

Ingres OpenROAD[®] 2006

Classic Getting Started

INGRES

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Ingres Corporation ("Ingres") at any time.

This Documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of Ingres. This Documentation is proprietary information of Ingres and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this Documentation for their own internal use, provided that all Ingres copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. The user consents to Ingres obtaining injunctive relief precluding any unauthorized use of the Documentation. Should the license terminate for any reason, it shall be the user's responsibility to return to Ingres the reproduced copies or to certify to Ingres that same have been destroyed.

To the extent permitted by applicable law, INGRES PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL INGRES BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USER OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF INGRES IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of any product referenced in this Documentation and this Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Documentation is Ingres Corporation.

For government users, the Documentation is delivered with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013 or applicable successor provisions.

Copyright © 2008 Ingres Corporation. All Rights Reserved.

Ingres, OpenROAD, and EDBC are registered trademarks of Ingres Corporation. All other trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

| | |
|---|-----------|
| Chapter 1: Introduction | 7 |
| Intended Audience | 7 |
| A Complete Solution for Developing New Applications | 8 |
| Application Development and Deployment in an N-tier Environment | 8 |
| Accessing and Integrating Data with OpenROAD | 9 |
| OpenROAD Documentation Set | 9 |
| Ingres Documentation | 10 |
| For More Information | 11 |
| | |
| Chapter 2: Features Overview | 13 |
| Multiple Platform Support | 13 |
| Open Database Access | 13 |
| Ingres Access | 14 |
| Enterprise Access Products | 14 |
| ActiveX Support | 15 |
| Interactive Development Environment | 15 |
| Complete Development Environment with Visual Tools | 16 |
| Application Workbench | 16 |
| Editors | 17 |
| Interactive Testing and Debugging | 19 |
| Report Writing | 19 |
| Application Management Utilities | 20 |
| Component Sharing | 20 |
| Team Development | 20 |
| Frame and Field Templates | 20 |
| Component Libraries | 20 |
| Open Architecture | 20 |
| Automatic Field Generation | 21 |
| Automatic Frame Generation | 21 |
| 4GL—A Fully Object-Oriented Language | 21 |
| System Classes and User Classes | 21 |
| Class Browser | 22 |
| Dynamic Applications | 22 |
| Active Repository | 22 |
| Component Reusability | 23 |
| Management Facilities | 23 |

| | |
|---|----|
| OpenROAD Server | 24 |
| Fixed Signature, Dynamic Data | 24 |
| Structured Data | 25 |
| eClient Deployment | 25 |
| eClient Runtime Cabinet File | 25 |
| eClient Application and Library Cabinet Files | 25 |
| How You Can Deploy Web Applications | 26 |

Chapter 3: Installing and Configuring OpenROAD on Windows **27**

| | |
|---|----|
| Upgrading from Earlier OpenROAD Releases | 27 |
| Prepare for Installation—Shut Down Ingres | 28 |
| Install OpenROAD on Windows | 28 |

Chapter 4: Installing and Configuring OpenROAD on UNIX **29**

| | |
|--|----|
| Install OpenROAD on UNIX | 29 |
| Install OpenROAD as a New Installation | 29 |
| Install OpenROAD into an Existing Ingres Installation | 30 |
| How You Can Perform Post-installation Customization | 33 |
| How You Can Update Your Registry on UNIX | 33 |
| MainWin System Core Subsystem | 36 |
| mwadm—MSC Administrator | 36 |
| Install MSC in Interactive Mode | 36 |
| Install MSC in Non-interactive Mode | 38 |
| Uninstall MSC | 39 |
| How an Administrator Can Administer the MainWin System Core | 39 |
| How You Can Display mwadm Usage Information | 40 |
| MSC Administrator Commands | 40 |
| How You Can Configure COM and DCOM | 52 |
| How You Can Use the DCOM Configuration Utility on UNIX | 52 |
| How You Can Set the Identity of an Out-of-process COM Server | 52 |
| Set the COM Server Executable File Ownership and Permissions to Run as a Specific User | 53 |
| Server Initialization Script | 54 |
| How You Can Set Up Distributed COM | 55 |
| How You Can Set Up DCOM for a Mixed Group of UNIX and Windows Hosts | 57 |

Chapter 5: Installing and Configuring OpenROAD on HP Tru64 **59**

| | |
|---|----|
| OpenROAD Installation on HP Tru64 | 59 |
| Install OpenROAD as a New Installation | 59 |
| Install OpenROAD into an Existing Ingres Installation | 60 |
| Update Your Registry on HP Tru64 | 61 |

| | |
|---|----|
| OpenROAD Utilities orspostart and orspostop | 62 |
| Configure COM and DCOM..... | 64 |

| | |
|--------------|-----------|
| Index | 65 |
|--------------|-----------|

Chapter 1: Introduction

This section contains the following topics:

[Intended Audience](#) (see page 7)

[A Complete Solution for Developing New Applications](#) (see page 8)

[Application Development and Deployment in an N-tier Environment](#)
(see page 8)

[Accessing and Integrating Data with OpenROAD](#) (see page 9)

[OpenROAD Documentation Set](#) (see page 9)

[Your Support Options](#) (see page 11)

OpenROAD® is an object-oriented, repository-based application development tool that lets you develop and deploy mission-critical, n-tier business applications in a variety of environments. Applications can be delivered seamlessly to a web browser, displaying rich graphical user interface (GUI) content usually seen only in desktop applications.

The *Getting Started* guide is your introduction to OpenROAD, providing what you need to know for a quick and productive start. It provides an overview of OpenROAD and its features, and explains how to install and configure the product on the Windows, UNIX, and HP Tru64 platforms.

Note: For more information on new and changed features, see the *Release Summary*.

Intended Audience

This guide is intended for application developers-first-time OpenROAD users as well as experienced users who want to familiarize themselves with some of the new features of this release.

To perform the instructions in this guide, you must have a working knowledge of operating system management or be in close contact with the operating system administrator. You must also be familiar with the windowing system on your target installation platform, including terminology, navigational techniques, and working with standard items such as menus and dialogs.

