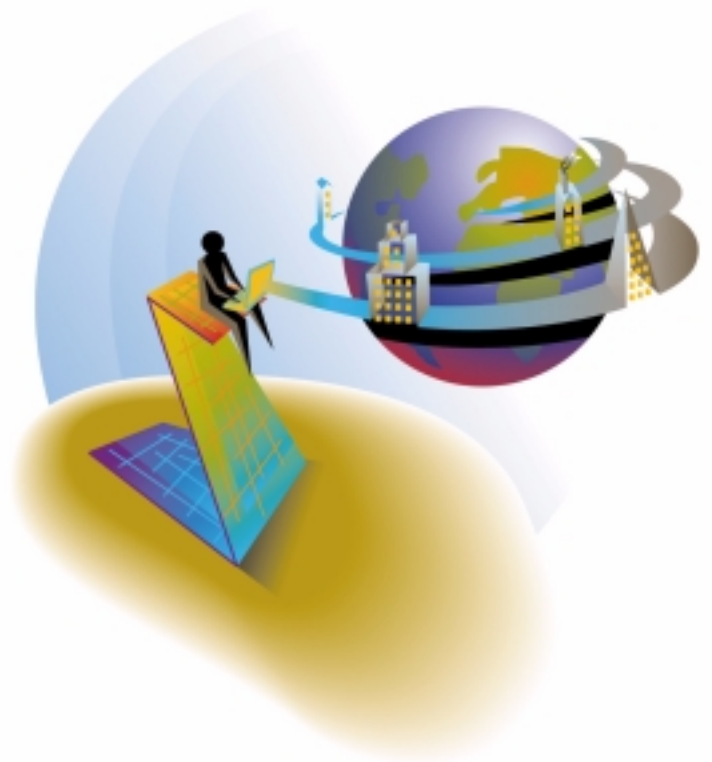


eDeveloper V9.4



What's New in eDeveloper Version 9.4

The information in this manual is subject to change without prior notice and does not represent a commitment on the part of MSE.

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

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

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

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

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

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

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

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

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

Btrieve® is a registered trademark of Pervasive Software, Inc.

Pervasive.SQL™ is a registered trademark of Pervasive Software, Inc.

C-ISAM™ is a trademark of INFORMIX.

PC/TCP® Network Software is a registered trademark of FTP Software Inc.

IBM®, Topview™, iSeries™, pSeries™, xSeries™, and RISC System/6000™ are trademarks of International Business Machines Corporation.

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

Oracle® is a registered trademark of the Oracle Corporation.

UNIX® is a registered trademark of UNIX System Laboratories.

InterSystems™, and Caché™ are trademarks of InterSystems Corporation.

GLOBEtrotter and FLEXlm are registered trademarks of GLOBEtrotter Software, Inc.

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

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

Version 9.40 2003

03 02 01 00 6 5 4 3 2 1

4190-06000

Copyright © 2003 by Magic Software Enterprises Ltd. All rights reserved.

Contents

1 Overview

Interactive Web Application Paradigm	9
eDeveloper Components.....	10
Event-Driven Engine	10
Data Management	10
Error Handling	11
Multi-Request Processing	11
GUI Enhancements	11
Productivity Enhancements	11
Toolkit Enhancements	12
eDeveloper Connectivity	12

2 Interactive Web Application Paradigm

Browser Client	14
Server Behavior	14
Context Management	15
Tools for Creating Interactive Web Applications.....	16
Classes and Styles Support	16
Functions for Interactive Web Applications	17
CALLJS.....	17
CALLOBJ	17
CALLURL	17
CALLPROGURL	17

3 eDeveloper Components

eDeveloper Component Integration.....	19
What's New in Magic Version 9.4	

What is a Component?	19
Creating the eDeveloper Component Interface	19

4 Event-Driven Engine

Event Terminology	21
Event Types	21
Event Handling	22
Control Level Handlers	22
User-Defined Events	22
User-Defined Events	23
Handlers Repository	23
Raise Event Command	23
Propagation	23
Handler Search Mechanisms	23
Component Events	24

5 Improved Data Management

Deferred Transactions.....	26
Transaction Mode Property	26
Transaction Cache Behavior	27
Handling Deferred Mode Flush Errors	28
Update and Delete Statements	28
Numeric Fields Update	29
DM Statements Invocation Order.....	30
Resource Locking	31
Lock.....	31
Unlocked	31
The Resource Lock File	31
SQL Range.....	31

6 Application Error Handling

Error Handling Mechanism	35
--------------------------------	----

Error Behavior Strategies	35
Abort Strategy	36
Recover Strategy	36
Error Handlers - Toolkit Definition.....	36
Error Handler Properties	36
Runtime Error Handling	38
Range and Locate a Task According to a Record's Position.....	38

7 Multi-Threaded Engine

Environment Settings.....	39
Maximum Number of Concurrent Requests	39
Current Application.....	39
INIPUT and INIGET.....	40
Security	40
TCP/IP Ports.....	40
Broker and the Generic Messaging Layer.....	40
Main Programs	40
Transactions	40
Locking	40
Memory Tables	41
Resident Tables	41
Shared Resources	41
Runtime Expressions	41

8 GUI Enhancements

Table Control	43
Multi-Marking Functionality	43
Multi-Marking Event Handlers	44
Multi-Marking Functions	45
Table Control Placement.....	45
.....	46

Tree Control.....	47
Drag and Drop Support.....	47

9 Productivity Enhancements

Main Program	48
Main Program Runtime Behavior	48
Main Program Toolkit Behavior	49
Main Program and Components.....	50
Main Program Functions	50
Data Control.....	51
Combo and List Box Control Properties	51
Models	52
Model Repository	52
Working with Models	54
Deploying an eDeveloper Flat File.....	56
Select Flat eDeveloper File Deployment for an Application	56
Save an eDeveloper Application as an MFF	57
Task Return Value	57
Left Outer Join for ISAM databases.....	57

10 Toolkit Enhancements

Argument Matching.....	59
Select Parameter	59
Parameters Table	59
Arguments List.....	59
Calling a Program with Selected Parameters Defined	60
Calling a Program With No Select Parameters Defined	60
Block Operation.....	60
If-Else	60
Loop	61
Vector Support.....	61

eDeveloper Wizard	61
Enhanced Checker	62
Comments	63
Interface Navigation	65
Repositories	66
Task	67
Cross-Reference	68
Bookmarks	68
Find and Replace	69
Working with the ANSI Engine	69
Changes in the Environment Settings	70
Functions	71
The OEM_ANSI.EXE Utility	72
Export/Import	72
Effect on Previous-version Applications	72
Referential Integrity	73
Table Repository	73
Automated Program Generator (APG)	74
Link Condition	75
Toolkit	75
User Interface	76

11 Connectivity to Other Environments

LDAP Support	77
Java Integration	78
Java Functions	78
Java Access Component Generator	78
J2EE Integration	78
COM Integration	78
eDeveloper COM Client	79

eDeveloper COM Interface	79
Full XML Support	79
XML Functions	79
XML Access Component Generator	79
Full Web Service Support.....	80
Using Web Services	81
Web Service Provider	81
Simple Network Management Protocol.....	81
Message Queueing Support	81
Connectivity Functions.....	82
Email Functions	82
Clipboard Functions	82
HTTP Functions	82

Welcome to eDeveloper, which has been designed to develop integrated enterprise-wide business applications using an enhanced Interactive Web Application paradigm. eDeveloper offers open extensive support for the industry's leading Internet Web browsers, Web servers, enterprise servers, languages, and databases.

This book highlights the enhanced features that have been added to eDeveloper.

Interactive Web Application Paradigm

The eDeveloper Platform lets you create browser-based applications where the browser is used as the client. The client side of the browser-based deployment paradigm provides all the runtime logic needed to maintain HTML controls with their associated data, such as trapping and handling events. The client has only minimal communication with the server.

The server side of the browser-based deployment paradigm consists of the following elements:

- A new type of program to handle Web processing.
- A new concept of defining relationships between eDeveloper tasks.
- Concurrently active programs.
- A mechanism that enables a eDeveloper program to continue running between requests.

eDeveloper Components

eDeveloper lets you define application objects as components that can be shared with other eDeveloper-based applications. Exporting components lets you share resources among eDeveloper-based applications.

The Component Interface Builder utility lets you build a Component Interface file from the objects displayed in the application development repositories.

In previous eDeveloper-based versions, you could not share application objects with other eDeveloper-based applications.

Event-Driven Engine

eDeveloper has created event handlers to implement different types of events. You are no longer limited by the application task flow.

