PROMPT command. The subsequent IF {x} commands test the response.

### Example 2

```
RUN 'MAIN' 'START.001'
IF ERRORLEVEL EQ 0 GOTO ALL-OK
IF ERRORLEVEL EQ 4 GOTO FATAL-ERR
IF ERRORLEVEL BREAK
```

### Example 3

```
IF NUM (#UPPER - #LOWER) > 10 THEN
REM DIFFERENCE EXCEEDS 10
ENDIF
```

### Example 4

```
IF EXIST c:\edwin\myfile.dia GOTO Connect
```

### Example 5

```
IF (25,10,4) 'help' OR '?' GOTO Help_Panel
```

### Example 6

```
IF VARIABLE &DEVICE EQ 'IRMA' GOTO USE-IRMA
```

## INPUT

---

**Syntax** INPUT [*'String'*] *Variable* [HIDDEN]

**Description** The INPUT command prompts the user for a value that is assigned to variable. For INPUT, the variable prefix (for example #) must be omitted from the variable name.

**Availability** DOS

### Parameters

Parameter Name	Explanation
String	Use a text string to prompt the user to input the variable value. <i>String</i> must be enclosed in single quotes.
Variable	A valid Dialog Variable.
HIDDEN	The optional HIDDEN keyword is used for secure fields; if included, HIDDEN prevents the user's response from being displayed while it is being entered.

### Example

```
INPUT 'PLEASE ENTER YOUR PASSWORD: ' PASSWORD HIDDEN
```

The above example prompts you for a password and stores the response in the variable #PASSWORD.

## KEYDELAY

---

**Syntax** KEYDELAY(*ss.nn*)

**Description** The KEYDELAY command sets a constant delay between keystrokes. A KEYDELAY may be needed if the communications device or control unit tends to jam when receiving keys at computer speed. KEYDELAY commands can occur anywhere in the dialog. They apply to all SEND commands until the next KEYDELAY command is encountered or the end of the dialog is reached.

**Availability** DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
ss.nn	ss specifies seconds and nn specifies hundredths of a second. The default KEYDELAY is 00.01 seconds.

## Example

```
KEYDELAY(1.03)
```

## LINE

---

**Syntax**            `LINE [COMMPORT(n)] [SPEED(sssss)] [PARITY(ppppp)] [DATABITS(d)] [STOPBITS(s)]`

**Description**        The LINE command is used only with the PRO and VT100 device drivers as protocol converters. It permits the communications parameters to be specified within a dialog, and overrides any parameters in the configuration file.

Use of the LINE command permits a single dialog to communicate with multiple protocol converters using different speeds, COMM ports or data configurations.

Whenever this command is encountered, dialog closes and reopens the communications driver with the new parameters. This activity resets the line parameters and clears the communication buffer. The LINE command should be used only before initiating a new dial-up and never during a connection.

The LINE command is followed by at least one, and as many as five keywords that specify the individual values for the line parameters. Any values that are not supplied in a LINE command receive the default values in the configuration file for protocol converters. The keywords may be entered using only the first two letters of the keyword as shown with the underline.

**Availability**        DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
<u>C</u> ommport( <i>n</i> )	This keyword specifies the COMM port on the PC to be used. <i>n</i> can be 1 or 2.
<u>S</u> peed( <i>sssss</i> )	This keyword specifies the speed of the line. Any of the following values can be used: 300, 1200, 2400, 4800, 9600, 19.2KB, 38KB, 56KB, 115KB.
<u>P</u> arity( <i>pppp</i> )	This keyword specifies the parity to be used. Any of the following values are acceptable. Only the first letter can be used as an abbreviation: <u>E</u> ven, <u>O</u> dd, <u>N</u> one, <u>M</u> ark, <u>S</u> pace.
<u>D</u> atabits( <i>d</i> )	This keyword specifies the number of bits used to represent data in each transmission character. <i>d</i> can be 7 or 8.
<u>S</u> topbits( <i>s</i> )	This keyword specifies the number of bits used to indicate the end of each transmission character. <i>s</i> can be 1 or 2.

## MANUAL

---

**Syntax**            `MANUAL [NOPROMPT]`

**Description**        The MANUAL command initiates full-screen interactive MANUAL keyboard mode, or terminal emulation mode. Upon entering MANUAL mode, a preliminary screen containing instructions is

displayed. When the preliminary screen is acknowledged, the full terminal emulation screen is displayed. MANUAL commands can occur anywhere in the dialog, and apply immediately to the point in which they occur.

When in MANUAL mode, keyboard macro programs, such as ProKey and Superkey, can be used to further automate terminal input.

**Availability**      DOS

### Parameters

Parameter Name	Explanation
NOPROMPT	If the optional parameter NOPROMPT is included in the command, the screen immediately displays the 3270 screen, bypassing the prompt screen that describes MANUAL mode.

### Example

```
MANUAL [NOPROMPT]
```

## NOBREAK

---

**Syntax**            NOBREAK

**Description**      The NOBREAK command prevents the CTRL+BREAK key from interrupting the dialog. This command can be repeated in the dialog and applies to any subsequent commands until a BREAK command is encountered or the end of the dialog is reached.

**Availability**      DOS

### Example

```
NOBREAK
```

## OFF

---

**Syntax**            OFF Condition

**Description**      The OFF command turns off a previously issued ON command. If no previous ON command was issued, then the OFF command is ignored.

**Availability**      DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
Condition	Any one of the following conditions: TIMEOUT, RUNOUT, BREAK, PROG $n$ , EVENT $n$ .
TIMEOUT	This turns off any previous ON TIMEOUT. If a TIMEOUT occurs without an active TIMEOUT, the dialog ends with a TIMEOUT error.
RUNOUT	This turns off any previous ON RUNOUT. If a RUNOUT occurs without an ON RUNOUT active, the dialog ends with a RUNOUT error.
BREAK	This turns off any previous ON BREAK. If you press CTRL+BREAK without an active BREAK, the DIALOG ends with a CTRL+BREAK pressed message. Note that if a NOBREAK command is issued, neither ON BREAK nor OFF BREAK has any effect.
PROG $n$	This turns off any previous ON PROG $n$ issued for that specific PROG key exit. You must specify PROG1, PROG2, or PROG3.
EVENT $n$	This turns off any previous ON EVENT $n$ issued for that specific 'text' EVENT. You must specify EVENT1, EVENT2, or EVENT3.

## Example

```
OFF PROG1
```

## OIA

---

**Syntax** OIA 'String'

**Description** The OIA command allows the DIALOG to define a 1- to 33-byte message that appears in bytes 41 to 73 of the Operator Information Area, line 25, of the 3270 screen. The message must be enclosed within single quotes if you wish lowercase characters to be displayed.

**Availability** DOS

## Parameters

Parameter Name	Explanation
String	The text <i>String</i> must be enclosed within single quotes if you want lowercase letters to be displayed.

## Example

```
OIA 'your message goes here'
```

## ON BREAK GOTO

---

**Syntax**            ON BREAK GOTO *Label*

**Description**      The ON BREAK GOTO command moves the control flow of the dialog to the command immediately following *Label* when the user presses the CTRL+BREAK key. If the NOBREAK command has been issued, then CTRL+BREAK is ignored and ON BREAK GOTO has no effect.

**Availability**      DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
Label	Specifies a valid DIALOG label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

### Example

```
ON BREAK GOTO RETRY_DIAL
```

## ON EVENT GOTO

---

**Syntax**            ON EVENT $n$  [(*rr,cc,ll*)] '*text*' [OR '*text2*'] ... [OR '*textn*'] GOTO label

**Description**      ON EVENT GOTO initiates action (the commands immediately following *Label*) to be taken whenever the presence of a specified *Text* string is found on the screen.

You can specify up to 3 different text events; the command must indicate which event by specifying EVENT1, EVENT2 or EVENT3. Whenever the *Text* is found on the screen, whether in regular SEND/WAITFOR mode or MANUAL mode, the GOTO specified in the command is executed. The GOTO specifies the label where execution continues if the *Text* event is found. Once the *Text* event is found, the ON EVENT GOTO command is turned off. To reestablish ON EVENT GOTO, the command must be executed again in the dialog. Note that it can simply be the previously issued ON EVENT GOTO being looped through again.

Optionally, multiple *Text* strings can be related with OR. If any one of the *Text* strings is found on the screen, the GOTO is executed. Any number of strings can be connected in this way. Up to 64 bytes can be used for all of the strings, counting each OR as one byte.

Another optional facility, using (*rr,cc,ll*), restricts the search for an event to a specific area of the screen. The search area cannot wrap the screen. The *Text* string must be found, in its entirety, within the specified area. The string cannot start inside the area and end outside the area. Limiting the search this way greatly decreases the overhead of ON EVENT GOTO in the DIALOG.



**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
n	The number of the <i>Text</i> string. The maximum limit of all strings for one ON EVENT <i>n</i> GOTO, including all OR statements is 64 bytes.
Text	The string that Dialog is looking for. The maximum length of <i>Text</i> is 64 bytes. <i>Text</i> must be enclosed within single quotes. No uppercase translation is performed on <i>Text</i> .
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.
rr	Specifies the beginning row of the search. <i>rr</i> can range from 1 to 24.
cc	Specifies the beginning column of the search. <i>cc</i> can range from 1 to 80.
ll	Specifies the length of <i>Text</i> to be searched. If <i>ll</i> is not specified then the search will start at <i>rr,cc</i> , and continue to the end of the screen.

### Example 1

```
ON EVENT2 'DFH' GOTO CICS-ERROR
```

This example performs a simple check for an event.

### Example 2

```
ON EVENT1 'ABEND' OR 'ABORT' GOTO ABEND-EXIT
```

This example performs a check for one of multiple events.

### Example 3

```
ON EVENT3(1,1,4) 'CSSN' GOTO SIGN-ON
```

This example performs a check at a specific location.

### Example 4

```
ON EVENT1(24,1,80) 'SIGN OFF IS COMPL' GOTO EXIT
```

This example performs a check over a range of locations.

### Example 5

```
ON EVENT2(4,1,8) '8/1' OR '8/01' GOTO AUG
```

This example performs a check over a range for one of multiple events.

## Example 6

```
ON EVENT1(24,1,4) '//xx' OR &ALT-EVENT GOTO EXIT-1
```

This example performs a check using the &variable &ALT-EVENT.

## ON PROG GOTO

---

**Syntax**            ON PROG $n$  GOTO *Label*

**Description**        The ON PROG $n$  GOTO form of ON allows specific exits from the MANUAL mode whenever you press certain PC keys. The configuration program, MAINCON, allows you to configure the 3270 keyboard for MANUAL mode. You can establish up to 3 keys called PROG1, PROG2 and PROG3 to be special MANUAL mode exit keys. The ON PROG $n$  command allows you to specify where (at which label) processing is to continue after an exit from the MANUAL mode. The special exit(s) from MANUAL mode takes place if the following requirements are met:

A PROG $n$  key has been configured in MAINCON.

The ON PROG $n$  GOTO command has been executed before the MANUAL command.

The user presses the PC key(s) associated by MAINCON with PROG $n$ .

If a PROG $n$  key was configured in MAINCON and no ON PROG $n$  GOTO command is in effect, then pressing the key(s) assigned in MAINCON to PROG $n$  will have no effect. A beep sounds, but the operator stays in MANUAL mode. Conversely, if ON PROG $n$  was specified in the dialog script and no PROG $n$  key was configured by MAINCON, then no exit from the MANUAL mode occurs.