Note: Some procedures in this guide require operating system privileges or permissions. If you do not have these privileges and the operating system expertise associated with them, have the operating system administrator perform those tasks.

A Complete Solution for Developing New Applications

OpenROAD provides a foundation for object-oriented, cross-platform application development. On this foundation, you can build applications that take advantage of the advanced features and functionality available in OpenROAD.

Principal features include the following:

OpenROAD Server

Provides a robust, high-performance, distributed application development platform. OpenROAD 4GL business objects are deployed on the OpenROAD Server, providing services to a wide range of client technologies across the Internet.

OpenROAD eClient

Enables OpenROAD applications placed on a web server to be deployed so that clients can access them from a web browser.

OpenROAD mClient

Enables OpenROAD applications to be deployed to Windows Mobile-Based Pocket PC devices.

Application Development and Deployment in an N-tier Environment

OpenROAD provides the capability to develop robust, scalable client/server applications against databases such as Ingres, Microsoft SQL Server, Oracle, and DB2 UDB.

Many business demands now are driven by the Internet, leading to a greater need for true distributed applications. (A distributed or n-tier application can be separated into any number of logical tiers using reusable components. These tiers operate in multiple configurations, using any number of physical computers.) OpenROAD meets this demand through eClient technology, which enables development of Rich Internet Applications that can access services deployed on an OpenROAD Server.

The OpenROAD Server lets you reuse existing application logic while you develop new client applications using OpenROAD, or other disparate technologies such as .NET and Java. These applications can access existing data from mainframes and distributed databases through Enterprise Database Connectivity (EDBC) and Enterprise Access.

Accessing and Integrating Data with OpenROAD

Modern applications require that data from multiple sources on different platforms be accessible in a single application. OpenROAD uses Enterprise Access to provide a common, portable, open interface across a variety of operating environments. This enables OpenROAD to offer solutions for full, transparent read/write access to all existing data without changing its location or structure.

OpenROAD supports all popular UNIX and Microsoft Windows-based database management systems such as Oracle, DB2 UDB, and Microsoft SQL Server through Enterprise Access. On IBM z/OS and OS/390 mainframes, EDBC (a separate product) provides the same level of access to native VSAM, DB2, CICS/VSAM, IMS, Advantage CA-IDMS/DB Database, and Advantage CA-Datcom/DB Database to enable you to access data from anywhere.

OpenROAD Documentation Set

The following list summarizes the guides in the OpenROAD documentation set:

Installation and Configuration Guide

Details how to install OpenROAD on Microsoft Windows, UNIX or Linux, and HP-Tru64

Getting Started

Gives a general overview of the OpenROAD product, including an introduction to the eClient, OpenROAD Server, and some of the other new features in the product

Readme

Contains information not covered in the *Getting Started* or any other OpenROAD document. This document contains important installation information, platform-specific information, and any last-minute details.

Release Summary

Describes new features and enhancements

Server Reference Guide

Gives a general overview of and describes how to use the OpenROAD Server

Language Reference Guide

Provides detailed descriptions of all OpenROAD language components

System Reference Summary

Provides a summary of all OpenROAD system classes

Programming Guide

Describes basic and advanced OpenROAD programming techniques

User Guide

Describes how to use the OpenROAD development environment

The OpenROAD Application Workbench also includes an online help system.

Ingres Documentation

The following guides from the Ingres 2.6 documentation set are included with OpenROAD:

Ingres Command Reference Guide

Describes the commands available to configure and maintain the Ingres relational database system

Ingres OpenSQL Reference Guide

Describes OpenSQL, a subset of Ingres/SQL that enables you to use OpenROAD with database management systems other than Ingres through Enterprise Access

Ingres SQL Reference Guide

Provides detailed descriptions of all SQL statements, examples of the correct use of SQL statements and features, and details that help you use SQL effectively

Ingres Embedded SQL for C/C++ Companion Guide

Ingres Embedded SQL for COBOL Companion Guide

Describes how to use Embedded SQL with the specified programming languages. Use this book for language-dependent details that are not included in the *Ingres SQL Reference Guide*.

Ingres System Administrator's Guide

Describes for system administrators the Ingres architecture, and how to configure Ingres and its components

Ingres Database Administrator's Guide

Provides Ingres database administrators with information about creating, maintaining, backing up, and recovering databases, as well as instructions for defining various types of users and authorizing user access and for working with different types of database objects

For More Information

After reading this *Getting Started*, you can explore the numerous resources available for additional information. Your product installation media contains instructional documents that showcase your software and provide detailed explanations about the product's components. In addition, you can obtain procedural information and answers to any questions you may encounter by accessing the Ingres Corporation website at: <http://ingres.com> (<http://ingres.com/>).

Your Support Options

Enterprise customers with active maintenance and support contracts have full access to Ingres Support, including telephone support and online use of our call tracking system and knowledge base, Service Desk. For Customer Support contact details, see <http://ingres.com/support/contact.php>.

If you have an active support contract and want to register for access to Service Desk (<https://servicedesk.ingres.com>), use the enrollment form at <http://www.ingres.com/users/register.php>. (Your six-digit Account Number/Site ID is required.)

If you do not have a support agreement for Ingres Corporation products and are interested in purchasing support, contact us at sales@ingres.com.

For more information about support options, visit <http://ingres.com/support.php>.

Free support is available from the Ingres Open Source Community. Community members may obtain assistance for Ingres Corporation products by registering with the Ingres Community Site and using the available tools. To register, go to <http://community.ingres.com/forums/home.php> and click Register in the upper right corner of the page.

The Community Forums also provide Ingres Open Source Community members the opportunity to ask questions and interact with other community members and Ingres Corporation technical staff. For more information, visit <http://community.ingres.com/forums/index.php>.