Data Management

Data management lets you organize the data as it is stored in the physical database. The following data management features are available in eDeveloper:

- Deferred Transactions
- Referential Integrity
- Link Condition
- eDeveloper Where Clause

Error Handling

The error handling feature lets you overwrite eDeveloper's default behavior for handling errors while the application is running. Developers can write handlers to specify responses to the different errors that may arise during Runtime. eDeveloper provides a list of known and expected errors that can be handled, and also lets you handle other errors specific to their database management system.

Multi-Request Processing

In eDeveloper, the eDeveloper background application server can process multiple requests at the same time. Each thread runs on a different runtime context and does not interact with other threads.

Online enterprise servers in both Toolkit and Runtime remain single-threaded.

GUI Enhancements

A multi-tabbed navigation pane, enhanced table control, menu item images, and comments are all part of the dynamic, easy-to-use interface.

Productivity Enhancements

The following Productivity enhancements are included in eDeveloper:

- Cross Referencing
- Main Program
- Data Controls
- Models
- Flat eDeveloper Application File (MFF)

Toolkit Enhancements

The following Toolkit enhancements have been added to eDeveloper Version 9.

- ANSI-based Engine
- Select Parameter
- Block If-Else, Block Loop
- Vector Support
- eDeveloper Wizard
- Task Return Value
- Enhanced Checker

eDeveloper Connectivity

eDeveloper V9.4 has greatly enhanced its connectivity to other environments, such as:

- LDAP Support
- Java Integration
- COM Support
- XML Support
- Web Service Support
- SNMP Support
- Message Queueing Support
- Mail, Clipboard and HTTP functions

Interactive Web Application Paradigm

2

eDeveloper offers a comprehensive solution for creating interactive Web applications.



This chapter includes the following topics:

- Browser Client
- Server Behavior
- Tools for Creating Interactive Web Applications
- Functions for Developing Interactive Web Applications

Browser Client

The browser is a very thin client that minimizes the need for client maintenance. The browser client provide by the eDeveloper can:

- Trap events that occur on the page and handle them.
- Re-compute and refresh data and the appearance of the interface elements.
- Execute any eDeveloper function that can be run locally. For example, all number, string, date, and time manipulation functions can be evaluated locally.
- Update the dataview.
- Support any field level validation definition as the online client of eDeveloper does.
- Maintain a local cache of data greater than what it displays, which lets you scroll through new data locally. This decreases the number of interactions necessary between the client and server.

All these new client enhancements are transparent to the eDeveloper application developer and makes programming such browser tasks very easy.

Server Behavior

The sever side is the most significant component of the interactive Web application. The server does the following:

- Identifies each remote client. The server keeps the context of the task for each client that opens a defined logical task.
- Establishes the context which is an internal description of the exact location of the client in the task and the manipulated data within the task's transaction.

- Translates the delivered data manipulations from the client and passes it to the databases that take part.
- Re-links a record on the client to retrieve the new linked record.
- Executes any other function that by nature cannot be run locally on the client.

When the server creates the client, it “tells” the client when to run the logic locally, and when to call it to run the logic when using the functions.

Context Management

The *Context Manager*, which intercepts and dispatches all requests, is responsible for maintaining all active contexts and for handling user requests.

The context management mechanism for the interactive Web application paradigm lets the program remain active between requests. The context mechanism includes the state of the Runtime engine, the Task tree, the database cursor Memory tables, and the I/O files.

Each context has its own identification, which is used to identify a context within the deployment environment.

The context is active until the program is closed, or until the *Inactivity Timeout value* expires. You can define the number of seconds an inactive context remains active by setting a value in the *Inactivity Timeout* environment property.



For more information, see the Distributed Application Architectures chapter of The eDeveloper *Reference Guide*.

Tools for Creating Interactive Web Applications

The last part of the interactive web-based solution is the eDeveloper Toolkit. The eDeveloper Toolkit actually enables the operation of the browser-based application.

A new task type of 'Browser' is added to the existing 'Online' and 'Batch' task types. This program type creates a fully functioning online browser client at Runtime.

Since the Browser interface is HTML-based, it provides a transparent integration with a third party Web Authoring tool of your choice. eDeveloper interacts smoothly with all the page elements.

Although the interface is external to eDeveloper, the Toolkit lets you handle any control on the page and makes it data-driven by assigning expressions to its properties.

Classes and Styles Support

When you work with a Browser interface, the way styles are used dictates the look and feel of the HTML page. eDeveloper's new Browser task supports the styles and classes listed below:

- Default Class
- MouseOverClass
- MouseDownClass
- Default Class
- MouseOverClass
- MouseDownClass

Functions for Interactive Web Applications

CALLJS

Calls a java script function that resides on a page.

CALLOBJ

Calls a method of an ActiveX or Java applet object that resides on the page.

CALLURL

Calls a URL through the Evaluate operation in a browser task.

CALLPROGURL

Calls a URL designated by an eDeveloper program that resides on an application server.



For more information, see the Distributed Application Architectures chapter of *The eDeveloper Reference Guide*.

eDeveloper lets you define application objects as a component. Exporting components lets you share resources among eDeveloper-based applications.

This chapter includes the following topics:



- eDeveloper Component Integration
- What is a Component
- Creating the Component Interface

eDeveloper Component Integration

eDeveloper lets you divide your application into components, where each component may contain any object used by eDeveloper.

What is a Component?

A component is another eDeveloper-based application added to your main eDeveloper application during development (Toolkit) mode.

A component is added either by loading the eDeveloper Component Interface. This is a text file (usually with the MCI extension) that describes which of the added applications' objects are available to be used and referred to by the developer.

The objects that can be published through the MCI can be any main object that is supported by eDeveloper: Models, Tables, Programs, Help Screens, Rights, Main program Events, a sub set of environment settings, Logical names, Services, and Database definitions.

Once a component interface is loaded, any published object of the added application can be referred to by the host application as if it were its own.

Creating the eDeveloper Component Interface

Although the eDeveloper Component Interface file is a very simple text file, eDeveloper provides a simple interface to define what objects of the application can be accessed by a hosting application.



For more information, refer to the Components chapter of *The eDeveloper Reference Guide*.

e Developer lets you define eDeveloper logic in response to implicit and explicit events that may occur during the execution of a task.

This chapter includes the following topics:



- Event Terminology
- Event Types
- Interactive Task Event Handling
- Batch Task Event Handling
- Event-Driven Toolkit
- Event-Driven Runtime

Event Terminology

The following are terms for the Event-driven Engine:

- **Event** - An event is the logical definition of a significant occurrence. An event can be handled by an event handler to perform a flow of operations.
- **Trigger** - An event can be assigned as a trigger by another user-defined event. Whenever the trigger event is raised it will also trigger the user-defined event.
- **Handler** - A set of operations designated to be performed when a specific event is raised.

Event Types

Events in eDeveloper are divided into two groups, predefined events and user-defined events.

Predefined events have the following categories:

- Internal eDeveloper events have the following qualities:
 - Events that are defined by the eDeveloper Runtime operation.
 - Events that have handlers hard-coded into the eDeveloper engine.
- System events - Events of various keystroke combinations.

User-defined events have the following categories:

- Timer events - Events that occur on a predefined interval.
- Expression evaluated events - Events that are raised when the expression to which they are defined evaluates to True.
- Application events - Events defined at the application level.

Event Handling

Events that are triggered usually have to be handled by the running application. Some events have default handlers defined by the eDeveloper engine, but you often want to supply your own handlers for a specific event.

In addition to the regular built-in handlers, such as Task Prefix, Task Suffix, Record Prefix, Record Main, Record Suffix, etc., eDeveloper lets you add your own handler for an event.

Control Level Handlers

The Control level has four predefined event handlers: the Control Prefix; the Control Suffix; Control Verification; and the Control Change.

These handlers let you control which operations are executed before entering a control, upon exiting a control, when the value of a control is changed, and during verification of a control.

User-Defined Events

You may define your own events in any task. The Task Event list displays all the user events that are defined in the parent task of the current task, and allows for the modification of user events in the current task.

A user defined event can be referred to from the task in which it is created and its subtasks. Main program events are available to all programs of the application, making them global events.

A user-defined event can be set with a trigger. A trigger is an additional event that will implicitly trigger and raise the user-defined event when this event is raised.

User-Defined Events

You may define your own events in any task. The Task Event list displays all the user events that are defined in the parent task of the current task and lets you modify user events in the current task.

You can refer to a user-defined event from the task where it is created and its subtasks. Main program events are available to all programs of the application, making them global events.

A user-defined event can be set with a trigger, which is an additional event that implicitly triggers and raises the user-defined event when the trigger event is raised.

Handlers Repository

You can define a handler for an event in the Handlers Repository. The Handler Repository is part of a task and displays the Handler flow and virtual variables.