The configuration file produced by MAINCON for "Video Monitor Mode" is named MAINCFG.VDT. This file must either be in the current directory or be pointed to by the SET EDMSYS command.

**Availability**        DOS

### Parameters

Parameter Name	Explanation
n	Specifies the manual mode exit key. The value of $n$ may be 1, 2, or 3.
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

### Example

```
ON PROG1 GOTO EXIT
```

## ON RUNOUT GOTO

---

**Syntax**            ON RUNOUT GOTO *Label*

**Description**        The label specified by ON RUNOUT GOTO command indicates where execution will continue if too many GOTOs occur for IF or GOTO. The last ON RUNOUT GOTO encountered before the RUNOUT occurs is the one whose GOTO will be executed.

**Availability**        DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

### Example

```
ON RUNOUT GOTO WARNING
```

## ON TIMEOUT GOTO

---

**Syntax**            ON TIMEOUT GOTO *label*

**Description**        This form of ON indicates that if a TIMEOUT condition occurs for any WAITFOR or SEND command, the commands after the *Label* specified by GOTO will be executed. Multiple ON TIMEOUT GOTO commands can be used within a dialog. The last ON TIMEOUT GOTO encountered before the TIMEOUT occurs is the one whose GOTO will be executed.

**Availability**        DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
Label	Specifies a valid Dialog label. Labels identify positions in the dialog. Only spaces can precede the label on the line. The label itself must begin with a colon (:) and end with a space. The maximum label length, excluding the colon, is 10 characters. Only comments (*) can follow the label on the label line. Commands cannot be placed on a label line.

## Example

```
ON TIMEOUT GOTO MESSAGE
```

## OPTION

---

**Syntax**            OPTION LANGLEVEL(*Lang*)

**Description**      The OPTION command establishes characteristics for dialog execution. LANGLEVEL identifies the language level of the dialog. If your dialog uses any variables other than prompted variables, you must include an OPTION command in your dialog that specifies LANGLEVEL(2).

**Availability**      DOS

### Example

```
OPTION LANGLEVEL(2)
```

## PAUSE

---

**Syntax**            PAUSE [ '*Message*' ]

**Description**      The PAUSE command causes the rest of the command line after PAUSE to be displayed on the PC. The dialog waits until the user presses any key before continuing. This allows you to set up a message that requires an acknowledgment by the user. When this command is encountered in the dialog, the *Message* is displayed on the screen followed by <<Press any key to continue>> on the next line.

PAUSE commands may occur anywhere in the dialog. They immediately apply to their location in the dialog. When the screen is in PC mode, the messages are displayed at the current line and remain visible until they are scrolled off the screen. When the /V option is on to display the 3270 screen, the messages are placed on the screen following the cursor, and then remain on the screen until the next screen output is created.

**Availability**      DOS

### Parameters

Parameter Name	Explanation
Message	The text string that will be displayed on the screen. If <i>Message</i> is enclosed within single quotes, it is displayed in uppercase and lowercase. If the message is not enclosed in single quotes, it is translated into uppercase.

### Example

```
PAUSE 'Waiting for your response'
```

## PRINT

---

**Syntax** PRINT FILE(*d:\path\filename.ext*) [ADD-ON]  
 or  
 PRINT SCREEN [PAGE]  
 or  
 PRINT '*Literal*'|*Variable*...[NEW-LINE]

**Description** The PRINT command sends screen images or literals to either the printer at LPT1 or a disk file. The PRINT command must be followed by one of the parameters listed below.

**Availability** DOS

### Parameters

Parameter Name	Explanation
FILE	This command can be given only once in a DIALOG script. It indicates the file where all subsequent PRINT commands will be sent. If this statement is not present, the output from all subsequent commands will be sent to LPT1, the printer. If you need to route PRINT output to a printer other than LPT1, use the command PRINT FILE(LPT2) or PRINT FILE(COM1).
ADD-ON	This optional parameter causes the data to be appended to the end of the file named. If the parameter is not present, the file is erased prior to collecting the data.
SCREEN	This command causes the current screen to be sent to the destination defined by a previous PRINT FILE command or the printer if no PRINT FILE was issued.
PAGE	This optional parameter causes a form feed character, hex 0c, to be sent to the destination after the screen image.
' <i>Literal</i> '  <i>Variable</i>	This variation of the PRINT command causes the contents of the listed variables and/or literals to be sent to the output destination.
NEW-LINE	This optional parameter causes carriage return and line-feed characters, hex 0D 0A, to be sent to the output destination following the literal. Control characters may be sent by entering them within the single quotes.

### Example

```
PRINT SCREEN PAGE
```

## PROMPT

---

**Syntax** PROMPT ['*Message*']

**Description** The PROMPT command causes the dialog to prompt the user for a one-character response. The response is not case sensitive. The difference between the PROMPT and PAUSE commands is that the user's response to PROMPT can be tested by an IF command.

Unlike prompted variables, which are resolved before the dialog is actually run, the PROMPT command is executed during the dialog. Therefore, the PROMPT command would not be suitable for an unattended DIALOG where no operator is present to respond.

**Availability** DOS

## Parameters

Parameter Name	Explanation
Message	The text string that will be displayed on the screen. If <i>Message</i> is enclosed within single quotes, it is displayed in mixed case. If the message is not enclosed in single quotes, it is translated into upper-case.

## Example 1

```
PROMPT 'Press Y for yes or N for no.'
```

## Example 2

```
:MENU
REM 1. Choice one
REM 2. Choice two
REM 3. Choice three
REM 4. Choice four
PROMPT Please enter your choice:
IF {1} GOTO got1
IF {2} GOTO got2
IF {3} GOTO got3
IF {4} GOTO got4
GOTO PROMPT
```

In example 2, a menu of choices is displayed by the REM command and a response is obtained via the PROMPT command. The subsequent IF commands test the response.

# QUIT

---

**Syntax** QUIT[(Return\_Code)] ['Return\_Message']

**Description** The QUIT command:

- Terminates the dialog immediately when encountered (except when the Return\_Code value is set to 8).
- Sets the DOS ERRORLEVEL, testable by a DOS Batch file, in DOS, or the DLG\_RESULT field in Windows and OS/2.
- Sends a text message to the console in DOS, or sets the DLG\_MESSAGE field in Windows and OS/2. In Windows and OS/2 the text terminates with a carriage return (hex 0D), line feed (hex 0A) and null (hex 00).

**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
Return_Code	May be an integer from 0 to 255. The default value is 4. A <i>Return_Code</i> value of 8 sets the TIMEOUT counter to 0 and restarts the dialog. If you specify any other <i>Return_Code</i> value for QUIT, make the value greater than 100 to avoid confusion with any of the standard DIALOG result codes.
Return_Message	<i>Return_Message</i> must be enclosed in single quotes and not contain any embedded single quotes. The text can be comprised of any ASCII characters from space (hex 20) through delta (hex 7F). The length of the text can be any length that fits in the normal limit of 255 bytes per command.

## Example

```
QUIT(101) 'LOGON failed'
```

In the above example, when the quit command is encountered, the dialog will terminate as follows:

- If run in DOS, the ERRORLEVEL will be 101, and the message LOGON failed will be displayed on the screen.

If run from Windows or OS/2, DLG\_RESULT will be set to 101, and DLG\_MESSAGE will contain LOGON failed, followed by CR, LF and null.

## REMARK

---

**Syntax**            REMARK 'Message'

**Description**     The REMARK command causes the rest of the command line after the REMARK command to be displayed on the PC screen. REMARK commands can occur anywhere in the dialog. They immediately apply to their location in the dialog.

When the screen is in PC mode, the messages are displayed at the current line and remain visible until they are scrolled off the screen. When the /V option is on to display the 3270 screen, the messages are placed on the screen, and then remain on the screen until the next screen output is created.

**Availability**     DOS

## Parameters

Parameter Name	Explanation
Message	The text string that will be displayed on the screen. If <i>Message</i> is enclosed within single quotes, it is displayed in mixed case. If the message is not enclosed in single quotes, it is translated into uppercase.

## Example 1

```
REM `Operator Information Message`
```

This example causes `Operator Information Message` to display on-screen.

## RETURN

---

**Syntax**            `RETURN`

**Description**      The `RETURN` command returns execution to the statement after the last active `GOSUB`.

**Availability**     `DOS`

### Example

```
RETURN
```

## RUN RUN(P)

---

**Syntax**            `RUN ['String'] ['String'] ...`

**Description**      The `RUN` command allows the dialog to execute a program so that the result code from that program can be interrogated by the `IF ERRORLEVEL` command. Multiple strings or variables can be specified. If the program is not in the current directory, the full path must precede the program name.

The `RUN` command differs from the `EXECUTE` command in the following ways:

- No batch files or MS-DOS commands can be executed via `RUN`; only `.EXE` and `.COM` files can be executed.

- The result code (if any) from the program executed via `RUN` can be tested by an `IF ERRORLEVEL` command.

- The program name and any passed execution parameters must be specified as separate strings.

- The MS-DOS path is not searched by `RUN`.

The `RUN(P)` format instructs `DIALOG` to pass `DEVICE` and `LINE` parameters to the program being `RUN`.

**Availability**     `DOS`

### Parameters

Parameter Name	Explanation
String	The first <i>String</i> or variable must be a program name. If neither <code>.EXE</code> nor <code>.COM</code> is appended to the program name, then <code>DIALOG</code> defaults to <code>.EXE</code> .



## Example

```
RUN 'C:\PCM\MAIN' 'SETUP.PCM'
```

In the above example, the program name, MAIN, is enclosed in a separate string from the parameter, SETUP.PCM, that is passed to MAIN.

## RUNOUT

---

**Syntax** RUNOUT(Limit)

**Description** The RUNOUT command limits the number of GOTO or GOSUB jumps that can be executed by a dialog. This allows you to set limits to avoid infinite loops. Any GOTO or GOSUB executed, whether part of an IF command or as a command itself, counts as one GOTO jump; all such jumps accumulate toward the specified limit.

Multiple RUNOUT commands can be used within a dialog. With multiple RUNOUT commands, the limit is set at the point where the RUNOUT command is encountered, and remains in force until the next RUNOUT command is encountered or the end of the dialog is reached.

If an ON RUNOUT command has been specified, the GOTO command for that ON RUNOUT is executed when a RUNOUT limit is exceeded. When the ON RUNOUT GOTO is executed, the new RUNOUT limit remains at the last activated limit, and the current count is set to 0. If no ON RUNOUT is encountered and the RUNOUT limit is exceeded, the dialog is terminated with an ERRORLEVEL of 6.

**Availability** DOS, WIN, NT, WIN95, OS/2

### Parameters

Parameter Name	Explanation
Limit	The default RUNOUT <i>Limit</i> is 1024. The maximum <i>Limit</i> is 64,000. If RUNOUT(0) is specified, then no limit is enforced. Each time that the RUNOUT command is encountered, the current count is reset to 0.

### Example

```
RUNOUT (100)
```

This example sets the RUNOUT limit to 100 and resets the current count to 0.

## SEND

---

**Syntax** SEND Keystrokes

**Description** The SEND command sends keystrokes to the communications device. These commands can occur anywhere in the dialog. No data is sent to the host until an attention key (ENTER, PF1 - PF24) is sent. Always include an attention key at the end of any SEND or series of SENDs that are transmitted to the host.