Chapter 2: Features Overview

This section contains the following topics:

- [Multiple Platform Support](#) (see page 13)
- [Open Database Access](#) (see page 13)
- [Ingres Access](#) (see page 14)
- [Enterprise Access Products](#) (see page 14)
- [ActiveX Support](#) (see page 15)
- [Interactive Development Environment](#) (see page 15)
- [Complete Development Environment with Visual Tools](#) (see page 16)
- [Interactive Testing and Debugging](#) (see page 19)
- [Report Writing](#) (see page 19)
- [Application Management Utilities](#) (see page 20)
- [Component Sharing](#) (see page 20)
- [Automatic Field Generation](#) (see page 21)
- [Automatic Frame Generation](#) (see page 21)
- [4GL—A Fully Object-Oriented Language](#) (see page 21)
- [Dynamic Applications](#) (see page 22)
- [Active Repository](#) (see page 22)
- [OpenROAD Server](#) (see page 24)
- [eClient Deployment](#) (see page 25)

OpenROAD can help you increase your productivity in many ways with the following features.

Note: For more detailed information about using OpenROAD, see the *User Guide*.

Multiple Platform Support

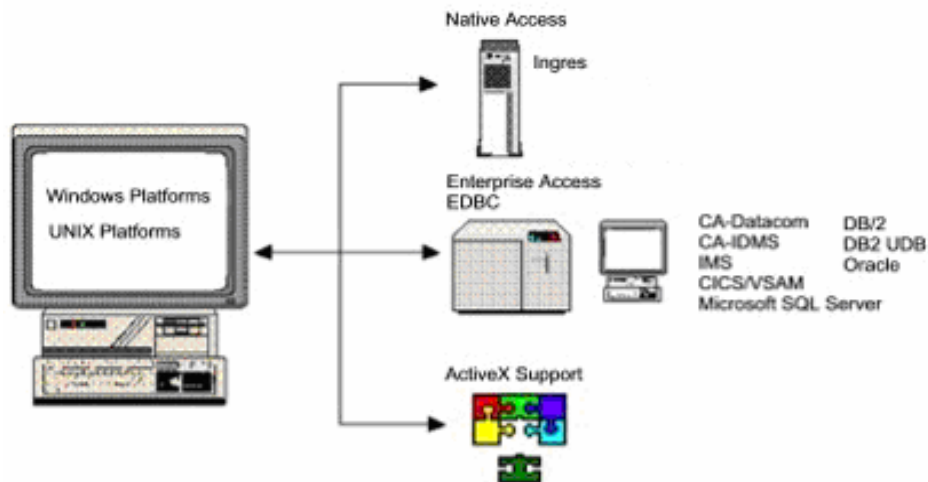
Applications built using OpenROAD are independent of the hardware and software platforms on which you develop them. This gives you the flexibility to write a single application that automatically supports multiple GUIs on Windows and UNIX platforms which can also access multiple databases.

Open Database Access

OpenROAD provides high-performance open database access using a number of Database Management System (DBMS) options:

- Development and runtime access to Ingres and enterprise access products (Oracle, DB2 UDB, and Microsoft SQL Server)
- Runtime access to ODBC products

Accordingly, OpenROAD takes advantage of the capabilities of these connectivity options and provides seamless access to data from desktop, server, mainframe, or network as shown in the following illustration:



Ingres Access

OpenROAD provides robust development access to the Ingres repository so that you can develop front-end applications.

Enterprise Access Products

You can also build and deploy OpenROAD applications against Enterprise Access servers, thereby accessing and updating legacy data in a number of databases using enterprise access products.

You can use the following databases to build and deploy OpenROAD applications:

- Oracle
- DB2 UDB
- Microsoft SQL Server

The following databases can be used currently only for deployment:

- Advantage CA-Datcom/DB Database
- Advantage CA-IDMS/DB Database
- DB2
- IMS
- CICS/VSAM
- RMS

ActiveX Support

OpenROAD supports Microsoft ActiveX technology, which enables applications or software components developed by various independent software vendors to be integrated seamlessly into your OpenROAD application.

For example, you could build multiple custom user interfaces—such as purchasing, accounts payable, accounts receivable, and cost estimation applications—that access the same spreadsheet application for the actual computations. Furthermore, you can avoid the labor-intensive task of creating your own spreadsheet application by using one from an independent vendor and then programming your user interfaces to launch and load the spreadsheet application. This is possible without the end user ever seeing the actual user interface of the spreadsheet application.

Similarly, you can plug ActiveX controls into your application. ActiveX controls are *external controls*, such as charts, grid controls, HTML controls, and web browsers, that are available from independent software vendors. These controls add sophisticated functionality to your applications without requiring additional programming.

Interactive Development Environment

Applications you develop using OpenROAD are largely independent of the DBMS in use, and can be easily ported from one DBMS to another with little or no changes in the underlying code.

Using OpenROAD, you can develop applications without writing and maintaining multiple versions of each application for different computing platforms. You can confidently meet the demands for enterprise-wide, n-tier applications that support hundreds of users in many departments in multiple locations across platforms, even across the intranet or Internet.

This explains the *Open* in OpenROAD-the applications you develop are wide open in terms of user interface, platform, and DBMS-but what about *ROAD*? ROAD stands for Rapid Object Application Development, which means you can develop applications quickly in a fully object-oriented, interactive development environment (IDE) that includes a repository, a workbench, a class browser, a debugger, interactive testing tools, and templates. The objects that you create for one application can be reused by other applications, further speeding up the development process.

Complete Development Environment with Visual Tools

The development environment is intuitive, even if you are a new user, giving you a quick and productive start. Yet it is sophisticated enough to provide the powerful features that advanced developers require.

For example, the Frame Editor lets you quickly and easily paint the user interface for an application and handle much of the programming logic behind the interface using event-driven, object-oriented code written in the OpenROAD 4GL language.

Advanced programmers can take advantage of additional features, such as:

- Using 4GL to its full extent to create applications capable of dynamically generating frames at runtime
- Exploiting the eClient, mClient, and OpenROAD Server technologies

The OpenROAD IDE includes the Application Workbench and visual editors that enable you to design the frames, menus, and other user interface components for your applications using point-and-click, drag-and-drop techniques. For immediate feedback on your design as it progresses, you can click Tools, Simulate to start a simulation.