Raise Event Command

A new command lets you raise events during the eDeveloper engine flow. The Raise Event command is handled in the same way as System events.

Propagation

You can propagate an event to an event handler defined in a higher level. When eDeveloper regards the event as not handled, the dispatcher searches for an event handler at the next level.

Handler Search Mechanisms

When a new task is loaded, the eDeveloper Runtime engine searches for an event handler. Each handler is registered by its triggering event and its place in the Task tree. When a task terminates, its handlers are removed from the handler search structure.

When an event is intercepted, the search structure looks for that event. The event search mechanism looks from the last handler registered to the first handler registered.

A disabled event causes the Runtime engine to continue searching the task tree for an event handler at a higher level.

Component Events

A component may export events that can be handled by its calling program.

A component may export event handlers to provide a default handling for an event. The exported event handler should be defined in the component's main program.



For more information, refer to the Application Engine chapter of *The eDeveloper Reference Guide*.

e Developer provides a development environment that allows for greater data integrity, maximal concurrency, and code simplicity.

This chapter includes the following topics:



- Deferred Transactions
- Data Manipulation (DM) Statements Invocation Order
- Resource Locking
- SQL Range

Deferred Transactions

In previous versions, eDeveloper-based applications implemented Data Manipulation (DM) statements (insert/update/delete) after the Record Suffix. This process, called physical transactions, updated the content of the physical database after each DM statement. Processing the changes of records in a dataview for an application in Runtime mode was time-consuming.

Deferred transactions delay the implementation of the DM statements until the time you must commit the changes made to records of a dataview. eDeveloper stores each DM statement in a cache. When you are ready to commit the changes to the database, all changes are implemented concurrently.

Transaction Mode Property

The Transaction Mode property has been added to the Task Control dialog. The Transaction Mode can accept one of the following four values:

Deferred	DM statements are stored in a cache. During the task flow, the DM statements are not sent to the physical database. The statements are only implemented in the database at the expected commit time. All the DM statements accumulated in the cache are implemented at once, and the transaction is closed.
Nested Deferred	Same as Deferred except that when such a task is called from another deferred task, it opens a new deferred transaction nested within the hosting one.
Physical	DM Statements are implemented after the Record Suffix (same as previous versions).
Within Parent	The task is implemented within the parent transaction. A physical transaction that is a parent will have a physical transaction as a child. A parent that is a deferred transaction will have a deferred transaction as a child.

The Break Level Transaction property has been moved to the task property level.

Transaction Mode

When the Transaction mode is set to Deferred, the following action occurs at the different task levels.

Record	All updates at the record level are collected as a group. At record update time, a transaction is opened, the changes are applied, and the transaction is committed.
Task	All updates at the task level are collected as a group. After the Task Suffix, a transaction is opened, the changes are applied, and the transaction is committed.
Group	The updates are collected at the task level, and are updated whenever a specific variable changes its value. When using the group level transaction, you must define the variable on which the group is based.

Locking Strategy Property

Deferred transactions can have the following lock set values:

- None
- On-Modify

SQL Where Statement

The SQL Where statement has been altered to work with tasks that are implemented within a deferred transaction.

Direct SQL

Direct SQL tasks can only be included within a physical transaction.

Transaction Cache Behavior

- The Transaction cache stores all data modifications made during a deferred transaction.
- Each deferred transaction has its own Transaction cache.

- All tasks within the deferred transaction are required to use *full data caching* for all their files.
- Every table has its own Table Update cache. A table is equivalent to an entry in the eDeveloper Table repository. Though two different entries may point to the same table, they will not share the same cache.

Handling Deferred Mode Flush Errors

When implementing a deferred transaction, an error can occur only when the DM statements are actually implemented in the physical database. An error handler defined for this error type and table will be activated. When the error handler starts, only the files or tables that are defined for the handler are visible.

When the error handler is invoked, you can access:

- The cached row value at the time of the error
- Error code
- Error message text

Update and Delete Statements

You may determine the procedure by which eDeveloper creates a WHERE clause for each update or delete statement that is issued. The possibilities are:

- By position only - Only fields that are part of the position will be included.
- By position and updated fields - The original value of the updated fields as well as the position fields.
- By position and selected fields - The original value of all the selected fields as well as the position fields.

A new property called Identify Modified Row has been added to the Table properties. This property may accept any of the above values. Note that this property is only enabled for deferred transaction tasks. The Identify Modified Row property has also been added to the task DB table.

Numeric Fields Update

Previous versions of eDeveloper-based applications allowed only absolute numeric updates. For example, you could only set value X for field FLD1 (FLD1=X).

eDeveloper lets you make differential updates as well. For example, you can update FLD1 by using the value FLD1+X (FLD1=FLD1+X).

The Update Style property has been added to the Field Properties sheet.

The values for the property are:

- Absolute - a fixed value allowed by previous eDeveloper-based applications (FLD1=X)
- Differential- differential values (FLD1=FLD1+X)

This property is only enabled for the Numeric field type that has a normal storage type and relates to an SQL table. This is not relevant for ISAM files.

The Update Style property has also been added to the Select Real operation. Besides the Absolute and Differential values, you can choose the As Table option, which takes the Update Style property from the Table properties.

DM Statements Invocation Order

eDeveloper lets you control the order of DM statement implementation in a deferred transaction. In previous versions, the DM statements implemented according to the Record Suffix order. For example, when a parent task called its child, the DM statements for the child were implemented before the DM statements of the parent, even if changes to the parent task were made prior to calling the child task.

The Sync Data Before Call property has been added to the Call operation. The property has the following values:

- **No** - same as previous versions
- **Yes** - the eDeveloper engine implements changes to the record before executing the Call operation
- Expression - a logical expression that is evaluated as either **Yes** or **No**, and will behave accordingly.

Resource Locking

Resource locking refers to the ability to create an imaginary entity (resource), which can be controlled by only one user at any given moment. A resource must have a unique name.

The two functions that control resource locking are described below:

Lock

Lock evaluates an expression and provides a set length of time in which eDeveloper will wait for the resource. The lock is successful if the expression evaluates to True. The lock is not successful if the resource is already locked or the timeout value has expired.

Unlocked

Unlock evaluates an expression to unlock a resource. Unlock is successful if the expression evaluates to zero. Unlock is not successful if the resource is already unlocked or locked by another user.

The Resource Lock File

A new resource file path name called *ResourceLockFilePath* has been added to the Environment settings.

SQL Range

An SQL Where range is not allowed for tasks that are executed within a deferred transaction. To deliver the SQL Where range functionality for deferred transactions, the Toolkit range definitions need to be altered.

The current known range types are:

- Select ranges – Ranges defined for the Select operation. This range is evaluated when a task is opened, and may change only during the Refresh View operation.

- SQL Where range – Available for SQL tasks only. The SQL Where range contains a part of an SQL Where clause to be appended to each SQL command that eDeveloper generates for that task.
- Task range – A part of a task's control properties. The Task range contains an eDeveloper expression, which is evaluated for every record fetched. The Task range filters out records that evaluate to FALSE.

The last two ranges are organized in a new ranges tab section that is under the task's property tab.

There is a fourth range type that is called the eDeveloper SQL Where range. It is similar to the existing SQL Where range with one exception – it is defined using a subset of eDeveloper expressions (unlike the existing SQL Where range, which uses free format text). The eDeveloper SQL Where range may only include those expressions that the engine can translate to SQL.

The eDeveloper SQL Where range is executed as follows:

- At Runtime, the eDeveloper SQL Where range is added to the other ranges using an AND relation.
- When running a task within a deferred transaction the SQL Where clause is ignored. eDeveloper only uses the remaining 3 ranges.
- To eliminate ambiguities, the existing range names will have the following changes:
 - SQL Where range is changed to DB SQL Where range
 - Task range is changed to Re-computed range
 - The new eDeveloper SQL Where range is changed to SQL Where range.
 - Within the new range tab section, a concatenation of the whole Where clause produced at Runtime is displayed. This includes the Select ranges, the new SQL Where Range, the re-computed range, and, for physical tasks (or tasks which inherit their transaction type) – the DB SQL Where range.

Note: Both the DB SQL where range and the eDeveloper SQL where range are both computed once, before the task prefix.



For more information, refer to the SQL Considerations chapter of *The eDeveloper Reference Guide*.

The Error Handling feature lets you overwrite eDeveloper's default behavior for the different errors that can arise during an application's execution.

This chapter includes the following topics:



- Error Handling Mechanism
- Error Behavior Strategies
- Error Handlers Toolkit Definition
- Runtime Error Handling
- Range and Locate a Task According to a Record's Position

Error Handling Mechanism

The error handling mechanism lets you overwrite eDeveloper's default behavior for the different errors that can arise during an application's execution.