**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
Keystrokes	The data to be sent to the host. <i>Keystrokes</i> can be represented by one or more of the following four items: String, Variable, Keywords, Linedrop.
String	Text <i>String</i> must begin and end with a single quote. Any apostrophes within <i>String</i> must be doubled. No uppercase translation is done; the text is sent as it appears in <i>String</i> .
Variable	<i>Variable</i> names begin with a \$, &, # or %, and are resolved prior to execution of the dialog. <i>Variable</i> names within SEND are treated the same as literals. SEND keywords cannot be specified with a variable.
Keyword	<i>Keyword</i> specifies attention keys, cursor movement, and other special keys. Any SEND item not enclosed in quotes and not beginning with \$ or & is considered a <i>Keyword</i> .
Linedrop	LINEDROP causes a 5 second line break followed by the dropping of the Ready To Send (RTS) and the Data Terminal Ready (DTR) signals on the serial port being used. This option only applies to protocol converters. The objective of this command is to force a hang up of the phone line for dial-up connections and to disconnect from a locally attached converter. Once the LINEDROP has been sent, no further connection is possible without re-dialing.

### Example 1

```
SEND Home 'TRANS1' Tab 'LIST' Enter
```

The above example positions the cursor to the home location on the screen, types the string TRANS1, tabs to the next field on the screen, types the string LIST and transmits the ENTER key.

### Example 2

```
SEND &VTAM-application Enter
```

The above example prompts the user to enter a VTAM-application at the start of the dialog, prior to execution. When this SEND is encountered, the user's response is sent in place of &VTAM-application.

### Example 3

```
SEND &CICS-name Tab $CICS-password Enter
```

The above example prompts the user for a CICS-name and CICS-password at the start of the dialog, prior to execution. The response to CICS-name is displayed on the screen, while the response to CICS-password is not.

### Example 4

```
SEND LINEDROP
```

### Example 5

```
SEND 'LOGON APPLID&ZLOGON' ENTER
```

## SET

---

**Syntax** SET Variable = [Variable] ['String?'] [Variable ['String?']...]

**Description** The SET command concatenates the values of the specified items and assigns the resulting string to a variable.

The equal sign (=) must be preceded and followed by at least one space.

**Availability** DOS

### Parameters

Parameter Name	Explanation
Variable	A valid Dialog variable. For SET, <i>Variable</i> prefix (for example: #) must be omitted from the target <i>Variable</i> (the variable to the left of the equal sign). If target <i>Variable</i> already exists, the original contents are lost. This applies to prompted as well as common variables.
String	Quotation marks can be omitted if the desired <i>String</i> contains no spaces.

### Example

```
SET FIRST = JOHN
SET LAST = DOE
SET MESSAGE = 'FIRST NAME: ' #FIRST ' LAST NAME: ' #LAST
REM #MESSAGE
```

The above command sequence displays the following message:

```
FIRST NAME: JOHN LAST NAME: DOE
```

## TIMEOUT

---

**Syntax** TIMEOUT(*sss*)

**Description** The TIMEOUT command sets the number of seconds that DIALOG will wait for a WAITFOR condition, SEND, or ON.

The TIMEOUT command can occur multiple times within a dialog. TIMEOUT applies to all WAITFOR, SEND and ON commands until the next TIMEOUT command or the end of the dialog is reached.

**Availability** DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
sss	If this parameter is set to zero, no TIMEOUT occurs, and the user must press CTRL+BREAK to exit the dialog when a response condition is not found true. The maximum value of sss is 255 seconds. The default is 30 seconds.

### Example 1

```
TIMEOUT (45)
```

### Example 2

```
TIMEOUT (&ZTIMEO)
```

## WAITFOR

---

**Syntax**            WAITFOR [(*rr,cc[,ll]*)] '*Text*' [OR '*Text2*'.[OR '*Textn*']]

**Description**      The WAITFOR command causes the dialog to wait until the specified text appears on the screen. WAITFOR commands can occur anywhere in the dialog. They apply immediately to their location in the dialog. Using *rr,cc,ll* restricts the search for an event to a specific area of the screen. This greatly decreases the overhead of WAITFOR in the dialog.

**Availability**      DOS, WIN, NT, WIN95, OS/2

## Parameters

Parameter Name	Explanation
Text	The string Dialog is looking for. <i>Text</i> must be enclosed within single quotes. No uppercase translation is performed on <i>Text</i> . If the search area is specified, the <i>Text</i> string must be found, in its entirety, within the specified area. The search area cannot wrap the screen.
n	The number of the <i>Text</i> string. <i>Text</i> strings must be separated with the OR keyword. You may specify any number of <i>Text</i> strings as long as they fit on a single command line. Each <i>Text</i> must be enclosed within single quotes.
rr	Specifies the beginning row of the search. <i>rr</i> can range from 1 to 24.
cc	Specifies the beginning column of the search. <i>cc</i> can range from 1 to 80.
ll	Specifies the length of <i>Text</i> to be searched. If <i>ll</i> is not specified then the search will start at <i>rr,cc</i> and continue to the end of the screen.

## Example 1

```
WAITFOR CLEAR
```

This form of WAITFOR requests a wait until a state is detected that has the following conditions:

- The keyboard is unlocked.
- The screen is clear of any data and format field attributes.

The cursor is in row 1 column 1, the upper left hand corner of the screen.

## Example 2

```
WAITFOR RESPONSE
```

This form of WAITFOR requests a wait for any response from the host. “Any response” is determined by checking to see that the 3270 emulator is not in the SYSTEM WAIT mode. The check is made for the length of time specified in the last FLICKER command.

When WAITFOR RESPONSE is used with serial devices, such as protocol converters, it only ensures that some data or cursor movement is received and is followed by a period of time equal to the FLICKER interval. This does not ensure a complete incoming message. Use WAITFOR *Text* with serial devices to be certain that the incoming message is complete.

## Example 3

```
SEND 'AT DT '
&PHONE-NO ENTER
WAITFOR 'CONNECT'
OR 'BUSY' IF 'CONNECT'
GOTO ON_LINE
IF 'BUSY' GOTO TRY_LATER
```

This above example demonstrates the use of WAITFOR *Text* instead of WAITFOR RESPONSE with a serial device.

If *Text* is not found, or if RESPONSE is not received within the interval specified by the last TIMEOUT command, one of the following two things will occur:

- The GOTO associated with the last ON TIMEOUT GOTO command will be executed. Or, The dialog will terminate with an error message if no ON TIMEOUT GOTO is encountered.

## Continuity Between Calls

---

The following dialog command settings remain in effect from call to call:

- DEVICE
- TIMEOUT
- RUNOUT
- FLICKER
- KEYDELAY

OIA

For the above commands, the last values set by the dialog script just executed remain in effect for the next call to EDMDIAL. Values are static from call to call.

The following dialog command settings do not remain in effect from call to call:

- ON
- TIMEOUT/RUNOUT/BREAK/EVENT $n$ / PROG $n$
- BREAK
- NOBREAK

PRINTFILE

**Note:** When an application program calls EDMDIAL multiple times, the first script passed must contain a DEVICE command to identify the communications device. Subsequent scripts need not contain the command.

## Predefined Variables

---

### Simple Predefined Variables

Simple predefined variables have specific, system-defined values. In all cases, the contents of the variable are established when the referencing command is executed.

Name	Length	Format
@DATE	10	MM/DD/YYYY
@TIME	8	HH:MM:SS
@MODEL	1	m (3270 model 2,3,4, or 5)
@ROW	2	rr (cursor row)
@COLUMN	3	ccc (cursor column)
@ERRORLEVEL	3	nnn (result of last RUN command)
@DEVICE	3	ddd (as in DEV command)
@RESPONSE	1	r (response from last PROMPT)
@HOST	1	h (host session, for example: A)

### Complex Predefined Variables

Complex predefined variables are simply functions; that is, they have subparameters which further define their action. When the subparameter is a variable, any type other than another complex predefined variable is acceptable. For subparameters other than variables, a numeric literal, or a numeric variable, is required. An acceptable form follows:

#### Example

```
SET ROW = 10
```

```
SET COL = 20
REM @SCREEN(#ROW,#COL,10)
```

There are three complex predefined variables: @FIELD, @SCREEN, and @SUBSTR.

Variable and Syntax	Description
@SCREEN( <i>row,col,len</i> )	This extracts data from the device image buffer at the specified position, for <i>len</i> characters. Attributes and control characters are returned as spaces.
@FIELD( <i>row,col,len</i> )	Same as @SCREEN, but truncates at the first field attribute encountered or the specific length, whichever occurs first.
@SUBSTR( <i>variable,start,len</i> )	This describes the substring of the first parameter beginning with position <i>start</i> for <i>len</i> characters.

## Example

```
SET START = 3
CALC LEN = 1 + 1
SET FIELD = ABCDE
REM @SUBSTR(#FIELD,#START,#LEN)
```

## Numeric Expressions

---

EDM DIALOG expressions may be used with the CALC and IF NUM commands. Expressions consist of:

- Numeric variables.
- Numeric integers.
- Operators.

Parentheses.

Integers may be signed or unsigned. All calculated values  $x$ , whether intermediate or final, must fall in the following range:

$$-4,294,967,296 \leq x \leq 4,294,967,296$$

## Numeric Variables

Numeric variables are variables or any type (common, predefined, etc.) whose contents represent valid numeric integers. The contents of the variable must be in the following form:

[ - ] nnnnnnnnnn

## Example

```
SET X = 50 ;SETS #X TO 50
SET Y = @ROW ;SETS #Y TO CURRENT SCREEN ROW
SET Z = @SUBSTR(@TIME,1,2) ;SETS #Z TO HOUR OF DAY
SET Q = -1 ;SETS #Q TO MINUS 1
```

Numeric variables also result when a variable is the target variable of a CALC command. The contents of such result variables is always in minimal form, meaning there are no leading zeros, and a sign is present only if the result is negative.

### Example

```
SET A = -001
SET B = -002
SET C = +002
CALC D = #A * #B           ;D = 2, not +2 or +002
CALC E = #A * #C           ;E = -2, not -002
```

Notice that in the above examples, a semicolon (;) precedes a comment instead of an asterisk (\*). Do not use an asterisk when commenting on any command containing an expression since it conflicts with the multiply operator.

## Operators

### Valid Operators (from high to low priority)

Operator	Description	Operator	Description
+	Unary plus	GT, >	Greater than
-	Unary minus	GE, >=, =>	Greater than or equal to
*	Multiply	NE!=", <>, ><	Not equal
/	Divide	NOT, !	Unary logical NOT
+	Add	AND, &&	Logical AND
-	Subtract	OR,	Logical OR
EQ	Equal to	XOR	Logical XOR
LT, <	Less than		

The relational and logical operators are generally used only with the IF NUM command.

### Example

```
SET NBR = 5
IF NUM (#NBR > 3) AND (#NBR < 10) THEN
REM 'Welcome' &LOGONID
ENDIF
```

Relational operators will evaluate to zero if the relationship is false, and to one if the relationship is true.

### Example

```
CALC RESULT = 1 > 2           ;RESULT WILL BE 0
CALC RESULT = 1 < 2           ;RESULT WILL BE 1
```

Logical operators consider any non-zero operand to be true, and any zero operand to be false. Logical operators evaluate to one or zero, if the result is true or false respectively.