Application Workbench

The OpenROAD Application Workbench is flexible and intuitive, providing quick access to key functionality, including the following modes:

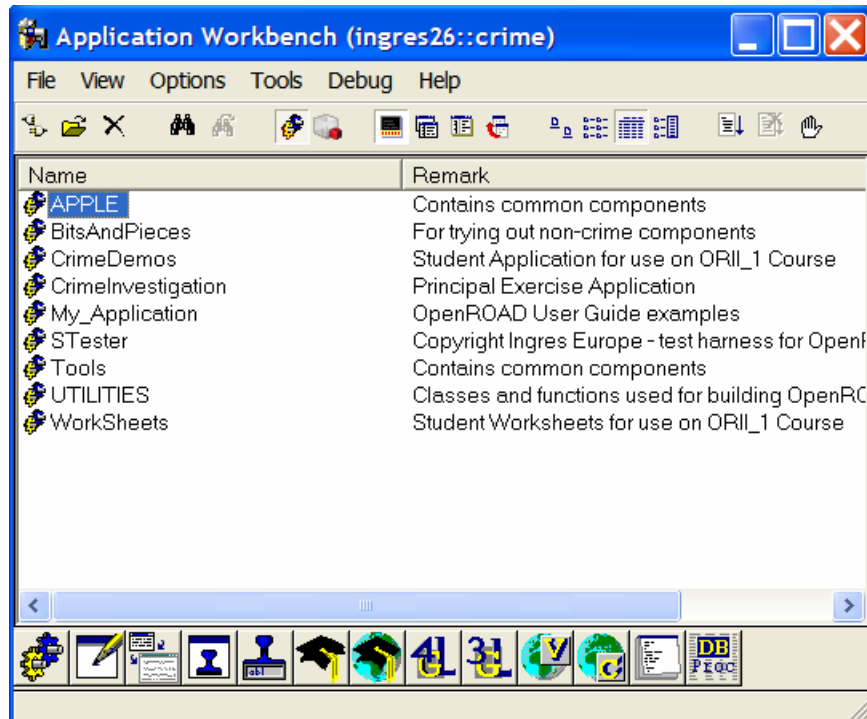
Application Mode

Lets you view, edit, and manipulate existing applications in your repository, as well as create new ones

Component Mode

Lets you view, edit, and manipulate the components of a specified application, as well as create new components

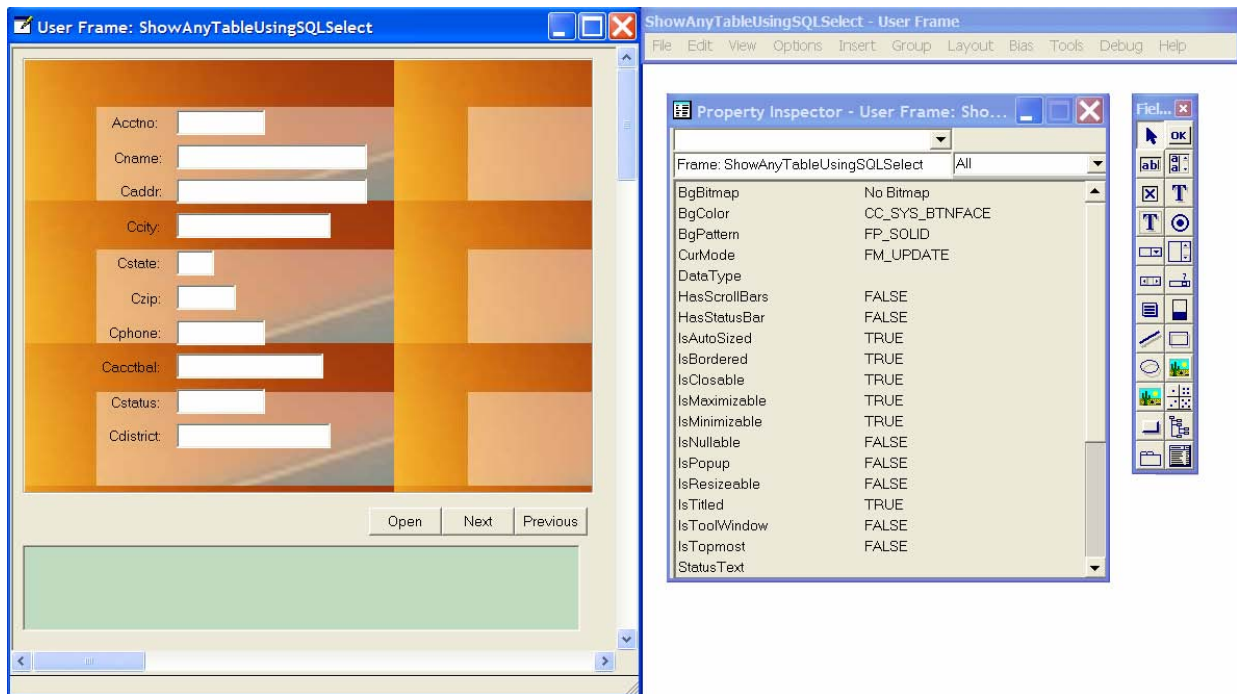
You can specify how the Application Workbench displays applications, components, and models-as icons only, list views, detailed list views, or list views with properties. The following illustration shows an example of the Application Workbench in Application Mode:



Editors

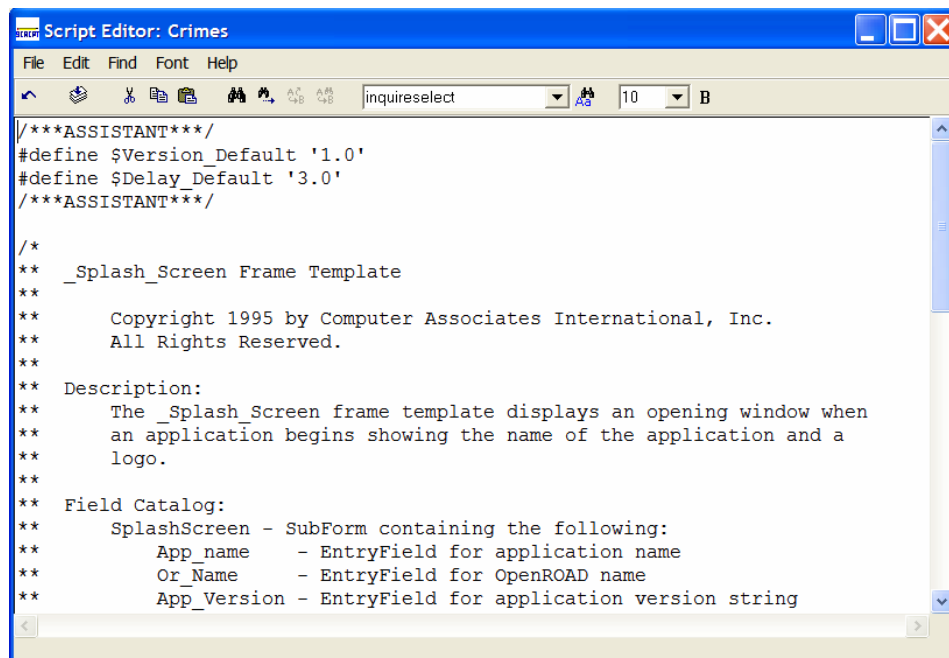
In Application or Component modes, a component palette provides quick access to all OpenROAD component editors. Some are visual editors, like the Frame and Frame Template Editors, which have their own tool palettes and menu bars. Editors also display an accompanying Property Inspector that lets you quickly and easily specify the properties for each component and each field or control in the component. You can optionally display a field tree that visually depicts the entire field hierarchy for a component.

The following illustration is an example of the Frame Editor with its accompanying Property Inspector, Field Palette, and menu bar:



In addition to the visual editors, OpenROAD provides a host of editors specifically tailored to deal with different aspects of your application in the most straightforward manner possible. These range from simple dialogs for declaring global variables and constants to a Script Editor for writing your code.

The following illustration is an example of the Script Editor:



```
Script Editor: Crimes
File Edit Find Font Help
inquireselect 10 B

/****ASSISTANT****/
#define $Version_Default '1.0'
#define $Delay_Default '3.0'
/****ASSISTANT****/

/*
**  _Splash_Screen Frame Template
**
**      Copyright 1995 by Computer Associates International, Inc.
**      All Rights Reserved.
**
**  Description:
**      The _Splash_Screen frame template displays an opening window when
**      an application begins showing the name of the application and a
**      logo.
**
**  Field Catalog:
**      SplashScreen - SubForm containing the following:
**          App_name - EntryField for application name
**          Or_Name - EntryField for OpenROAD name
**          App_Version - EntryField for application version string

```

Interactive Testing and Debugging

The development environment enables you to run an application, or any of its components, at any point while you are designing it. Development and execution modes are fully integrated; you move between editing an application prototype and running it with a mouse click.

This feature promotes fast prototyping and quick feedback when you make changes to the application, and enables you to test and debug your applications efficiently using the interactive debugger built into the development environment.

Report Writing

OpenROAD includes the Reporter, a repository-based report writer that lets you display reports online or print them. The Reporter features a graphic designer with open API capability, a graphical layout editor, and a query editor.

Application Management Utilities

In addition to the many tools integrated into the development environment, OpenROAD provides several application management tools accessible from the Application Workbench toolbar, its Tools menu, or the command line interface. These tools let you deal with the application as a whole and include capabilities such as creating and running an application image, importing and exporting applications, and compiling them.

Component Sharing

The following features enable components to be shared among a development team, increasing your individual and group productivity.

Team Development

The OpenROAD development environment is designed to be shared by a team of developers, each member working on individual components without fear of conflicting with the work of other members.

Frame and Field Templates

Another key productivity feature is the ability to define and use frame and field templates that you can share within a single application and among several applications. Frame and field templates contribute to the development of your application by letting you define user interface standards, including the visual components and the code that make them operational.

Component Libraries

To promote component reusability, you can set up your OpenROAD application to include other applications. This way, you can develop libraries of components, such as user classes and 4GL procedures, that are generic and intended for use by many applications. This is the concept behind the libraries provided with OpenROAD.

Open Architecture

You can also access database and 3GL procedures, allowing you to share components developed independently of OpenROAD. You simply declare the procedures in your application and call them, much as you would call a 4GL procedure.

Automatic Field Generation

To enhance the look-and-feel of your application frames, you can use an extended set of field templates contained in a number of OpenROAD libraries. You can add various types of fields-such as spin controls and many desktop tools-with a few mouse clicks. You can also create fields dynamically using assistant technology, which steps you through the field customization process.

Automatic Frame Generation

OpenROAD enables you to generate frames using its assistant-based frame templates that solve a wide variety of common business problems. All frames that you generate using this feature are fully operational, including completely functional database access with update, delete, and insert capability. No frames require you to write a single line of code. The generated frames are completely customizable, as are the generation procedures.

4GL—A Fully Object-Oriented Language

OpenROAD gives you the power of a single, high-level, graphical application development language for all platforms, user interfaces, and databases. Therefore, a key part of the OpenROAD development environment is 4GL, an object-oriented programming (OOP) language that lets you create and reuse objects across applications and platforms.