There are two levels in which you are able to control eDeveloper's behavior at error time. The first level is similar to the existing On Error mechanism, but more robust. The Error Behavior Strategy property is defined on the Task Level, as one of the Task properties, and provides a choice of two predefined strategies, which are described in detail below.

The second is a more sophisticated error handling level that lets you write actual eDeveloper code for the errors, using error handlers. eDeveloper provides a list of known and expected errors that can be intercepted, and lets you handle other errors specific to your DBMS error code. This level also provides you with a list of new engine directives to control the behavior of the eDeveloper engine after implementing the error handler.

The errors that are described in this chapter are for Database related errors only.

Error Behavior Strategies

eDeveloper provides two different error-handling strategies – *Abort* and *Recover*. The strategy option is set at the Task level in the Task Properties window. This means that if the strategy is chosen carefully, you will not have to change any of the defaults of the Error Handling Task properties or write handlers for error handling. Of course, you can still change the default settings by writing error handlers in those places where the default strategy does not apply.

Abort Strategy

Whenever an error occurs, eDeveloper rolls back the current transaction, whether physical or deferred, removes the current dataview, and aborts the task in which the error occurs. This does not apply for all errors. The exception is for errors that the end-user can recover from, such as locking errors and incorrect login.

Recover Strategy

Whenever possible eDeveloper will keep the current dataview, stay on the current task, and allow the end-user to recover from the error. Recover here refers to the end-user continuing to work in the application after an error has occurred. On some errors, such as `Locking` and `Maximum connections exceeded`, recover means that eDeveloper retries the operation automatically, without input from the end-user.

Error Handlers - Toolkit Definition

You can write an error handler in the tasks's Execution table, which is the same table that lets you define event handlers.

Error Handler Properties

An error event has the following properties:

- Level
- Type
- Scope
- Enabled
- Engine Directive

Level

The level should be set to *Handler*.

Event

In the event column you can zoom to set the event as an Error type and select the actual error event from the error list.

Engine Directive

In the details column by the ‘Dir.’ label you can define the desired engine directive. The Engine Directive property indicates to eDeveloper the action that will follow after the error handler has been executed.

By the ‘Msg.’ label you can define Yes/No values or expressions that will determine whether eDeveloper will issue its default message box.

Scope

The Scope property allows for the following options:

- SubTree - default
- Task - Indicates if this handler will execute for the specific task only, or can be executed for the entire sub-tree of the task.

Enabled

Indicates if the handler is enabled. This property can be an expression. If it evaluates to **No**, the handler will not intercept the error.

Runtime Error Handling

When eDeveloper encounters an error situation, the eDeveloper error handling mechanism searches for a defined error handler to intercept the error in the same task, according to the error name. If the error handler exists, eDeveloper performs the operations that are defined for the handler, and then performs the corresponding action according to the engine directive.

The search for an error handler is similar to the search for an event handler. If the handler does not exist in the task, higher task levels are searched for an appropriate handler, according to the error name.

If the required error handler does not exist, and a handler for “Any” is defined, the error handler for “Any” is executed.

Range and Locate a Task According to a Record's Position

An important feature for error handling is to provide you with a method to display the error record or range of records according to their position.

eDeveloper lets you:

1. Execute a function to return the current record's position.
2. Execute a function to return the position of the record on which the error occurred.
3. Locate a task according to a specific position.



For more information refer to The Application Engine chapter of *The eDeveloper Reference Guide*.

Multi-threading lets a single background engine process multiple requests simultaneously. The multi-threaded architecture saves on resources and provides a faster performance.



This chapter includes the following topics:

- Environment Settings
- Runtime Expressions

Environment Settings

Maximum Number of Concurrent Requests

The Maximum Concurrent Requests setting is under the Partitioning Web tab from the Environment Settings dialog. You can enter the number of threads the Application Server is allowed to create. You are only limited by server capacity and type of license. The MAGIC.INI file and Command Line name is MaxConcurrentRequests.

Current Application

The current application is always the same for all running threads within a single process. To close the application, all threads must first be closed.

INIPUT and INIGET

The MAGIC.INI file is stored for each Runtime context in a separate memory area. The INIPUT function for a specific Runtime context is written in the MAGIC.INI file only if it is specifically requested by using the following flag:

INIPUT (<assignment>,<force write>) (default - N)

Security

You can add or remove user rights to each thread using the Runtime (RT) functions as described in the Actions and Functions chapter

TCP/IP Ports

Each application server process uses one TCP/IP port.

Broker and the Generic Messaging Layer

No interoperability between eDeveloper and any previous eDeveloper-based applications is possible.

Main Programs

A separate copy of the Main Program is available for each Runtime context.

Transactions

Each thread acts like a separate process, independent of the other threads.

Locking

Each thread acts as a separate process. Each Application Server process uses one terminal number.

Memory Tables

Each context opens a separate copy of a memory table.

Resident Tables

All contexts share the same copy of a resident table.

Shared Resources

The following resources are shared by every Runtime thread:

- DBMS connections
- External devices

The following functions allow global variables to be shared across contexts:

- SharedValueSet
- SharedValueGet

The following functions let you send global variables defined in one context to be sent to another context:

- ParamsPack
- ParamsUnpack

Runtime Expressions

The following Runtime functions are available:

- RQEXE - Requests an eDeveloper Request Broker to load a new entry from a predefined list of executables.
- RQRTTRM - Terminates an Application Server.
- RQRTINF - Returns the current number of busy threads, the maximum number of

threads allowed by a license, and the most frequently used threads.

- DISCSVR - Disconnects the current thread that is no longer required by eDeveloper.
- DBRELOAD - Loads a resident table during Runtime, and returns a logical value indicating the success or failure of the load.



For more information, refer to the Application Partitioning chapter of *The eDeveloper Reference Guide*.

e Developer has an exciting new look and feel. A multi-option navigation pane, folders, and multi-marking in tables are all part of the dynamic and easy-to-use interface package.



This chapter includes the following topics:

- Table Control
- Tree Control
- Drag and Drop Support

Table Control

eDeveloper provides an enhanced Table control that allows various new features: multi-marking, full placement, and column handling (auto sort and resizing) in Toolkit and Runtime.

Multi-Marking Functionality

Multi-marking is similar to the multi-marking in a Windows application.

Allow Multi-Marking Property

To enable multi-marking in the Table control of a task, click **Yes** in the Allow Multi-marking property of the table control property sheet. The Allow Multi-marking property is enabled for interactive online tasks only.

Marked rows are shown by a different color.

On marked records you can:

- Delete a record in Toolkit and Runtime
- Check the syntax of a record in Toolkit
- Define your own logic by a multi-marking enabled handler.

Marking Columns

To make a column a marking column, click **Yes** in the Marking Column property of the Column Property dialog. When the Marking Column property is enabled, the record cells of the column appear raised. Clicking a record on an enabled column will mark the record. Clicking on a marked record will unmark the record.

Multi-Marking Event Handlers

When an event is raised while multiple records are marked, the handler will be performed in a batch-like manner for each marked record.

In addition, for each processed marked record the record prefix is performed, and if the record was updated or the record suffix is forced, then the record suffix will be performed.

As the same handler flow is similarly performed for all marked records in the same manner, new functions are created to indicate that multiple records are marked and to specify which record is the first to be marked and which is the last.

The processing of records is performed according to their location in the dataview and not according to the order by which they were marked.

Multi-Marking Functions

The multi-marking functions are listed below.

MMCOUNT	The MMCOUNT function provides the number of marked rows. If the MMCOUNT function returns a value greater than zero it means that the records are marked.
MMCURR	The MMCURR function displays the number of the current row out of the number of the marked rows in the counting process.
MMCLEAR	The MMCLEAR function clears the marking of the rows.
MMSTOP	The MMSTOP function lets the user halt the multi-mark handler. If the MMCURR function returns 1 it means that the first marked record is being processed. If the MMCURR function equals to the MMCOUNT function it means that the last record is being processed.

Table Control Placement

When you resize a form grid that has a Table control, the Table control may resize accordingly.

For horizontal placement, the Table control has the Size Placement property that controls the resizing percentage for the table. For example, if the Size Placement property is set to 80%, and the size of the form is increased by 10 pixels, the size of the Table control is increased by 8 pixels.

For vertical placement, the Table control will display more or less rows according to how the form is resized. The Table control displays empty rows when no more data is available. The parked Table control row is always displayed.

Column Resizing in Toolkit Mode

Each column is resized according to the resizing of the table. Each column has a Placement property. When the Placement property is set to **Yes**, the column is resized in proportion to the total number of columns. For example, if each of the four columns has Yes for the Placement property, each column makes up twenty-five percent of the table.