## Example

```
CALC RESULT = 1 AND 0           ;RESULT WILL BE 0
CALC RESULT = 1 OR 0            ;RESULT WILL BE 1
```

## Send Keywords

---

### Valid SEND Keywords

Keyword	3270 Meaning	Keyword	3270 Meaning
ENTER	ENTER key	DUP	Duplicate
PF1 through PF24	PF1 through PF24	MARK	Mark
PA1 through PA3	PA1 through PA3	INS	Insert toggle
CLEAR	Clear	DEL	Delete
SYSREQ	System Request	TAB	Tab forward
TEST	Test	BACKTAB	Tab backward
ATTN	Attention	HOME	Home position
UP	Up arrow	BS	Back space
DOWN	Down arrow	NEWLINE	New line
LEFT	Left arrow	RESET	reset
RIGHT	Right arrow	ERASE-INPUT	Erase input
ERASE-EOF	Erase to end of field	ID	ID
PRINT	Print	DEVCON	Device cancel

## Sample EDM Dialog Files

---

EDM Client includes several DIALOG files. The most commonly used two are CONNECT.DIA and DISCONN.DIA. These files are listed on the following pages for DOS, Windows 3.x, Windows NT, Windows 95, and OS/2.

### CONNECT.DIA for DOS

```
TIMEOUT (&ZTIMEO)
DEV (&ZDEVTP)
host (&zhost)
KEYDELAY (00.01)
SEND CLEAR
WAITFOR RESPONSE
SEND 'LOGON APPLID&ZLOGON' ENTER
WAITFOR 'COMMAND ==>'
SEND 'TRANSFER' ENTER
WAITFOR 'X'
```

### CONNECT.DIA for Windows 3.x, Windows NT, Windows 95, OS/2

```
DEV (&ZDEVTP)
HOST (&ZHOST)
KEYDELAY (00.01)
TIMEOUT (&ZTIMEO)
SEND CLEAR
```

```

SEND 'Logon APPLID&ZLOGON' ENTER
WAITFOR 'COMMAND ==>' OR 'EDM:'
if 'COMMAND ==>' goto SND_TRANS
if 'EDM:' goto SND_RETRY
QUIT(16)
:SND_TRANS
SEND 'TRANSFER' ENTER
WAITFOR 'X'
QUIT(0)
:SND_RETRY
QUIT(8)

```

### **DISCONN.DIA for DOS**

```

TIMEOUT (&ZTIMEO)
DEV (&zdevtp)
HOST (&zhost)
SEND PF3
WAITFOR 'COMMAND'
SEND PF3
WAITFOR RESPONSE

```

### **DISCONN.DIA for Windows 3.x, Windows NT, Windows 95, OS/2**

```

TIMEOUT (&ZTIMEO)
DEV (&zdevtp)
HOST (&zhost)
RUNOUT (5)
ON RUNOUT GOTO EXIT
:LOOP
SEND PF3
WAITFOR RESPONSE
IF 'COMMAND ==>' GOTO WRAPUP
GOTO LOOP
:WRAPUP
SEND PF3
WAITFOR RESPONSE
:EXIT

```

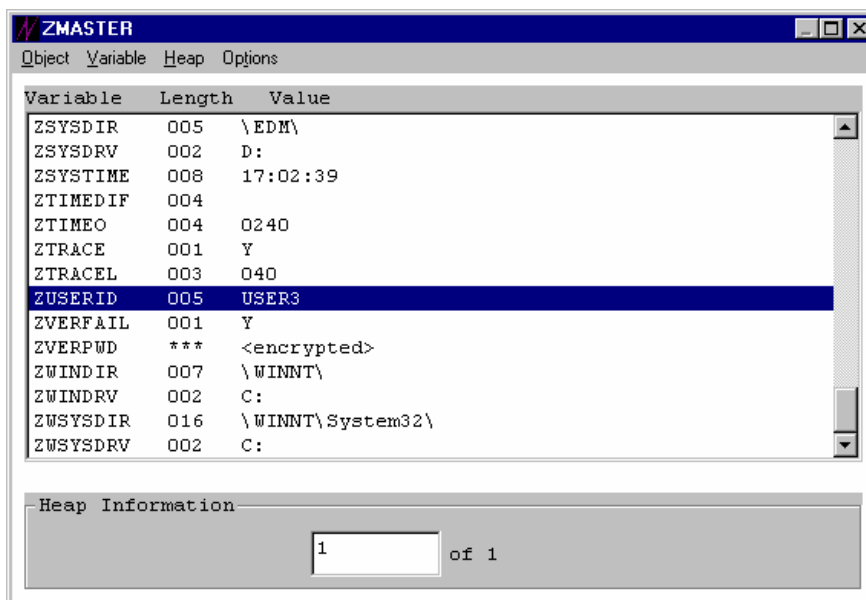
# Understanding the Resolution Process

The EDM Manager employs a procedure called *the resolution process* to accomplish a unit of work in response to a service request. The unit of work is defined by the contents of the EDM database, and parameters included in the service request itself. In other words, what EDM does depends upon what information is stored in its database, and what information accompanies the request for EDM to perform some action.

For example, the EDM Client Connect submits service requests to the EDM Manager, and the EDM Manager performs a *resolution process* in response to each such request.

One way that the EDM Manager recognizes a service request is when it receives an *object*. An object is simply a storage structure, that is, a record or record set. It contains variables, including both their definition (variable name, length, type, etc.), and their value (for example, the ZUSERID variable contains the user's userid). An object consists of *heaps* (called *instances* in some contexts). A heap is an occurrence of a set of variables. It is analogous to a record in a flat file, a row in a relational database table, or a row in an array. Objects can consist of a single heap (for example, the ZMASTER object), or multiple heaps (for example, the ZRSOURCE object).

You can inspect (and modify) desktop objects using the EDM Client Explorer. Here is a view of the desktop ZMASTER object as it might look when opened in the EDM Client Explorer:



According to the Heap Information in the example, the ZMASTER object consists of a single heap. The scrolling list shows the variables contained in the ZMASTER object, including each variable's name, length (in characters), and current value. This ZMASTER object contains a variable named ZUSERID which is 5 characters in length, and whose current value is USER3.

The following excerpt from the EDM Client Connect script shows the initiation of a service request from the EDM Client to the EDM Manager. The EDM Client sends the EDM Manager the ZMASTER object from the client desktop:

```

CONNECT.EXT - Notepad
File Edit Search Help
*****
** Begin Resolution Process for ZMASTER **
*****
setind 19
write user window (10,5, EDM:Manager is Identifying You ...)
&(ZCONVARS.ZSYS)edmaster.exe ZMASTER 0
* Program: edmaster ZMASTER RC: &(ZBRC) MRC: &(ZMRC)

```

The **edmaster.exe** program sends the ZMASTER object to the EDM Manager. This initiates a resolution process in the EDM Manager.

The purpose of the resolution process is to:

tailor the generalized specifications contained in the EDM database using a set of input accompanying a service request into a specific set of operations, and perform the operations indicated as a result of the tailoring.

In a sense, the resolution process is an interpreter computer with a few built-in functions. Its functionality can be extended in any way the user desires, by plugging in executable routines (called *methods*) at any desired point in the resolution process.

The built-in functions include:

- **Maintaining a global memory:** Global memory is a transient storage area in the EDM Manager whose contents the EDM Manager maintains for the duration of the resolution process. The resolution process defines variable attributes in objects contained in this global memory, and their current value is maintained there. The current value of a variable attribute is the value it holds as of its most recent reference during the resolution process. During resolution, a variable attribute is defined (by name) in an object in global memory when it is first encountered. If an identically named variable attribute is encountered later in the resolution process, the value for that attribute can be updated in global memory. When the EDM Manager receives an object, it stores the variables, contained in that object, in global memory.

**Note:** Objects can be *persistent* or *transient*. Persistent objects are created in global memory from instances of persistent classes in the EDM Database, or from objects sent to the EDM Manager by the EDM Client. Transient objects are created from instances of transient classes in the EDM Database. A persistent object remains in global memory for the duration of the resolution process, and can be stored in the EDM Database or on the client desktop. A transient object exists only while it is being processed during resolution, and values of variables in a transient object may optionally replace (in global memory) values of identically named variables in the most recently processed (during resolution) persistent object. Whether an object created from an instance in the EDM Database is persistent or transient is determined by the class definition from which the object is instantiated. See “*Working with Classes*” in Chapter 1 of this manual.

- **Maintaining a system message:** The system message is a special variable whose value can be used to conditionally effect the execution of class connections and methods during resolution. Class connections are followed and methods are executed if the class connection attribute name or the method attribute name matches the current value of the system message. Conventionally, EDM starts resolution during the client connect by setting the system message to “EDMSETUP”, which is why you see this as the default name for class connection attributes in the EDM Database. The special name “\_ALWAYS\_” indicates to the resolution process that the connection will be made or the method will be executed no matter what the current value of the system message is.
- **Performing symbolic substitution:** Before the values of attributes are acted upon by the resolution process, the current attribute can have *symbolic substitution* performed on its value. Symbolic substitution consists of replacing symbols in the value of an attribute with the current value of the symbol. A symbol in the value of an attribute is the name of variable in global memory, preceded by an ampersand.

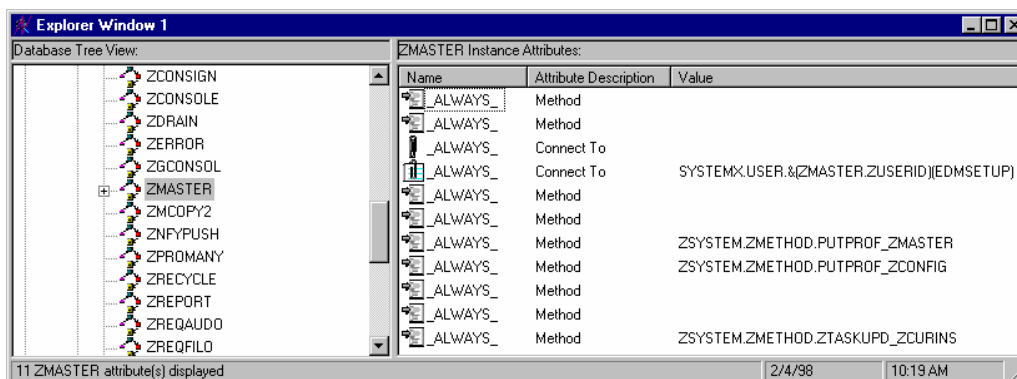
- **Providing an entry point for processing to begin:** When the EDM Manager receives an object, it first stores the variables for that object in global memory. Then it searches the Primary file's ZSYSTEM domain for an instance of the ZPROCESS class whose name matches the name of the object it received. If the instance is found, resolution continues using that instance.

Let's see how this works when the EDM Manager receives a ZMASTER object.

First, the variables contained in the desktop's ZMASTER object are stored in the EDM Manager's global memory. The ZMASTER object in global memory is persistent because it was sent to the EDM Manager by the EDM Client. Based upon the example of USER3's desktop ZMASTER object shown above, the EDM Manager global memory's ZMASTER object would contain the following ("Etc..." indicates that the remaining variables, which are not shown, are also stored in the global memory's ZMASTER object):

Variable Name	Value
Etc...	
ZSYSDIR	\\EDM\
ZSYSDRV	D:
ZSYSTEMTIME	17:02:39
ZTIMEDIF	
ZTIMEO	0240
ZTRACE	Y
ZTRACEL	040
ZUSERID	USER3
ZVERFAIL	Y
Etc...	