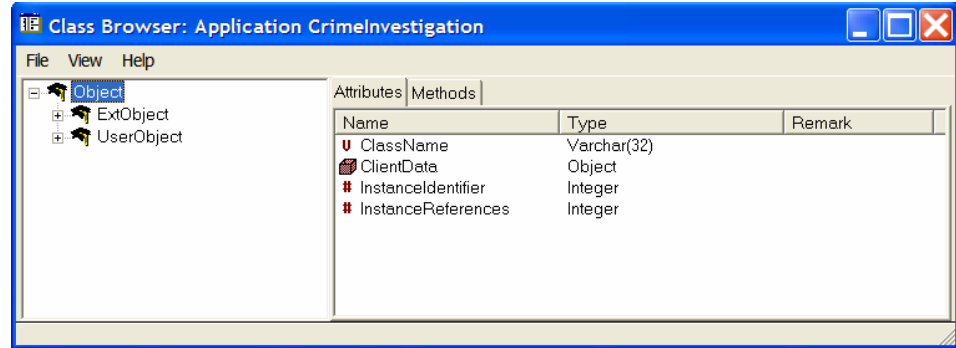
System Classes and User Classes

OpenROAD includes more than 100 predefined system classes and more than 200 methods to simplify the task of designing enterprise applications. These include classes that provide access to native Windows controls such as tab folders, tree views, and list views. Class definitions are integrated in the OpenROAD development environment, so that you have the power of OOP without requiring an in-depth understanding of object-oriented theory.

You can also define your own user classes with OpenROAD. User-defined classes exhibit the same object-oriented features as system classes. These features-inheritance, abstraction, encapsulation, and polymorphism-enable you to reuse components and maximize productivity.

Class Browser

The OpenROAD Class Browser enables you to browse the methods and properties of external class objects, system class objects, and user class objects. The following is an illustration of the Class Browser:



Dynamic Applications

OpenROAD with its template architecture uniquely empowers you to build *dynamic applications*-applications that shape themselves based on what actions are initiated by the application, the data values, or the end user. Dynamic applications are also *scalable*-they respond dynamically to variations in data type complexity and database size.

In the visual development environment, you can build different functionality into an application based on who the end user is or which mode the end user is operating in. That is, every field and menu item in OpenROAD has a specified *bias*, such as dimmed, changeable, or invisible, which determines whether it is available to the end user and what events it can receive. Likewise, every frame has specified sets of biases, called *modes*, which determine whether it is read-only, resizable, and so on. Additionally, you can define new fields, user classes, and 4GL expressions at runtime, or create new frames at runtime based on user on-demand query options and sort preferences.

The dynamic nature of OpenROAD applications increases end-user productivity and, more importantly, reduces the burden of coding for your application development team. You specify details only once and let OpenROAD generate the application for you.

Active Repository

OpenROAD uses whatever database you choose as a central storage area, or *repository*, for storing and tracking all components of an application-everything from classes, procedures, and methods to frames, menus, and fields.

Component Reusability

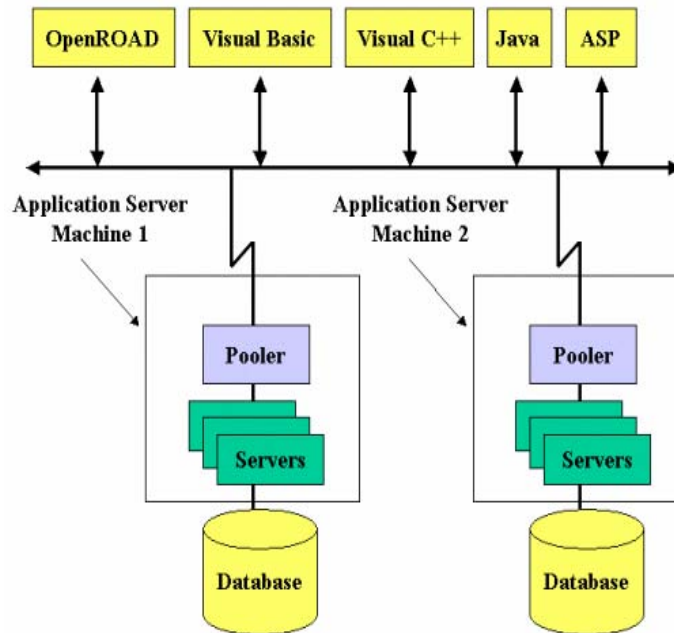
The repository is seamlessly integrated into the development environment, promoting the reuse of common components and objects in a single application or across multiple applications. It provides a cross-reference facility you can use to determine component dependencies within an application.

Management Facilities

Because enterprise-wide, n-tier applications are complex and have long application lifetimes, and because team development is equally challenging, OpenROAD provides sophisticated management facilities for your applications. The repository is shared by your development team, and change management and version control facilities let you track and manage many application components-and all versions of these components-to avoid any conflict during development.

OpenROAD Server

The *OpenROAD Server* is an object-relational application server—a powerful, object-oriented environment that equips you to build and deploy robust business objects in a number of environments. The OpenROAD Server is a collection of objects and services that support n-tiered applications written in the OpenROAD 4GL language. OpenROAD Server facilities allow 4GL business logic to be shared by a wide variety of client programming environments and languages, including web server scripting languages, as shown in the following illustration:



The OpenROAD Server lets customers and Value Added Resellers (VARs) build complete distributed eBusiness solutions.

Fixed Signature, Dynamic Data

The OpenROAD Server uses a generalized method signature so that you do not need to construct and deploy separate proxy/stub code for customized Component Object Model (COM) signatures for each OpenROAD 4GL procedure. The data passed through this signature is dynamically mapped to the actual signature of the called 4GL procedure.

This generalized signature provides an easy-to-deploy baseline facility, which is the only interface directly supported by the OpenROAD Server. Developers may also build their own customized COM objects and method signatures to wrap the underlying interface.

Structured Data

OpenROAD supports structured data in the form of user classes and user class arrays. A user class is analogous to a row of data with a well-defined set of column types, and a user class array is analogous to multiple rows of data with the same column types.

OpenROAD user classes and user class arrays can be nested to any number of levels, allowing the construction of very complex hierarchical data structures to be passed as parameters to remote 4GL procedures.

eClient Deployment

The OpenROAD eClient lets you deploy an OpenROAD application in a web browser, permitting the application to access an OpenROAD Server without having to install OpenROAD on the client machine.