When the column Placement property is set to **No**, the column size will not change. For example, assume a Table control that is 80 inches wide. The Size Placement property is set to 100%. There are four columns, each 20 inches wide. Two columns have Placement property = **Yes**, and two columns have Placement property = **No**. When the form grid is increased by 8 inches, two columns will be 20 inches wide, and two columns will be 24 inches wide.

If no column is set with Placement=Yes, then no horizontal placement will be set for the table even if the table has a corresponding placement setting.

Column Resizing

You can set the columns to be resizable using the new table control. This will allow the end user to change the width of the columns of a table.

A column can be resized simply by placing the pointer on the column divider and dragging the divider sideways.

Columns can also be resized in the following two ways:

In default mode you move the divider between the adjacent columns causing the size of both columns to be changed respectively to the shifting of the divider.

Dragging the divider while pressing the SHIFT key will change the width of the left-hand column and shift all the columns to the right of the divider, respectively.

Tree Control

eDeveloper provides a new control, the Tree Control, which is used to display the hierarchy of a task dataview in an hierarchical way using a tree structure with parent and child nodes that represent different data levels.

The different levels that can be displayed are:

- Root Value – This is the starting point for the data tree. The root value is displayed as the top node of the data in the tree.
- Parent Node - A variable representing a parent record in the task dataview.
- Node - A variable representing a child record in the task dataview.

The Tree control can only display variables from the main table or from a Link Join table that are defined in a single database table.

Drag and Drop Support

Drag and drop functionality is now supported within an eDeveloper-based application and between an eDeveloper and other applications running on the same computer. You can now pass information within an eDeveloper-based application and between applications by dragging and dropping, which is standard behavior in Windows applications.

The Allow Drag control property determines the ability to perform a drag operation, and the Allow Drop control property determines the ability to perform a drop operation.

New events and functions have been added to let you control the data in the drag operation.

e Developer provides many new and enhanced productivity capabilities.

This chapter includes the following topics:



- Main Program
- Data Control
- Models
- Flat eDeveloper-based File
- Task Return Value

Main Program

The Main Program is executed when opening an application in Runtime or when toggling from Toolkit to Runtime. The Main Program lets you define global resources for the entire application. The Main Program cannot be deleted.

Main Program Runtime Behavior

- The Main Program automatically runs in the background when eDeveloper opens a new application in Runtime, or when you switch the application from Toolkit to Runtime. Also, the Main Program is run when you click Execute from the Program repository or Generate Program from the Table repository.
- The Main Program does not have a main table. It executes as a batch program and ends

after one cycle: Task Prefix, Record Main, and Task Suffix.

- When the Main Program is implemented, the defined local variables, forms, DB tables, Events and handlers can be accessed throughout the execution of the whole application.

Main Program Toolkit Behavior

- The Main Program is always the first program listed in the Program repository.
- The Main Program is automatically created when a new application is created.
- The Main Program cannot be deleted.
- You cannot manually implement the Main Program.
- The Main Program task type is set to Batch and cannot be modified.

Main Program and Components

The Main Program cannot be exported as a component.

When you access a component in Runtime, the Main Program of the host application is loaded before the component.

When a component is loaded into another application, the Main Program of the component is inserted in the task execution stack after the Main Program of the application and before all other programs.

Main Program Functions

RUNMODE

In previous eDeveloper-based applications, the type of engine, Toolkit or Runtime, determined if certain programs were automatically implemented, such as, the Start and End Programs. To provide for the same behavior, the RUNMODE function returns a value code for an eDeveloper engine.

When a program is run under a full Runtime engine (regular and background engines), RUNMODE () = 0.

When a program is run under the Toolkit engine and the application initially opened in Runtime, RUNMODE () = 1.

When a program is run under the Toolkit engine, and the application was switched to Runtime (CTRL+T), RUNMODE () = 2.

When a program is run under the Toolkit engine, and the program is run by clicking F7 or by opening the Automatic Program Generator, RUNMODE () = 3.

PROG

Displays the task path of the current task. PROG does not display the Main Program task of the task path because every task path includes the Main Program.

TDEPTH

Returns the number of task levels in a Runtime tree. The TDEPTH function does not count the Main Program as a task level.



For more information, refer to the Application Engine chapter of *The eDeveloper Reference Guide*.

Data Control

In previous eDeveloper-based applications, you could only set Combo or List Box controls with an alpha field or expression. You often had to develop a subtask to create a control options string that contains other strings delimited by commas.

eDeveloper lets you define the Combo or List box as a data control. This lets you choose a value range directly from a database field.

Combo and List Box Control Properties

The following properties have been added to the Control and List Box controls:

- Source Table - The table index as it appears in the Table repository.
- Displayed Field - The field index from the selected table. This field will be displayed for each record of the source table.
- Linked Field - The field index from the selected table. This field will be returned whenever the corresponding display field will be selected in runtime.
- Index - An index name from the selected table. The index can be selected from the table indexes.
- Field Ranges - Lets you insert From/To ranges for each field.



For more information, refer to the Display Forms chapter in *The eDeveloper Reference Guide*.

Models

A model is a set of inheritable properties for a specific object type. eDeveloper lets you define models for the following objects:

- Fields
- Controls
- Forms
- Help Screens

When model type properties are updated, the values are reflected for each associated object for all property values, except for those defined individually.

Model Repository

The Model repository displays object models. Object models serve as a template for an object type. Each object is automatically assigned a set of default property values. Every object associated to the object model will have the same property values. You can break the inheritance of a property value by changing the value.

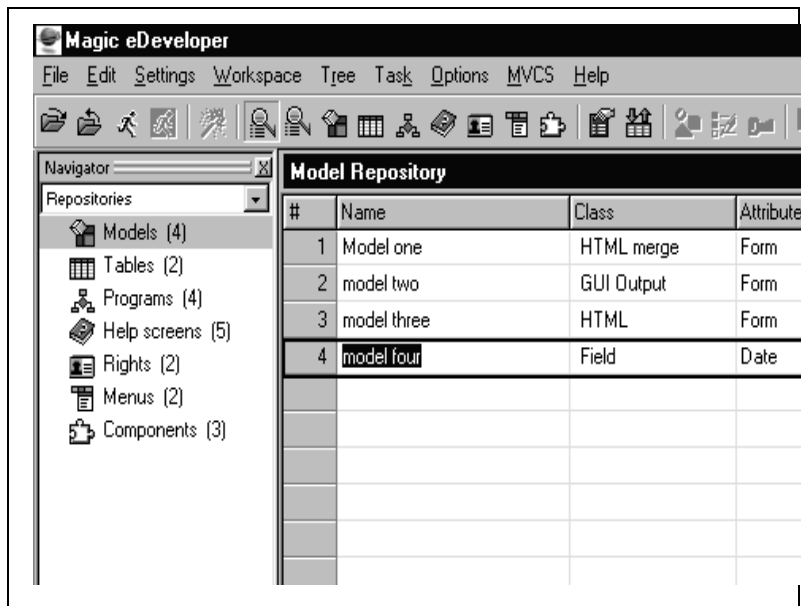


Figure 9-1 The Model Repository

Model Repository Attributes

- Name - The name of the user-defined model.
- Class - Model- assigned objects, such as:
 - Fields
 - Help screens
 - Interface types such as: Browser, GUI Display, GUI Output, Text based, HTML, Frameset and HTML Merge.
- Attribute – Each class has its own subdivision of attributes
- Fields attributes:
 - Alpha - A string of alphanumeric characters

- Numeric - An integer or decimal number
- Logical - 0 or 1, usually stored internally as a single byte
- Date - The numeric date field counts days
- Time - The numeric time field counts seconds
- Memo - A variable length alpha parameter
- Blob - Binary information, not created in eDeveloper
- Help screens attributes:
 - Internal - Helps defined within the eDeveloper application
 - Windows - Helps defined outside of the eDeveloper application
- Interface types attributes:
 - The attributes of the various interface types are the available controls of the selected interface type class.

Working with Models

A new object is associated with the system-based model by default. When a new application is created, it is automatically defined with system-based models and system based model is marked as (default).