Next, EDM Manager locates the PRIMARY.ZSYSTEM.ZPROCESS.ZMASTER instance in the EDM Database, which looks like this:



The EDM Manager then proceeds to resolve this instance. It steps through the instance, attribute by attribute. The first two attributes are empty method attributes (i.e. they have no value). The EDM Manager ignores empty attributes. It simply skips them.

The next attribute is an empty class connection attribute, which the EDM Manager also skips.

The next attribute is a class connection attribute, which has a value. The name of the attribute is the special name “\_ALWAYS\_”, which indicates to the EDM Manager that this class connection should be followed regardless of the current value of the system message. (The system message currently has no value because it hasn't yet been set during this sample resolution).

Before following the connection, the EDM Manager performs symbolic substitution on the value of the class connection attribute. The value is:

**SYSTEMX.USER.&(ZMASTER.ZUSERID)(EDMSETUP)**

The symbol to be substituted is **&(ZMASTER.ZUSERID)**, which will be replaced with the value of the ZUSERID attribute of the ZMASTER object transferred from the client desktop. This value is the userid associated with the client.

Since the client's userid is USER3, the connection value evaluates to:

**SYSTEMX.USER.USER3(EDMSETUP)**

after symbolic substitution.

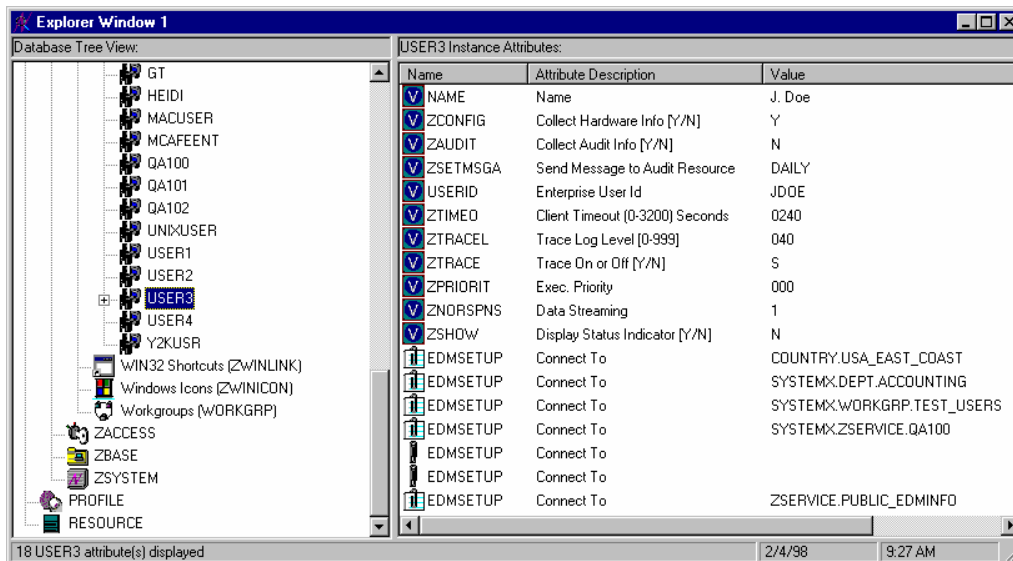
The **(EDMSETUP)** specification in this value directs the EDM Manager to set the value of the system message to:

**EDMSETUP**

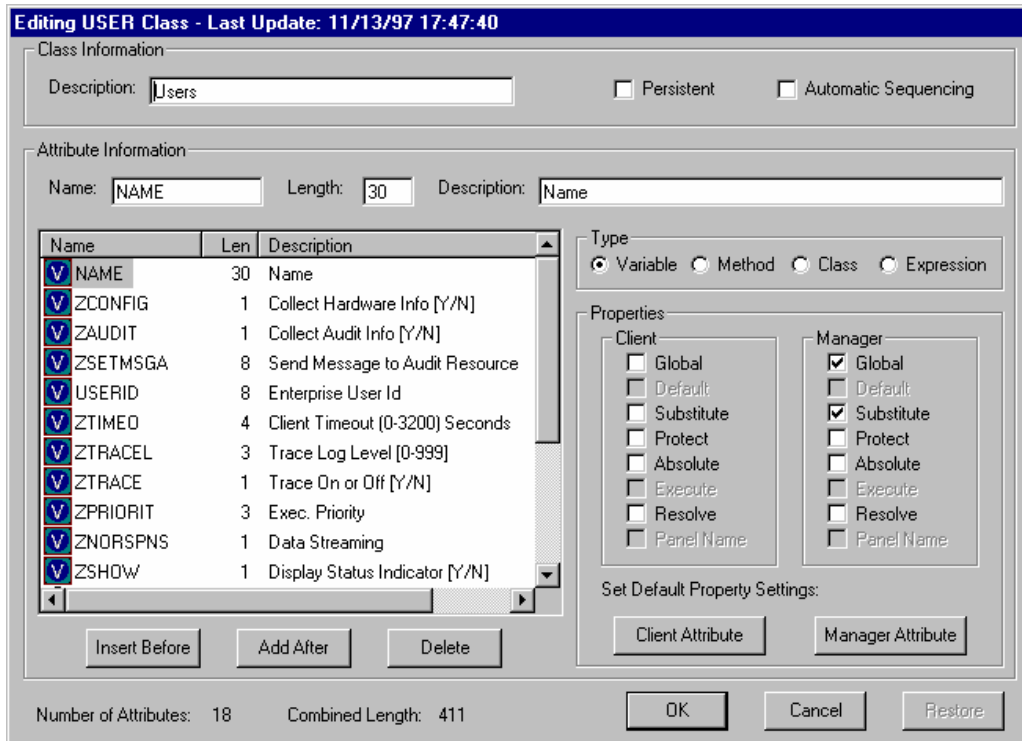
The EDM Manager then continues the resolution with the connected instance,

**SYSTEMX.USER.USER3**

which looks like this:



The first eleven attributes are variables, and the EDM Manager stores them in the ZMASTER object in global memory. Why ZMASTER? At this point, the ZMASTER object is the only (and therefore the most recently processed) persistent object in global memory. The USER object instantiated from the SYSTEMX.USER.USER3 instance is transient, because the class definition for the USER class in the EDM Database lacks the "Persistent" attribute, as follows:



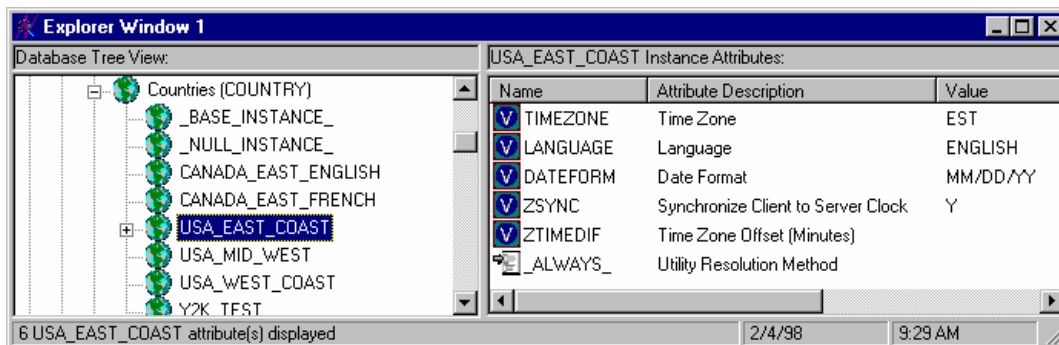
The variable attributes in the User class definition all have the Manager Global property, indicating that values of the variables in objects instantiated from instances of the User class will replace values of identically named variables in most-recently-processed persistent objects, during resolution.

Continuing with our example, the variables in the USER object (NAME, ZCONFIG, ZAUDIT...) are already defined in global memory in the ZMASTER object, so the values of these variables are set in the ZMASTER object in global memory to the values they have in the transient USER object. Later in the Client Connect process, the ZMASTER object will be copied from the EDM Manager global memory to the EDM Client desktop; in this way, changes made by the EDM Administrator to user information in the EDM Database for this client's desktop, are stored on the client desktop.

The next attribute is a class connection whose name is EDMSETUP. This name matches the current value of the system message, so the EDM Manager will follow this connection to the instance:

### COUNTRY.USA\_EAST\_COAST

in the SYSTEMX domain, which looks like this:

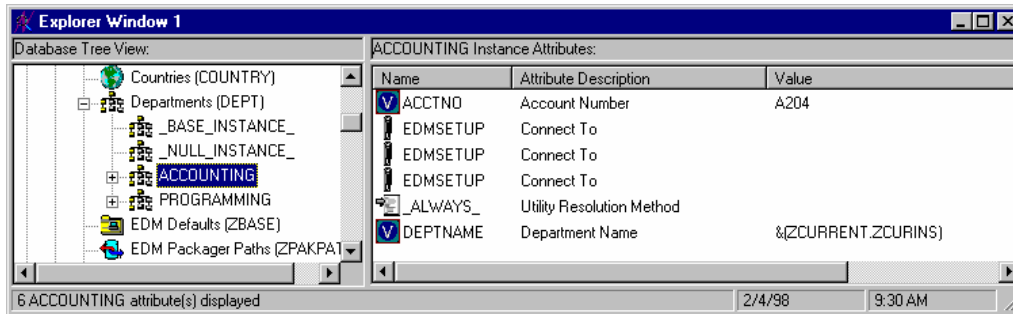


The first five attributes are variables already defined in the persistent ZMASTER object, and the EDM Manager stores their values in the ZMASTER object in global memory. The remaining attribute is an empty method attribute, which the EDM Manager skips because it has no value. After completing the resolution of this instance, the EDM Manager continues resolution with the next as-yet-unprocessed attribute in the SYSTEMX.USER.USER3 instance. This is the

attribute following the class connection attribute that connected with COUNTRY.USA\_EAST\_COAST. It is a class connection attribute, also named EDMSETUP, which connects to:

### SYSTEMX.DEPT.ACCOUNTING

Because the name of the attribute matches the current value of the system message, the EDM Manager will follow the connection to this instance, which looks like this:



The first attribute is a variable which is already defined in the ZMASTER persistent object, and the EDM Manager stores its value in the ZMASTER object in global memory. The next four attributes are empty, so the EDM Manager skips them. The last attribute is a variable, whose value contains a symbol. The EDM Manager performs symbolic substitution on the value, and stores the current value of the ZCURINS attribute of the ZCURRENT object as DEPTNAME in global memory. Then, having completed resolution of this instance, the EDM Manager continues resolution with the next unprocessed attribute in the SYSTEMX.USER.USER3 instance, which is a class connection attribute connecting to SYSTEMX.WORKGRP.TEST\_USERS.