The OpenROAD eClient is distinct from the original OpenROAD client. The original client continues to be used for typical deployment of OpenROAD applications. The two environments can coexist on the same machine. The OpenROAD eClient can also coexist with OpenROAD 2006 and OpenROAD 4.1 client runtimes.

Note: For requirements on deploying browser-based applications using the eClient, see the readme.

eClient Runtime Cabinet File

The OpenROAD eClient runtime is contained in a single cabinet (CAB) file that can be hosted on your web server and referenced by your HTML. Ingres Corporation digitally signs this CAB file, and automatically downloads and installs the runtime if it has not already been installed. Users need only to click OK on the common digital signature trust dialog in Internet Explorer. No other installation or configuration is required.

eClient Application and Library Cabinet Files

Each OpenROAD eClient application must be packaged into a CAB file to enable it to automatically download and install to client machines. You can also package the shared 4GL libraries into CAB files for automatic installation. You can use the eClient packaging tools to create eClient CAB files. For more information, see [How You Can Deploy Web Applications](#) (see page 26)

How You Can Deploy Web Applications

You deploy applications to the Web by using the MakeCab utility. To deploy an application to the Web, you would perform the following steps.

1. Use the MakeCab utility to create an HTML page that references the OpenROAD eClient runtime and your application, using the HTML OBJECT tags.
2. Install the HTML page, the eClient runtime CAB, and your application CAB on your web server.
3. The end user then references the HTML page from an Internet Explorer browser, and the application launches.

The administrator for the web server can provide new versions of the application, the application libraries, or the OpenROAD eClient runtime by simply placing updated CAB files on the web server, and then updating the version number referenced in the OBJECT tags on the HTML page. The updated components are automatically downloaded and installed on users' machines the next time the component is referenced.

Note: For more information on deploying web applications using the OpenROAD eClient, see the *User Guide*.

Chapter 3: Installing and Configuring OpenROAD on Windows

This section contains the following topics:

[Upgrading from Earlier OpenROAD Releases](#) (see page 27)

[Prepare for Installation—Shut Down Ingres](#) (see page 28)

[Install OpenROAD on Windows](#) (see page 28)

This chapter details how to install and configure OpenROAD on Microsoft Windows platforms. The following options are available:

| Option | Purpose |
|-----------------|--|
| Development | Develop OpenROAD applications |
| Runtime | Deploy and execute an existing OpenROAD application or one that has been transformed from ABF |
| OpenROAD Server | Deploy a distributed OpenROAD application |
| eClient | Deploy an OpenROAD application on a web server for access from a web browser |
| mClient | Install the mClient runtime files to your system for use with the Windows Mobile-Based Pocket PC 2003 Operating System |
| Documentation | Learn about OpenROAD development |

Important! Review the readme before you begin installation. The Windows readme file is named `readme_OpenROAD.html`.

Upgrading from Earlier OpenROAD Releases

The installer will automatically upgrade from previously installed versions of OpenROAD.

Prepare for Installation—Shut Down Ingres

If Ingres is currently running on your machine, you must shut it down before installing OpenROAD.

To shut down Ingres

1. Click Start, Programs, Ingres II, Ingres Service Manager.
Ingres Service Manager opens.
2. Click Stop to shut down Ingres.
3. When Ingres is completely stopped, close the Ingres Service Manager.

Install OpenROAD on Windows

You install OpenROAD on Windows platforms from the installation media using the installation wizard.

To install OpenROAD on Windows

1. Insert the OpenROAD installation media in the CD-ROM drive of your computer.

The installer should start automatically. If it does not, click Start, Run.

The Run dialog appears.

2. In the Open field, enter the following:

```
cdrom_drive: \setup.exe
```

cdrom_drive

Represents the drive letter of the CD-ROM drive. For example, d: \setup.exe.

3. Click OK.
4. Follow the instructions in the installation wizard.
5. Configure the virtual nodes required to access your development repository.

Note: For detailed instructions on setting up virtual nodes, see the Ingres documentation.

Chapter 4: Installing and Configuring OpenROAD on UNIX

This section contains the following topics:

[Install OpenROAD on UNIX](#) (see page 29)

[MainWin System Core Subsystem](#) (see page 36)

[How an Administrator Can Administer the MainWin System Core](#) (see page 39)

[How You Can Configure COM and DCOM](#) (see page 52)

This chapter details how to install and configure OpenROAD on a UNIX platform.

Install OpenROAD on UNIX

OpenROAD installs either as a new client installation, or into an existing Ingres installation. For instructions, see the appropriate section, following.

Before starting the installation, thoroughly review the information in this chapter. It contains pertinent information regarding Registry upgrades and Distributed Component Object Model (DCOM) support.

Note: You must have superuser (root) privileges to complete the installation.

Install OpenROAD as a New Installation

You can install OpenROAD as a new client installation using the following procedure.

To start the installation

1. Log in as root.
2. Mount the CD-ROM drive and insert the OpenROAD installation media in the drive.

3. Change the directory to the CD-ROM drive and then enter the following command:

```
#sh ./install.sh
```
4. Select PackageInstall from the menu, and then select one of the following packages:
 - OpenROAD Runtime
 - OpenROAD Runtime and Development
 - Then respond appropriately to the prompts to complete the installation.

Install OpenROAD into an Existing Ingres Installation

You must use ORINSTALL.ING to install OpenROAD into an existing Ingres installation.

Note: The registry file created prior to OpenROAD 4.1 Service Pack 3 is incompatible with this release. The installer can upgrade your existing registry file to the new format, or you can perform the upgrade manually after completing the installation. For more information on upgrading your registry, see How You Can Update Your Registry on UNIX in this chapter.

The installer can also install the MainWin System Core (MSC) subsystem during the installation process, or you can follow the instructions to install the MSC subsystem later. For more information on installing the MSC subsystem, see Install MSC in Interactive Mode *or* Install MSC in Non-interactive Mode in this chapter.

Prepare for Installation

Before you install OpenROAD into an existing Ingres installation, you must perform the following procedure.