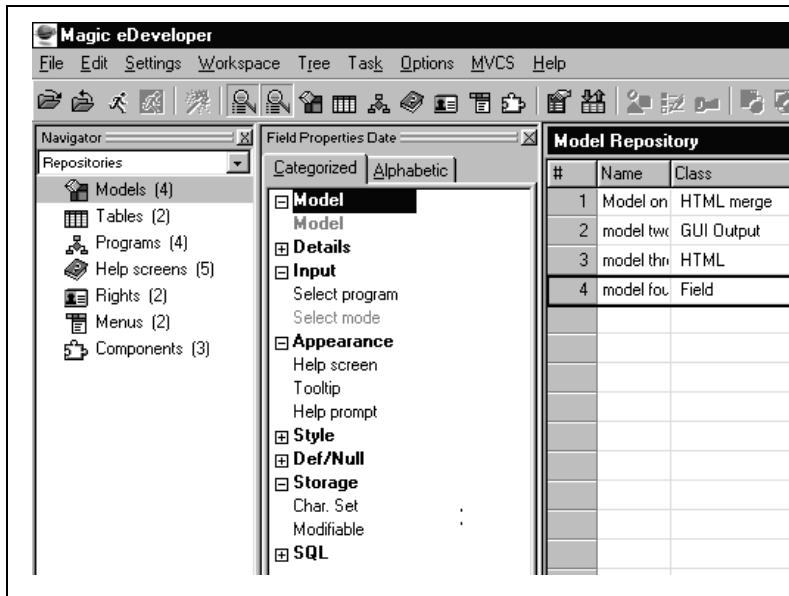


Figure 9-2 The Model Properties Sheet

Properties

For each class type a default set of properties have been defined. a default property is displayed in italics. You can break the inheritance of a property to an object model by clicking the Break Inheritance button that appears to the left as displayed below. A broken property does not appear in italics.

Breaking Model Properties

You can define an object value locally. When an object is defined locally, it no longer inherits the current or updated property value of the associated model. Property values that have been *broken* can be re-inherited.

Selecting a Different Model for an Object

When the model is replaced by another, the associated object inherits the properties of the new model. Properties that are defined locally are *broken* from the system-based default properties. Locally defined properties will not be updated with the update of the system-based default properties.



For more information, refer to the Models chapter in *The eDeveloper Reference Guide*.

Deploying an eDeveloper Flat File

The eDeveloper Flat File (MFF) lets you deploy a database-independent eDeveloper Application File. The application, which is created as a binary file, can be stored anywhere on the user's computer system and is used for deployment only.

MFF files are meant to be used in a multi-platform environment. For example, you can transfer an MFF from Windows NT to UNIX without having to import or export the application.

Select Flat eDeveloper File Deployment for an Application

The Flat MFF Deployment field is displayed on the Application Properties dialog. If the user clicks **Yes**, the MAGIC.INI file is modified to let you only read information from a flat MFF file.

When selecting an application as a flat MFF file, you do not need to define the Database field.

A eDeveloper application cannot be created both as a Flat MFF Deployment file and as a Compressed file. When one parameter is selected, the other is automatically disabled.

Save an eDeveloper Application as an MFF

You can save an existing eDeveloper application as an MFF by clicking the Save as MFF option from the File menu. When the eDeveloper application is saved as an MFF, you cannot open the file in Toolkit. The file is designated for deployment only.



For more information, refer to the Settings chapter in *The eDeveloper Reference Guide*.

Task Return Value

The eDeveloper task has a new property called *Return Value* that can be assigned with a variable or an expression. Upon completion of a task, this value is returned.

Using a new function called CALLPROG(), a program can be called by defining an expression that contains the function, which is evaluated as the return value of the called program.

Left Outer Join for ISAM databases

You can now select the Link Outer Join option for a Link Operation regardless of the database. For ISAM tables, eDeveloper opens the table and retrieves the linked data as a Link Query operation.

e Developer provides many new and enhanced Toolkit features.

This chapter includes the following topics:



- Argument Matching
- Block Operation
- Vector Support
- eDeveloper Wizard
- Comments
- Interface Navigation
- Find and Replace
- Working with the ANSI Engine
- Referential Integrity
- Link Condition

Argument Matching

In previous eDeveloper-based applications, you could not differentiate variables from parameters. eDeveloper lets you declare specific local variables as parameters. Such a declaration is reflected in the Call operation as expected arguments.

Select Parameter

The Parameter variable option has been added to the Select operation. Now the Select operation has three variable options:

- Real
- Virtual
- Parameter



Parameters receive arguments passed by another task.

If no Select Parameter type is set in a task, virtual variables receive the passed arguments.

Parameters Table

Parameters are defined and displayed in the Parameters table.

Arguments List

The Parameters list of the Call operation has been renamed as the Arguments list.

The Select Parameter type is reflected in the Argument repository of a Call Program operation. The Argument repository displays the number and details of the parameters in the Call Program.

Calling a Program with Selected Parameters Defined

When a Call operation calls a task with parameters, you can zoom to the Arguments list to display the parameters of the called program or task. You can add or move entries to match the arguments to the parameters.

Calling a Program With No Select Parameters Defined

When a Call operation calls a task that has no Select parameters, no new lines are displayed in the Arguments list. The selection of variables for the argument has no restrictions. The arguments are matched by virtual fields of the called task according to their order.



For more information, refer to the Operations chapter of *The eDeveloper Reference Guide*.

Block Operation

The Block operation has been enhanced.

If-Else

The Block operation now includes an Else option. In previous eDeveloper-based applications, you had to create a new block for each separate condition. eDeveloper lets you define an If-Else Block so that if the first condition proves false, the engine will go to each Else option defined in a single Block operation.

You can have multiple Else options defined within one Block operation.

Loop

eDeveloper has introduced a Block Loop operation. Block Loop instructs the eDeveloper engine to repeatedly execute the operations contained within the block until a condition is met.

You can monitor the number of times the Block Loop operation was executed by using the LoopCounter function.



For more information, see the LoopCounter function in the chapter on Expression Rules *The eDeveloper Reference Guide*.

Vector Support

eDeveloper now supports vectors or arrays. To support vectors, a new variable attribute has been added, Vector. A vector in eDeveloper is based on a model. This model governs the properties of each cell in the vector.

Various functions are provided to work with vectors:

- VecSet – This adds data to a cell in the vector
- VecGet – This retrieves the data from a cell to a vector
- VecSize – This fetches the total number of cells created in the vector
- VecCellAttr – This retrieves the cell attribute of the vector

eDeveloper Wizard

The eDeveloper Wizard lets eDeveloper developers easily access the eDeveloper development environment and quickly create a working application.

The eDeveloper Wizard lets you create data objects, interactive programs, reports, and batch programs. The eDeveloper Wizard is available from the Tools menu in a eDeveloper application.

Enhanced Checker

eDeveloper offers an enhanced syntax checker that displays the messages in a separate floating and dockable window with easy navigational capabilities.

You can control the display of the checker results using various environment settings.

Comments

eDeveloper lets you attach text comments to various objects in the application as displayed below:

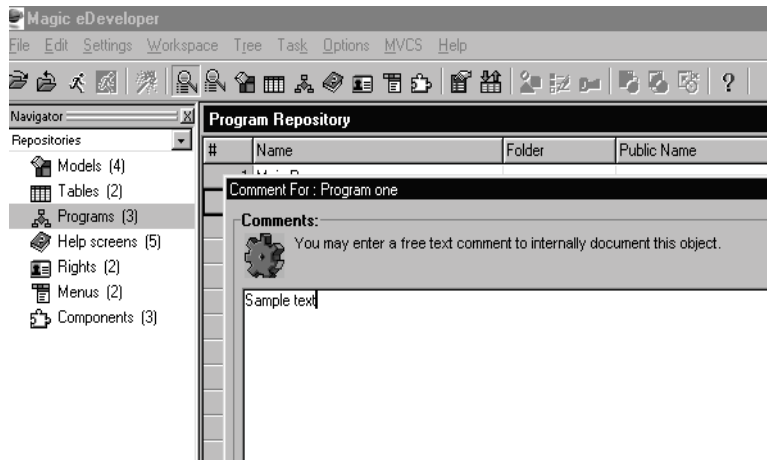


Figure 10-1 The Comments Dialog

The application objects are:

- Models
- Tables
 - Column
 - Indexes
 - Foreign keys

- Programs
 - Tasks
 - Local fields (virtuals and parameters)
 - I/O - each I/O
 - Forms and Controls
 - Handlers
- Helps
- Rights
- Menus
- Components



For more information, refer to the Introduction chapter of *The eDeveloper Reference Guide*.