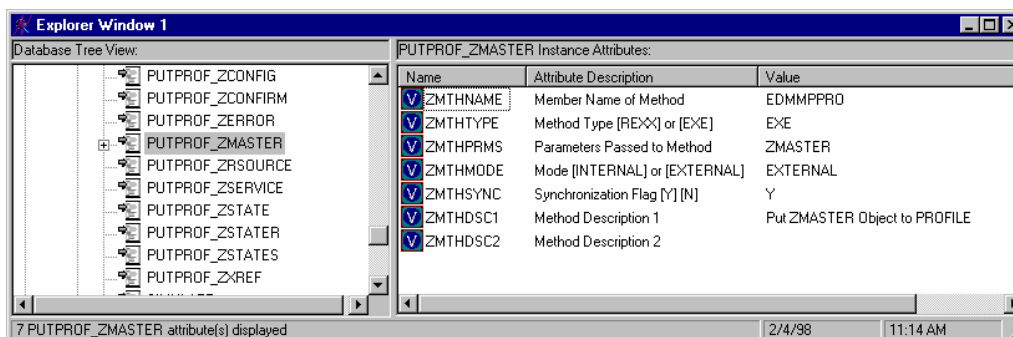
As we've seen, the resolution process steps through connected class instances, attribute by attribute, performing functions as determined by the attributes encountered.

If the EDM Manager encounters a non-empty method attribute whose name either matches the current value of the system message, or whose name is “\_ALWAYS\_”, the EDM Manager will execute the method identified in the attribute. You can write custom methods and plug them into the resolution process wherever you desire. In this way, EDM can be customized to meet any requirement.

Eventually, after all attributes in the SYSTEMX.USER.USER3 instance (and any attributes of instances connected to that instance, and so on...) have been evaluated and processed, resolution returns to the ZMASTER instance, to the attribute following the one which connected to the SYSTEMX.USER.USER3 instance. Since it, and the following attribute have no value, they are skipped. The next attribute is a method attribute named “\_ALWAYS\_”, which calls for the execution of the method specified in the

### ZSYSTEM.ZMETHOD.PUTPROF\_ZMASTER

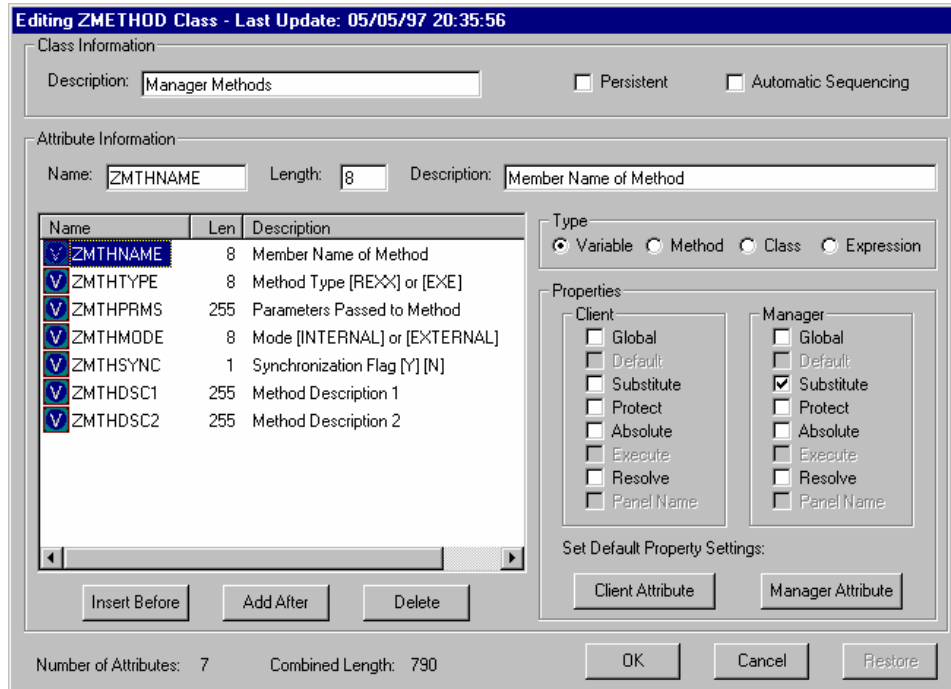
instance, which looks like this:





The EDM Manager executes the EDMPPRO manager method, passing “ZMASTER” as a parameter. This causes the contents of the ZMASTER object to be written to the Profile file of the EDM Database.

The variables in the object instantiated from the ZSYSTEM.ZMETHOD.PUTPROF\_ZMASTER instance are not defined in global memory by the resolution process, because the ZSYSTEM.ZMETHOD class is not a persistent class, and each of the variables lacks the Manager Global property:



Resolution terminates normally when there are no more attributes to process. After completing the processing of all attributes in the ZSYSTEM.ZPROCESS.ZMASTER instance, resolution terminates.



# INDEX

---

## A

- about EDM System Explorer ..... 6
- adding a bitmap image to a radio button and check box sampler screen..... 184
  - assigning settings..... 184
- adding a new OODB class ..... 29
- adding a new OODB instance ..... 39
- adding an OODB domain to a file ..... 26
- adding heaps to an EDM object ..... 116
- adding the cancel button to a push button and list sampler screen ..... 152
  - assigning properties ..... 153
- adding the change trace settings check box to a radio button and check box sampler screen 181
  - assigning properties ..... 182
- adding the controls check box to a radio button and check box sampler screen..... 183
  - assigning properties ..... 183
- adding the edit box to a push button and list sampler screen ..... 160
  - assigning properties ..... 161
- adding the group box
  - to a push button and list sampler screen 165
- assigning properties..... 166
  - to a radio button and check box sampler screen ..... 180
- assigning properties..... 181
- adding the launch list button to a push button and list sampler screen..... 159
  - assigning properties ..... 160
- adding the launch listbox button to a push button and list sampler screen ..... 156
- adding the launch listbox to a push button and list sampler screen
  - assigning properties ..... 156
- adding the list box to a push button and list sampler screen ..... 154
  - assigning properties ..... 155
- adding the list button to a push button and list sampler screen ..... 157
  - assigning properties ..... 158
- adding the next file button to a push button and list sampler screen..... 162
  - assigning properties ..... 163
- adding the OK button
  - to a push button and list sampler screen 150
- assigning properties..... 151
  - to a radio button and check box sampler screen ..... 173
- assigning properties ..... 174
- adding the prev file button
  - to a push button and list sampler screen 162
- assigning properties ..... 163
  - to a radio button and check box sampler screen ..... 174
- assigning properties ..... 175
- adding the radio button to a radio button and check box sampler screen ..... 178
  - assigning properties..... 178
- adding the show radio panel button
  - to a push button and list sampler screen 153
  - to a screen
- assigning properties ..... 187
- adding the ZMASTER.ZTRACE description text to a radio button and check box sampler screen ..... 175
- adding the ZMASTER.ZTRACE value label to a radio button and check box sampler screen 176
- adding variables to EDM objects ..... 104
- advanced editing options in EDM Publisher 82
  - client behaviors tab ..... 84
  - client management tab..... 82
  - data options tab ..... 84
- advanced tab option in EDM System Explorer ..... 54
- aligning
  - check boxes with graphic in a radio button and check box sampler screen..... 186
  - push buttons with group box in a radio button and check box sampler screen ..... 186
- appearance of an EDM System Explorer window, controlling ..... 10
- applications, packaging
  - deploying to a machine vs. a userID 59
  - editing how to publish ..... 68
  - editing package components ..... 67
  - selecting system components..... 63
  - summary ..... 70
- applications, publishing
  - summary ..... 81
- ASK, EDM extended batch command 259
- assign a title
  - to a new push button and list sampler screen 150
  - to a new radio button and check box sampler screen ..... 171

assigning the ZMASTER.ZTRACE value static text properties to a radio button and check box sampler screen ..... 177

attribute types for OODB classes ..... 32

availability of EDM Client Explorer on EDM Client platforms ..... 88

availability of EDM Dialog scripting commands ..... 322

availability on platforms EDM extended batch commands .... 242

## B

bitmap image adding to a radio button and check box sampler screen..... 184

assigning settings ..... 184

inserting in a screen ..... 223

assigning properties..... 223

BREAK, EDM dialog command..... 332

build sample object EDM Screen Painter ..... 146

built-in functions of resolution process403

buttons that launch other screens EDM Screen Painter ..... 153

## C

CALC, EDM dialog command ..... 333

CALL, EDM extended batch command260

cancel button adding to a push button and list sampler screen ..... 152

assigning properties..... 153

CD, EDM extended batch command. 261

change trace settings check box adding to a radio button and check box sampler screen..... 181

assigning properties..... 182

changing a screen title ..... 205

changing attribute types for OODB classes 32

changing fonts in a screen..... 208

changing view of OODB in EDM System Explorer ..... 12

changing your drive/ directory path. 126

CHDIR, EDM extended batch command261

check boxes aligning with graphic in a radio button and check box sampler screen..... 186

inserting in a screen ..... 214

assigning properties..... 215

client behaviors tab advanced editing options in EDM Publisher 84

client management tab advanced editing options in EDM Publisher 82

CLOCK, EDM dialog command..... 334

closing screen without saving changes204

command line modifiers EDM extended batch commands.....248

common push buttons EDM Screen Painter ..... 146

compare EDM screen objects and regular EDM objects ..... 139

comparison chart for editing objects created on different platforms..... 91

continuity between calls for dialog commands.....392

controlling appearance of an EDM System Explorer window ..... 10

controlling features of EDM System Explorer ..... 49

controlling screen sequencing..... 189

controlling the user window ..... 245

controls changing the font of on a screen ....208

making them the same height in EDM Screen Painter.....231

controls check box adding to a radio button and check box sampler screen ..... 183

assigning properties ..... 183

controls toolbar EDM Screen Painter ..... 141

controls, making them the same width in EDM Screen Painter ..... 233

conventions to follow when creating dialog files ..... 324

COPY, EDM extended batch command262

copying an OODB class..... 27

copying an OODB instance ..... 38

create a new push button and list sampler screen using EDM Screen Painter ..... 149

create a new radio button and check box sampler screen using EDM Screen Painter..... 171

create sequenced screens ..... 190

creating a push button and list sampler screen adding the cancel button..... 152

adding the edit box..... 160

adding the group boxes ..... 165

adding the launch list button ..... 159

adding the launch listbox button ..... 156

adding the list box ..... 154

adding the list button ..... 157

adding the next file button ..... 162

adding the OK button ..... 150

adding the prev file button ..... 162

adding the show radio panel button. 153

assigning a title ..... 150

testing the screen.....	167
creating a radio button and check box sampler screen	
adding a bitmap image.....	184
adding the change trace settings check box	181
adding the controls check box.....	183
adding the group box .....	180
adding the OK button .....	173
adding the prev file button.....	174
adding the radio button .....	178
adding the ZMASTER.ZTRACE description text .....	175
adding the ZMASTER.ZTRACE value label	176
aligning check boxes with graphic...	186
aligning push buttons with group box	186
assigning a title.....	171
assigning the ZMASTER.ZTRACE value static text properties .....	177
testing the screen.....	187
creating an EDM object .....	92

## D

data options tab	
advanced editing options in EDM Publisher	84
default values, global, in EDM Publisher	85
defining	
EDM Packager vs EDM Publisher ..	57
defining the user window.....	244
definition of object information in screen entries .....	100
DEL, EDM extended batch command	280
DEL_VAR, EDM extended batch command	265
DELAY, EDM dialog command.....	336
DELAY, EDM extended batch command	264
deleting an OODB class.....	37
deleting an OODB instance.....	41
deleting heaps from an EDM object ..	120
deleting variables from EDM objects	107
deploying applications to a machine vs. a userID .....	59
DESTROY_IND, EDM extended batch command .....	266
DESTROY_USER_WINDOW, EDM extended batch command .....	267
determining problems	
scenario.....	129
DEVICE, EDM dialog command .....	337
dialog commands	
continuity between calls .....	392
numeric expressions .....	395
valid operators specified from high to low priority .....	396
valid SEND keywords.....	398
dialog files	
conventions to follow when creating	324

dialog syntax	
understanding .....	324
dialog variables .....	325
common variables.....	326
environment variables.....	327
predefined variables.....	327
prompted variables .....	325
DISABLE_MENU_EXIT, EDM extended batch command .....	268
display information screen for EDM objects	99
display variable names without resolving symbolic substitution for the screen .	236
DISPLAY_USER_WINDOW, EDM extended batch command.....	269
distribution models for EDM .....	2
DLGBOX, EDM dialog command .....	339
drag-and-drop connection	
making.....	43
drive/directory path	
changing .....	126
DROP_OBJECT, EDM extended batch command .....	270
duplicate heaps in an EDM object .....	118

## E

ECHO, EDM extended batch command	271
edit box	
adding to a push button and list sampler screen .....	160
assigning properties .....	161
inserting in a screen.....	209
assigning properties .....	210
editing an OODB class .....	30
editing an OODB instance .....	39
editing how to publish package when packaging an application.....	68
editing object variables .....	98
editing objects created on different platforms	
comparison chart .....	91
editing options, advanced, in EDM Publisher .....	82
client behaviors tab .....	84
client management tab.....	82
data options tab .....	84
editing package components when packaging an application .....	67
editing variables associated with EDM objects .....	108
EDM	
standard mouse actions.....	7
EDM Client	
local objects .....	128
objects normally present on desktop	128
EDM Client Explorer .....	87

availability on EDM Client platforms	88	availability of	322
object menu map reference	131	EDM distribution models	2
path menu map reference	132	EDM Explorer	1-54
variable view menu reference	133	EDM extended batch command	
heap menu map	135	ASK	259
object menu map	136	CALL	260
variable menu map	134	CD	261
EDM dialog command		CHDIR	261
BREAK	332	COPY	262
CALC	333	DEL	280
CLOCK	334	DEL_VAR	265
DELAY	336	DELAY	264
DEVICE	337	DESTROY_IND	266
DLGBOX	339	DESTROY_USER_WINDOW	267
ELSE	341	DISABLE_MENU_EXIT	268
END	342	DISPLAY_USER_WINDOW	269
ENDIF	343	DROP_OBJECT	270
EXECUTE	344	ECHO	271
EXECUTE (p)	344	EDM_ADD_INST	272
FLICKER	346	EDM_COPY	273
GOSUB	347	EDM_DEL_INST	274
GOTO	348	EDM_DELETE	275
HOST	349	EDM_ERASE	276
IF 350		EDM_MOVE	277
INPUT	355	EDM_SET_INST	278
KEYDELAY	356	ENABLE_MENU_EXIT	279
LINE	357	ERASE	280
MANUAL	359	EXIST	281
NOBREAK	360	MD	282
OFF	361	MKDIR	282
OIA	362	MOVE	283
ON BREAK GOTO	363	RD	288
ON EVENT GOTO	364	reference	258
ON PROG GOTO	367	RELEASE	284
ON RUNOUT GOTO	369	REM	285
ON TIMEOUT GOTO	370	REN	287
OPTION	371	RENAME	286
PAUSE	372	RMDIR	288
PRINT	373	SET_ABORT_TITLE	289
PROMPT	375	SET_ABORT_WINDOW	290
QUIT	377	SET_CANCEL_TITLE	292
REMARK	379	SET_CANCEL_WINDOW	293
RETURN	380	SET_CONTINUE_TITLE	295
RUN	381	SET_CONTINUE_WINDOW	296
RUN(p)	381	SET_DECRYPT_OFF	299
RUNOUT	383	SET_DECRYPT_ON	298
SEND	385	SET_ECHO_TITLE	300
SET	387	SET_ECHO_WINDOW	301
TIMEOUT	388	SET_FRONT_WINDOW	303
WAITFOR	389	SET_IND_TITLE	304
EDM Dialog Command reference	321-400	SET_PROCESS_NAME	305
EDM dialog commands		SET_USER_BACKGROUND	306
quick reference	328	SET_USER_FOREGROUND	307
specifying	331	SET_USER_TITLE	308
EDM dialog files		SET_USER_WINDOW	309
samples	399	SET_VAR	311
EDM Dialog scripting commands		SETIND	312
		SINGLE_STEP_OFF	314

- SINGLE\_STEP\_ON ..... 313
- STARTIND ..... 315
- TRAP ..... 317
- WRITE\_USER\_WINDOW ..... 318
- EDM extended batch commands 241–319
  - availability on platforms ..... 242
  - command line modifiers ..... 248
  - using symbolic notation ..... 245
- EDM extended batch quick reference 255
- EDM Manager selection editor screen
  - control properties ..... 194
- EDM Manager selection program..... 191
- EDM Manager selection screen
  - control properties ..... 193
- EDM objects
  - adding heaps to ..... 116
  - adding variables ..... 104
  - creating ..... 92
  - deleting heaps from ..... 120
  - deleting variables ..... 107
  - display information screen for ..... 99
  - duplicate heaps in ..... 118
  - editing variables ..... 108
  - filtering ..... 110
  - finding heaps in ..... 122
  - from remote EDM desktops
  - working with in EDM Client Explorer 128
    - increase size of heap ..... 123
    - opening ..... 92
    - saving ..... 103
    - sort the heaps of ..... 101
    - viewing in object list box ..... 93
    - opening ..... 60
    - overview ..... 58
    - packaging an application ..... 63
    - phases defined ..... 61
- EDM Packager vs EDM Publisher
  - tool definitions ..... 57
- EDM Publisher ..... 71–85
  - advanced editing options in ..... 82
- client behaviors tab ..... 84
- client management tab ..... 82
- data options tab ..... 84
  - global default values in ..... 85
  - overview ..... 74
- EDM screen engines
  - defined for techniques 1 and 2 ..... 189
- EDM screen objects and regular EDM objects comparison ..... 139
- EDM Screen Painter
  - build sample object ..... 146
  - buttons that launch other screens ... 153
  - common push buttons ..... 146
  - controlling screen sequencing ..... 189
  - controls toolbar ..... 141
  - create a new push button and list sampler screen ..... 149
  - create a new radio button and check box sampler screen ..... 171
  - horizontally aligning controls ..... 225
  - inserting a bitmap image ..... 223
  - inserting a check box ..... 214
  - inserting a group box ..... 218
  - inserting a list box ..... 219
  - inserting a list button ..... 221
  - inserting a radio button ..... 216
  - internal variables ..... 238
  - for a PANEL instance ..... 238
  - valid ZPCTYPE values ..... 238
    - making controls the same height ..... 231
    - making controls the same width ..... 233
    - opening ..... 200
    - opening a new screen ..... 202
    - opening an EDM screen object ..... 201
    - properties window ..... 143
    - push button and list sampler screen ..... 145
    - radio button and check box sampler screen ..... 169
    - reference ..... 199
    - re-sizing push buttons ..... 164
    - saving a screen ..... 203
    - screen and EDM object resolution ... 138
    - screen creation tutorial ..... 144
    - selecting controls ..... 224
    - testing a new screen ..... 235
    - troubleshooting ..... 240
    - vertically aligning controls ..... 228
    - z-named variable reference ..... 237
- EDM Screen Painter screen
  - defined ..... 140
- EDM System Explorer
  - about ..... 6
  - changing view of OODB ..... 12
  - control features of ..... 49
  - controlling window appearance ..... 10
  - filtering view of OODB ..... 17
- filter specifications ..... 18
  - open ..... 8
  - options ..... 49
- advanced tab option ..... 54
- general tab option ..... 50
- instance options tab ..... 51
- EDM terminology ..... 4
- EDM\_ADD\_INST, EDM extended batch command ..... 272
- EDM\_COPY, EDM extended batch command ..... 273
- EDM\_DEL\_INST, EDM extended batch command ..... 274
- EDM\_DELETE, EDM extended batch command ..... 275
- EDM\_ERASE, EDM extended batch command ..... 276

- EDM\_MOVE, EDM extended batch command ..... 277
- EDM\_SET\_INST, EDM extended batch command ..... 278
- EDMCONS
  - extended batch driver for DOS..... 249
- EDMCONSW
  - extended batch driver for windows and OS/2 250
- EDMCONSX
  - extended batch driver for Macintosh and Unix 251
- ELSE, EDM dialog command ..... 341
- ENABLE\_MENU\_EXIT, EDM extended batch command..... 279
- END, EDM dialog command ..... 342
- ENDIF, EDM dialog command..... 343
- ERASE, EDM extended batch command280
- examples of valid filter expressions .... 21
- EXECUTE (p), EDM dialog command344
- EXECUTE, EDM dialog command .... 344
- EXIST, EDM extended batch command281
- explore EDM Explorer..... 1-54
- extended batch driver for DOS
  - EDMCONS ..... 249
- extended batch driver for Macintosh and Unix
  - EDMCONSX..... 251
- extended batch driver for windows and OS/2
  - EDMCONSW..... 250
- extended batch files
  - running ..... 248
- extended batch graphical user interface definition ..... 243
- extended batch usage notes ..... 254
- F**
- filtering EDM objects ..... 110
- filtering view of OODB in EDM System Explorer ..... 17
  - filter specifications..... 18
- finding heaps in an EDM object..... 122
- FLICKER, EDM dialog command..... 346
- fonts, changing in a screen..... 208
- G**
- general tab option in EDM System Explorer 50
- global default values in EDM Publisher85
- go to specific heap of an EDM object 122
- GOSUB, EDM dialog command..... 347
- GOTO, EDM dialog command..... 348
- group box
  - adding to a push button and list sampler screen ..... 165
  - assigning properties..... 166
  - adding to a radio button and check box sampler screen ..... 180
- assigning properties ..... 181
  - inserting in a screen.....218
- assigning properties ..... 218
- H**
- heaps
  - adding to EDM objects.....116
  - deleting from EDM objects.....120
  - duplicating in EDM objects .....118
  - finding in EDM objects .....122
  - increase size of .....123
  - of selected EDM object
- sort ..... 101
- horizontally aligning controls in
  - EDM Screen Painter .....225
- HOST, EDM dialog command ..... 349
- I**
- IF, EDM dialog command ..... 350
- increase size of a heap ..... 123
- information screen for EDM objects
  - display.....99
- information properties, setting for OODB classes ..... 31
- INPUT, EDM dialog command ..... 355
- inserting a bitmap iamge
  - EDM Screen Painter
  - assigning properties ..... 223
- inserting a bitmap image
  - EDM Screen Painter .....223
- inserting a check box
  - EDM Screen Painter .....214
  - assigning properties ..... 215
- inserting a group box
  - EDM Screen Painter .....218
  - assigning properties ..... 218
- inserting a list box
  - EDM Screen Painter .....219
  - assigning properties ..... 220
- inserting a list button
  - EDM Screen Painter .....221
  - assigning properties ..... 222
- inserting a push button in a screen ... 211
  - assigning properties.....211
  - description of push button actions ...212
- inserting a radio button
  - EDM Screen Painter .....216
  - assigning properties ..... 216
- inserting an edit box in a screen ..... 209
  - assigning properties.....210
- inserting static text in a screen ..... 206
  - assigning properties.....206