Interface Navigation

eDeveloper has dynamic Toolkit repositories and a user-friendly Navigator. The eDeveloper main screen is displayed below:

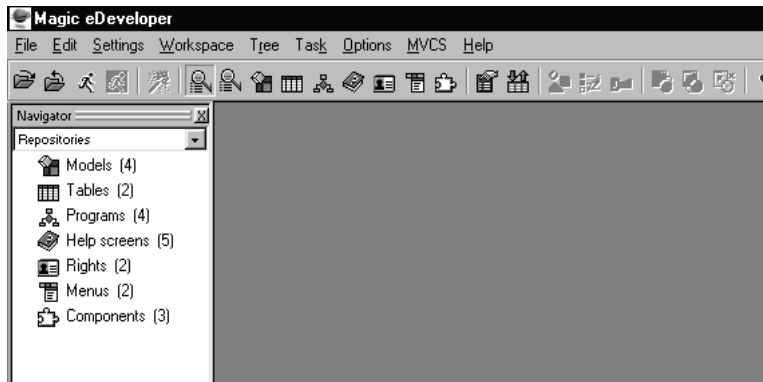


Figure 10-2 The eDeveloper Main Screen

The Navigator lets you navigate through the application. You can resize the Navigator pane to allow for greater work space.

The Workspace pane displays the data and settings for the repositories, property sheets, and forms of a eDeveloper application. You can also define a keyboard shortcut to change the focus from one screen to the other.

By default the Navigator pane appears on the left side of the eDeveloper screen. You can also dock the navigator to any border of the eDeveloper window or have it undocked and floating above or below.

There are four options available from the Navigator box. They are:

- Repositories
- Task
- X-ref
- Bookmarks

The Navigator can be closed at any time to give more MDI space to the application. The Navigation pane also automatically closes when you switch from Toolkit to Runtime or when you invoke a program.

Repositories

The Repositories tab displays the Folder tree. Each main entry represents a repository object such as models, tables, and programs. The Folder tree can be expanded to another level to show the user-defined folders for the relevant repository as shown below:

The Folder Tree view provides the following information:

- Collapse and Expand entry image
- Entry name
- Numbers of entry items

A folder is an object that lets you organize entries in a repository. A folder lets the user organize a range of entries in a repository. For example, a folder may contain tables 10 to 13 and another folder may contain tables 14 to 27.

Each repository can hold up to 100 folders. Each folder represents a sequential range of entries in the repository. When a user moves an entry from one folder to another, eDeveloper automatically rennumbers the entries to maintain an unbroken sequence of listed entries.

An entry does not need to be stored in a folder. Individual entries appear at the top of the entry list, and are visible when the entire repository is viewed.

The application's repositories are displayed on selecting the 'Repositories' option from the Navigator box. The user can create user-defined folders for the following repositories:

- Models
- Table
- Program
- Help
- Rights
- Components
- Events

In each of the repositories listed above, a Folder column lets the user specify the folder where an object is placed.

Task

The Task option is available in the Navigator when developing a program.

The Task view of the Navigator provides a hierarchical view of the program tree.

In this tree view of the task you may:

- Navigate through the main task and its subtask
- Manipulate the program tree structure.
- Copy and paste task subtrees.

Cross-Reference

The Navigator has an X-ref option which keeps the cross-reference results. Each Cross-reference result and the resulting objects are displayed in a tree format.

You can Click the result entry to switch to the corresponding location in the in the Workspace pane and park on the selected entry.

Cross-references have the following characteristics:

- You can delete a cross-reference result.
- You can save the Cross-reference results to a text file.
- You can print the cross reference result.
- You can set the maximum number of Cross-reference results to be kept.

Bookmarks

The Navigator's Bookmark option shows the bookmarked objects of an application.

Bookmark Object

The Bookmark Object action lets you create a link to the object displayed under the bookmark option of the Navigator. The user can define a bookmark by clicking **CTRL+B**, or using the bookmark option under the *Options* menu.

Each bookmark is automatically provided with a default name, which you can rename.

Bookmarks have the following characteristics:

- Bookmark entries appear in order of their creation.
- Bookmarking a location creates the entry at the top of the list.
- You can delete a bookmark entry from the Bookmarks tab.
- You can set the number of kept bookmarks in The Number of Kept Bookmarks environment setting, under the Preference tab of the Environment dialog.
- All bookmarks for each application are stored in the Windows registry according to the name of the application.

Find and Replace

eDeveloper now enables a developer to perform Find and Replace of text throughout the application. The results are displayed in the navigation pane as cross-reference results. For Replace, eDeveloper runs through the application and replaces text while enabling the developer to confirm the replace operation.

eDeveloper enables a developer to Find and Replace objects within a respository.

Working with the ANSI Engine

In previous versions, eDeveloper used an OEM character set for all of its internal files and all of its keyboard-to-display uses. Windows uses an ANSI character set for the same purposes. eDeveloper's foreign language Toolkit and Runtime engines for foreign language support now use the ANSI standard. All of the various strings of a eDeveloper Control File (MCF) file are stored in ANSI.

Changes in the Environment Settings

The GUI Translation File

The GUI Translation file (*Settings/Environment/External Files tab*) has been removed.

The OEM_ANSI Translation File

The Native Translation File has been replaced with the OEM_ANSI Translation file. This file maintains the location and filename of the external file that holds the translation table for eDeveloper internal character symbols and the platform character table. In eDeveloper, the OEM_ANSI Translation file lets the user translate from OEM to ANSI and from ANSI to OEM, or from whatever character set translation the user requires. If this file is not set, then any OEM/ANSI conversion will be done according to the OEM/ANSI conversion of the operating system. The default filename is OEM_ANSI.

The OEM_ANSI Translation parameter appears in the Field and I/O property sheets. This property is called *Character-set to use*, and accepts the values of either ANSI or OEM. When a field or I/O is set to ANSI, its content is regarded as ANSI and no translation occurs. When a field or I/O is set to OEM, its contents is regarded as OEM and eDeveloper translates the content from ANSI and from OEM to ANSI when it reads from it.

Security File

The Security File Convention environment property appears in the External Files tab. If the property is set to ANSI, the file is considered ANSI and no conversion is performed. If the property is set to OEM, the file is considered OEM and the data is converted from screen to screen.



For more information, see the Settings chapter of *The eDeveloper Reference Guide*.

Standard and Internal I/O Files

In the I/O repository, the Standard and Internal Files have been replaced by *File*. Internal files are imported as File with the Character Set to Use parameter set to OEM. Standard files are imported as File with the Character Set to Use parameter set to ANSI.

Standard and Internal Fields

The translation parameter for fields has been changed to *Character set to use*. Fields that were set to Internal translation are imported with the Character Set to Use parameter set to OEM. Fields that were set to Standard translation are imported with the Character Set to Use parameter set to ANSI.

Functions

Two new functions have been added to let users convert data from ANSI to OEM and OEM to ANSI. These functions receive a string and return another string, which is the translation of the source string according to the OEM_ANSI translation file if set, or to the operating system conversion if not set.

The syntax for each function is as follows:

OEM2ANSI (*string*)

ANSI2OEM (*string*)



For more information, refer to the Expressions chapter of *The eDeveloper Reference Guide*.

The OEM_ANSI. EXE Utility

A conversion utility lets you convert an entire series of external text files from ANSI to OEM and from OEM to ANSI. The purpose of this utility is to convert various files from previous versions that are kept in OEM (mainly support files) to be used in eDeveloper applications.

Export/Import

eDeveloper exports documents as ANSI text.

The Import utility detects the documents imported from previous eDeveloper-based applications. When a document from a previous version is detected, eDeveloper treats the file as written in OEM. eDeveloper then translates the document to ANSI by using the OEM_ANSI translation file.



For more information, see the Utilities chapter of *The eDeveloper Reference Guide*.

Effect on Previous-version Applications

If you have an English-language application which uses the standard English character set, the change will probably not affect your application

If you are using non-English language applications note the following:

- Any previous-version application, which used the native translation file to keep the data in ANSI will no longer require translation. However, if you have an application that did not use the native translation file, and you kept the data in an OEM character set, you should use the eDeveloper translation mechanism to convert it to the ANSI character set.
- If you used the CHR function to insert or display characters which correspond to the OEM character set, you should change these expressions to correspond to the ANSI character set. This also applies for the ASC function.

Referential Integrity

SQL lets you define the relationship between tables using Foreign Keys. The Foreign Key is defined as several table segments that point to a Primary Key defined in another SQL table.

Setting Foreign Keys for table segments lets you maintain referential integrity between the Primary Key and all of the table segments related to it. For example, you cannot delete the table's defined Primary Key without also deleting the table segments with Foreign Keys pointing to the main table.

eDeveloper lets you define Primary Keys and Foreign Keys in the Table repository.

Table Repository

Foreign Keys

A Foreign Keys column has been added to the Table repository. When zooming in the column, the Foreign Keys repository appears.

The Foreign Keys repository has two tables: the Foreign Keys Definition table and the Foreign Keys Segment table.