- installed EDM extended batch programs 319
- instance options tab in EDM System Explorer ..... 51
- internal variables in
  - EDM Screen Painter ..... 238
- for a PANEL instance ..... 238
- valid ZPCTYPE values ..... 238
- K**
- KEYDELAY, EDM dialog command. 356
- L**
- launch list button
  - adding to a push button and list sampler screen ..... 159
- assigning properties..... 160
- launch listbox button
  - adding to a push button and list sampler screen ..... 156
- assigning properties..... 156
- launching a file
  - push button and list sampler screen 146
- LINE, EDM dialog command..... 357
- list box
  - adding to a push button and list sampler screen ..... 154
- assigning properties..... 155
- inserting in a screen ..... 219
- assigning properties..... 220
- list button
  - adding to a push button and list sampler screen ..... 157
- assigning properties..... 158
- inserting in a screen ..... 221
- assigning properties..... 222
- M**
- making a drag-and-drop connection... 43
- making controls the same height in
  - EDM Screen Painter ..... 231
- making controls the same width in
  - EDM Screen Painter ..... 233
- manipulating OODB components ..... 24
- manipulating view of OODB in EDM System Explorer ..... 12
- MANUAL, EDM dialog command... 359
- MD, EDM extended batch command 282
- MKDIR, EDM extended batch command 282
- mouse, standard actions in EDM..... 7
- MOVE, EDM extended batch command 283
- N**
- new OODB class
  - adding..... 29
- new OODB instance
  - adding ..... 39
- next file button
  - adding to a push button and list sampler screen ..... 162
- assigning properties ..... 163
- NOBREAK, EDM dialog command .. 360
- numeric expressions
  - in dialog commands ..... 395
- O**
- object information screen entries
  - defined ..... 100
- object list box
  - viewing EDM objects in..... 93
- object menu map reference in EDM Client Explorer ..... 131
- object variables
  - searching on ..... 112
  - viewing and editing ..... 98
- objects created on different platforms
  - comparison chart ..... 91
- OFF, EDM dialog command ..... 361
- OIA, EDM dialog command ..... 362
- OK button
  - add to a push button and list sampler screen
- assigning properties ..... 151
- adding to a push button and list sampler screen ..... 150
- adding to a radio button and check box sampler screen ..... 173
- assigning properties ..... 174
- ON BREAK GOTO, EDM dialog command 363
- ON EVENT GOTO, EDM dialog command 364
- ON PROG GOTO, EDM dialog command 367
- ON RUNOUT GOTO, EDM dialog command ..... 369
- ON TIMEOUT GOTO, EDM dialog command ..... 370
- OODB
  - changing view of in EDM System Explorer 12
  - filtering view of in EDM System Explorer 17
- filter specifications ..... 18
- OODB class
  - adding new..... 29
  - attribute types ..... 32
  - changing attribute types..... 32
  - copying..... 27
  - deleting ..... 37
  - editing..... 30
  - setting information properties ..... 31
- OODB components
  - manipulating..... 24
- OODB domain
  - adding to a file..... 26

- OODB instance
  - adding new ..... 39
  - copying ..... 38
  - deleting ..... 41
  - editing ..... 39
  - renaming ..... 41
- opening a new screen in EDM Screen Painter ..... 202
- opening an EDM object ..... 92
- opening an EDM screen object ..... 201
- opening EDM Packager
  - using EDM Administrator ..... 60
- opening EDM System Explorer ..... 8
- opening the EDM Screen Painter ..... 200
- OPTION, EDM dialog command ..... 371
- overview
  - EDM Packager ..... 58
  - EDM Publisher ..... 74
- P**
- packaging applications
  - deploying to a machine vs. a userID 59
  - editing how to publish ..... 68
  - editing package components ..... 67
  - selecting system components ..... 63
  - summary ..... 70
  - using EDM Packager ..... 63
- path menu map in EDM Client Explorer 132
- PAUSE, EDM dialog command ..... 372
- phases in EDM Packager, defined ..... 61
- predefined variables
  - complex
  - specifications ..... 393
  - simple
  - specifications ..... 393
- prev button
  - adding to a radio button and check box sampler screen ..... 174
- assigning properties ..... 175
- prev file button
  - adding to a push button and list sampler screen ..... 162
- assigning properties ..... 163
- PRINT, EDM dialog command ..... 373
- problem determination scenario ..... 129
- process, resolution
  - built-in functions ..... 403
- PROMPT, EDM dialog command ..... 375
- properties window
  - EDM Screen Painter ..... 143
- publishing applications
  - summary ..... 81
- push button and list sampler screen.. 145
  - create ..... 149
  - launching a file ..... 146
  - selecting a file ..... 145
- push buttons
  - aligning with group box in a radio button and check box sampler screen ..... 186
  - common
    - adding in EDM Screen Painter ..... 146
    - inserting in a screen ..... 211
  - assigning properties ..... 211
  - description of push button actions .... 212
- Q**
- quick reference
  - EDM extended batch commands ..... 255
  - for EDM dialog commands ..... 328
- QUIT, EDM dialog command ..... 377
- R**
- radio button
  - adding to a radio button and check box sampler screen ..... 178
- assigning properties ..... 178
- inserting in a screen ..... 216
- assigning properties ..... 216
- radio button and check box sampler screen
  - create ..... 171
  - EDM Screen Painter ..... 169
- RD, EDM extended batch command. 288
- reference
  - EDM Dialog Command ..... 321–400
- RELEASE, EDM extended batch command 284
- REM, EDM extended batch command 285
- REMARK, EDM dialog command .... 379
- REN, EDM extended batch command 287
- RENAME, EDM extended batch command 286
- renaming an OODB instance ..... 41
- re-sizing push buttons on a screen .... 164
- resolution process
  - built-in functions ..... 403
- resolve value of a variable that uses symbolic substitution ..... 124
- RETURN, EDM dialog command ..... 380
- RMDIR, EDM extended batch command 288
- RUN(p), EDM dialog command ..... 381
- RUN, EDM dialog command ..... 381
- running extended batch files ..... 248
- RUNOUT, EDM dialog command .... 383
- S**
- sample EDM dialog files ..... 399
- sample object, build
  - in EDM Screen Painter ..... 146
- saving a screen in EDM Screen Painter 203

- saving EDM objects..... 103
- scenario for determining problems..... 129
- screen and EDM object resolution
  - EDM Screen Painter ..... 138
- screen creation tutorial
  - EDM Screen Painter ..... 144
- screen sequencing
  - controlling..... 189
- screen sequencing technique 2
  - defined..... 190
- searching on an object variable ..... 112
- selecting a file
  - push button and list sampler screen 145
- selecting controls in
  - EDM Screen Painter ..... 224
- selecting system components when packaging an application..... 63
- SEND keywords
  - for dialog commands..... 398
- SEND, EDM dialog command ..... 385
- SET, EDM dialog command ..... 387
- SET\_ABORT\_TITLE, EDM extended batch command..... 289
- SET\_ABORT\_WINDOW, EDM extended batch command..... 290
- SET\_CANCEL\_TITLE, EDM extended batch command..... 292
- SET\_CANCEL\_WINDOW, EDM extended batch command..... 293
- SET\_CONTINUE\_TITLE, EDM extended batch command..... 295
- SET\_CONTINUE\_WINDOW, EDM extended batch command ..... 296
- SET\_DECRYPT\_OFF, EDM extended batch command..... 299
- SET\_DECRYPT\_ON, EDM extended batch command..... 298
- SET\_ECHO\_TITLE, EDM extended batch command..... 300
- SET\_ECHO\_WINDOW, EDM extended batch command..... 301
- SET\_FRONT\_WINDOW, EDM extended batch command..... 303
- SET\_IND\_TITLE, EDM extended batch command ..... 304
- SET\_PROCESS\_NAME, EDM extended batch command..... 305
- SET\_USER\_BACKGROUND, EDM extended batch command ..... 306
- SET\_USER\_FOREGROUND, EDM extended batch command..... 307
- SET\_USER\_TITLE, EDM extended batch command ..... 308
- SET\_USER\_WINDOW, EDM extended batch command ..... 309
- SET\_VAR, EDM extended batch command 311
- SETIND, EDM extended batch command 312
- setting OODB class information properties 31
- show radio panel button
  - adding to a push button and list sampler screen ..... 153
  - adding to a screen
    - assigning properties ..... 187
- SINGLE\_STEP\_OFF, EDM extended batch command ..... 314
- SINGLE\_STEP\_ON, EDM extended batch command ..... 313
- sort the heaps of selected EDM object101
- specifications of complex predefined variables ..... 393
- specifications of simple predefined variables ..... 393
- specifying EDM dialog commands ... 331
- standard mouse actions in EDM..... 7
- STARTIND, EDM extended batch command ..... 315
- static text
  - inserting into a screen.....206
  - assigning properties ..... 206
- summary of packaging an application 70
- summary of publishing applications .. 81
- symbolic substitution
  - resolve value of a variable that uses124
- T**
- technique 1
  - in controlling screen sequencing .... 189
- technique 2
  - in controlling screen sequencing .... 189
- terminology in EDM..... 4
- testing a newly created push button and list sampler screen..... 167
- testing a newly created radio button and check box sampler screen ..... 187
- testing a screen ..... 235
- TIMEOUT, EDM dialog command ... 388
- title
  - assign to a new radio button and check box sampler screen ..... 171
  - assign to a push button and list sampler screen ..... 150
  - changing in a screen.....205
- tour EDM Explorer ..... 1-54
- TRAP, EDM extended batch command317

- troubleshooting
  - EDM Screen Painter ..... 240
- U**
- understanding dialog syntax..... 324
- use of variables ..... 328
- user window
  - controlling ..... 245
  - definition ..... 244
- using
  - EDM extended batch commands .... 241–319
- using a mouse in EDM
  - standard mouse actions ..... 7
- using EDM Publisher..... 71–85
- using installed EDM extended batch programs
  - ..... 319
- using symbolic notation
  - EDM extended batch commands .... 245
- V**
- valid filter expressions
  - examples ..... 21
- valid operators for dialog commands
  - specified from high to low priority.... 396
- valid SEND keywords
  - for dialog commands ..... 398
- variable names
  - display without resolving symbolic substitution for the
    - screen..... 236
- variable usage ..... 328
- variable view menu reference in EDM Client Explorer ..... 133
- variable, resolve value if uses
  - symbolic substitution ..... 124
- variables
  - adding to EDM objects ..... 104
  - deleting from EDM objects ..... 107
  - editing for EDM objects ..... 108
- vertically aligning controls in
  - EDM Screen Painter ..... 228
- viewing EDM objects in object list box 93
- viewing object variables..... 98
- viewing objects on a different drive/ directory path ..... 126
- viewing portions of OODB in EDM System Explorer ..... 17
  - filter specifications ..... 18
- W**
- WAITFOR, EDM dialog command ... 389
- working with local objects ..... 127
- working with objects from remote EDM desktops
  - ..... 128
- working with OODB classes ..... 27
- working with OODB instances ..... 38
- WRITE\_USER\_WINDOW, EDM extended batch command ..... 318
- Z**
- ZMASTER.ZTRACE description text
  - adding to a radio button and check box sampler
    - screen ..... 175
- ZMASTER.ZTRACE value label
  - adding to a radio button and check box sampler
    - screen ..... 176
- ZMASTER.ZTRACE value static text properties
  - assigning to a radio button and check box sampler
    - screen ..... 177
- z-named variable reference in
  - EDM Screen Painter ..... 237