The Foreign Keys Definition table displays the following information:

- Foreign Key Number - An internal, automatically generated number that represents the order of the Foreign Key.
- Foreign Key Name - The name of the Foreign Key in eDeveloper. This name is also the database name for Foreign Keys created in the database.
- Referenced Table - A selected from all the eDeveloper tables for all the databases. Only eDeveloper tables can be referred.
- Use Index - Selecting a specific predefined index from the referred table (not mandatory).
- Create in DB - This check box will be checked by default if the referenced table is in the same SQL database.

The Foreign Keys Segments table displays the following information:

- Current Table Segment Number - An internal, automatically generated number that represents the order of the current table segment.
- Current Table Segments - These segments can be selected from the current table by zooming in the column list. Each segment in the current table must have a matching segment in the referenced table.
- Referenced Table Segment Number - An internal, automatically generated number that represents the order of the referenced table segment.
- Referenced Table Segments - These segments can be automatically inserted when selecting the index defined for the referenced table. The order of the segments can be changed only when no index is defined. Exchanging one index for another overwrites the segments with the new index segments.

Automated Program Generator (APG)

When the APG is executed for a table with Foreign keys, the default task contains only fields from the table itself (same as previous eDeveloper-based applications).

The Links tab, which displays a list of all the Foreign Keys defined for the table, has been added to the APG interface. Each Foreign Key has a corresponding number that indicates its order. You can change the Foreign Key order by changing the number. A Foreign Key is excluded from the order if you assign the number 0 to it.

Selecting a Foreign Key adds all of the referenced table columns, separated by a dividing line, to the selected column list. Though you can delete some of the fields in the selected columns list, the Primary Key segments of the referenced table or the Primary Key segments from the referencing table cannot be deleted. After selecting the Foreign Keys, and determining their order, the task will include a Link operation for each selected Foreign key.

Links to the referenced table appear at the end, after the selected columns of the main table. Each link has the following remark:

Link according to *<Foreign Key Name>* Foreign Key.

If the main and referenced tables are from the same SQL database, the created link is a Link Inner Join. Otherwise, it is a Link Query with validation = Yes.

The APG searches for Foreign Keys from the main table only. You cannot nest Foreign Keys in a referenced table.



For more information, refer to the Table Repository chapter of *The eDeveloper Reference Guide*.

Link Condition

eDeveloper-based applications do not use the Conditioning of Link operation to control the actual fetching of a record. eDeveloper has an enhanced condition to control the actual execution of the link.

Toolkit

You can define the new Link condition by entering either Yes, No, or an Expression value in the Cnd column of the task's Flow table.

If the Cnd column value is **Yes** or evaluates to a True value, the Link condition fetches the required record.

If the Cnd Column value is **False** or is evaluated to a False value, the record will not be fetched. The record will behave as a failed link, and the following will occur:

- All values of the Link operation's selected columns return to their default values.
- The return value of the link is False.
- Update operations do not take effect until the condition evaluates to True.
- If the link condition is evaluated to False at the end of a Record Suffix, any modification of a linked record that may have been fetched since the Record Prefix of the current logical record is disregarded.

User Interface

A new property has been added to the Link Property dialog to define the timing by which the Link condition is evaluated.

Evaluate link condition = Task/Record

Link Property Dialog

The CND column is now used for the new Link condition, while the Validation condition has been removed to the Link Properties dialog box.

The Validation condition is enabled for Link Query and Link Join.

The Validation condition combined with the Link Query can substitute any use of the Link Validate option, which has been removed.

When importing a previous version, the Link Validate is converted to the Link Query with the condition of the Link Validate placed in the validation condition.

Evaluate Link Condition Property

When the Evaluate Link Condition property is set to “Task”, the condition is evaluated once before fetching the entire dataview. If the condition is False, the link will not be part of the Select statement.

All records will not include the linked records. For example, the SQL statement for Link Joins does not include the Join part.

When you enter the Record Prefix and the condition for this record is evaluated as True, the link will not be re-computed as a Link Query.

When you set the property to Record for a Link Join or Link Outer operation and the Link condition evaluates to False, the linked data is fetched, whether it exists or not, and the fetched linked data is replaced with the default values of the linked fields. In the re-computation of the link, the Link Join/Outer Join acts as the regular Link Query.



For more information, refer to the Operations chapter of *The eDeveloper Reference Guide*.

Connectivity to Other Environments

11

eDeveloper provides many new and enhanced features that let you connect your eDeveloper application to other environments, and also let those environments connect to your eDeveloper application.



This chapter includes the following topics:

- LDAP Support
- Java Integration
- COM Support
- XML Support
- Web Service Support
- SNMP Support
- Message Queueing Support
- Mail, Clipboard, and HTTP functions

LDAP Support

eDeveloper can fetch user authorization from a LDAP (Lightweight Directory Access Protocol) environment. This means that the end-user does not have to log on twice, and the developer can use eDeveloper functions to retrieve information about the logged on user.

Java Integration

eDeveloper can interface with a Java class or with Enterprise JavaBeans (EJB) by using pseudo-references that represent instances of Java classes or EJB files.

Java Functions

eDeveloper has a set of internal functions that are used to simulate Java calling methods. These functions enable you to create an instance of a Java class that lets you activate a method or query and update variables in the class using a pseudo-reference. The pseudo-reference is a eDeveloper BLOB variable that represents the Java class instance.

Class static methods and variables can also be easily accessed in a similar way.

Java Access Component Generator

The Java Access Component Generator is a eDeveloper wizard that lets you automatically access Java classes or EJB files by creating a component that eliminates the need to specify JNI signatures.

J2EE Integration

eDeveloper can provide services as an EJB component within a J2EE framework. eDeveloper lets you create EJB components using your existing programs. The Component Builder generates and packages the modules you want to deploy in a J2EE environment and saves them in the form of a JAR file. The JAR file, which is deployed in a J2EE application server, enables a eDeveloper application to become an integral part of the J2EE environment.

COM Integration

eDeveloper has enhanced COM functionality so that you can easily call COM objects and provide a COM interface to external applications.

eDeveloper COM Client

eDeveloper provides you with full functionality for handling OLE objects and ActiveX controls. Using a simple interface, eDeveloper can call methods, get and set properties, and handle ActiveX events.

Two new variable attributes, OLE and ActiveX, define a variable as a COM variable.

A new Call operation, Call COM, lets you interface with the COM object.

When working with a COM object, the data that the object requires is often a variant, which is a COM datatype. eDeveloper has a set of functions that let you create a variant and fetch data from the variant object.

eDeveloper COM Interface

The eDeveloper COM Interface builder lets you create a COM interface for your application, which enables other non-eDeveloper-based applications to interface with your eDeveloper application.

Full XML Support

XML has become the standard format for transferring data between applications. eDeveloper fully supports XML, and can read and write to XML files.

XML Functions

A eDeveloper program can read XML data using a set of internal functions, which let you search, query, and fetch data from an XML file.

A new IO type, XML Direct Access, opens an XML file.

XML Access Component Generator

The eDeveloper XML Access Component Generator (XCG) simplifies the integration of

an application with an XML schema by generating a eDeveloper component that lets you access an XML document's data from within a eDeveloper application. The XCG uses an XML Schema Definition (XSD) file to create the component.

Full Web Service Support

eDeveloper has full Web Services integration, which lets you call a Web Service and lets a eDeveloper application provide Web Services.

Using Web Services

Sometimes a eDeveloper program must use Web Services provided by another application. A eDeveloper program can now call a Web Service by using the new Call Web Service (Call WS) operation. eDeveloper also provides an assistant to help the developer fill in the required Call properties.

Web Service Provider

The eDeveloper Web Service builder lets you give your eDeveloper application a Web Service interface that allows other applications to call your application using Web Services. The Web Service builder constructs WSDL files that provide the necessary information about the Web Services.

Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) governs network management and monitors network devices and their functions. Applications and various eDeveloper modules can send trap messages to a pre-configured SNMP monitor, which lets the system administrator query and control an application or a eDeveloper module when alarms, failures, or other exceptional events occur.

Message Queueing Support

Distributed applications often send messages from one application to another via message queues. eDeveloper has released a eDeveloper component that enables a developer to open a message queue, read from the message queue, write to the message queue, and close the message queue. Publish and subscribe functionality is also supported. eDeveloper supports three messaging systems: Microsoft (MSMQ), Sun (JMS API), and IBM (WebSphere MQ).

Connectivity Functions

eDeveloper has sets of internal functions that let you communicate with a mail server, the machine clipboard, and an external URL.

Email Functions

eDeveloper has a set of email functions that let you send and read messages from the mail server. For email retrieval, POP3 and IMAP are supported.

Clipboard Functions

eDeveloper now has a set of functions that let you write to the computer clipboard and read from it.

HTTP Functions

Using HTTPPOST and HTTPGET, you can send and fetch data from a remote URL.