To prepare for installation

1. Shut down your existing Ingres installation.
2. Shut down the OpenROAD Server.
3. Perform a backup of your Ingres installation.
4. Set the II_SYSTEM environment variable to the existing installation you want to upgrade.

5. Decide which OpenROAD package you want to install:

- ordev for Runtime and Development
- orrun for Runtime only

Note: The ordev and orrun packages do not include the Ingres Net 2.6 SP2 components. The installer does not allow an upgrade because you are installing into an existing Ingres installation. To upgrade your Ingres Net in this situation, you must acquire and use an Ingres installer.

6. Decide if you want to let the installer install the MSC subsystem.

If you want the installer to install the MSC, ensure that you have superuser (root) privileges for this installation step. When the installer is ready to install the MSC subsystem, it prompts for the root password.

7. Decide if you want to let the installer upgrade the existing registry file.

Note: Before you make your decision, see How You Can Update Your Registry on UNIX in this chapter. We recommend that you let the installer create a new registry rather than upgrade an existing registry to the new format.

If you plan to upgrade an existing registry file at the time of installation, ensure that you have set the MWREGISTRY to your registry file using the full path name. This applies even if you are using the default name for the registry file.

- If you are upgrading from OpenROAD 4.1/0003 or from OpenROAD 4.1/0109 to this release and you are using the default registry name, set the MWREGISTRY as follows:

```
%setenv MWREGISTRY $HOME/windows/registry.bin
```

- If you are upgrading from OpenROAD 4.1/0302 (SP2) release to this release and you are using the default registry name, set up your OpenROAD runtime environment, and then set the MWREGISTRY as follows:

```
%setenv MWREGISTRY $HOME/windows/hk1m_$MWOS.bin
```

- If you are using a user-defined registry file name, then set MWREGISTRY to your registry file name with full path as follows:

```
%setenv MWREGISTRY /full_path/existing_registry_name
```

Note: The installer skips the registry conversion step if the MWREGISTRY environment variable is not set. In addition, an X Server must be available when an OpenROAD application performs the conversion.

Start the Installation

You install OpenROAD from the installation media using ORINSTALL.ING.

To start the installation

Mount the CD-ROM drive and insert the OpenROAD installation media into the drive. Then, enter the following commands:

```
%cd $II_SYSTEM/ingres
%tar -xvf /cdrom/platform_string/orr5.tar ORINSTALL.ING
%./ORINSTALL.ING /cdrom/platform_string/orr5.tar {ordev | orrun}
```

platform_string

Specifies a unique string for each platform. Use one of the following strings:

| Platform | String |
|-----------------|---------------|
| Solaris | su4_us5 |
| AIX | rs4_us5 |
| HP | hpb_us5 |
| Linux | int_lnx |

During installation, the installer prompts you to confirm the upgrade of an existing registry to the new format. It also prompts you to confirm that you actually want to install the MSC subsystem.

How You Can Perform Post-installation Customization

After installing OpenROAD, you can customize the runtime environment by following these steps.

1. The MSC subsystem is set up to run as a system service, `S99mwcore_services`, as a daemon process at system boot time.

If you have previously set up a daemon process `S99orspo` to start the OpenROAD Server at system boot, ensure that the OpenROAD Server Pooler daemon runs *after* the MSC subsystem daemon process.

2. Modify your OpenROAD environment setup script:
 - If you have a previous OpenROAD 4.1 installation and your environment setup script sets up the `MWREGISTRY` environment variable, you must remove this environment variable; it is no longer required.
 - If your search path directory contains `ORbin`, remove the `ORbin` directory.
 - If you have a previous OpenROAD 4.1 installation and `MWHOME` is set to `$II_SYSTEM/ingres/mainwin` in your environment setup script, change `MWHOME` to `$II_SYSTEM/ingres/mainwin/mw` in your setup script.
3. If you have previously set up a daemon process `S99orspo`, to start the OpenROAD Server at system boot, modify the script to reflect the new runtime environment setup as described in the previous bullet.

How You Can Update Your Registry on UNIX

The registry file created prior to OpenROAD 4.1 SP3 is incompatible with this release. After the installation is complete, do *one* of the following:

- Create a new registry file
- Convert an existing registry file to the new format

Note: We recommend that you create a new registry file for this release. For instructions, see the appropriate section, following.

Create a New Registry File

There is no special setup procedure to create a new registry file. When you launch an OpenROAD application, the MainWin runtime creates a new registry automatically, if needed. To view or modify the registry information, use the MainWin supplied utility, `regedit`.

If you create a new registry, you must re-register the OpenROAD Server.

Note: The `mwregedit -new` command used in previous releases is obsolete. If your registry file gets corrupted, or you need to recreate a new registry, delete the old registry file, and then start the Application Workbench or use `w4glrun` to run any OpenROAD application; a new registry file with system keys is recreated automatically. However, if you are using the OpenROAD Server, you must re-register it.

To re-register the OpenROAD Server

Enter the following commands:

```
%setenv USERNAME host_login
%setenv COMPUTERNAME host_name
%w4glrun asreg.img
```

host_login

Specifies a valid host login, such as "ingres"

host_name

Specifies the name of your UNIX machine

If the registry did not previously exist, this command creates a new registry file in `$HOME/.mw/hkcu.bin` and populates it with system keys and the appropriate OpenROAD Server keys.

Upgrade an Existing Registry File from OpenROAD 4.1

If you are upgrading from OpenROAD 4.1/0003 or 4.1/0109, you must upgrade the registry file to complete the installation. You must run the following two upgrade utilities:

mwregconv

Converts your existing registry file to a new format

ru41sp2.img

Updates existing OpenROAD Server keys

Note: Before running the upgrade utilities, ensure that your OpenROAD runtime environment is set up correctly.

To update the default registry file

Run the following upgrade procedures:

```
%mwregconv $HOME/windows/registry.bin
%w4glrun ru41sp2.img
```

To upgrade a user-defined registry file

Run the following upgrade procedures:

```
%mwregconv /full_path/existing_registry_file_name  
%w4glrun ru41sp2.img
```

full_path

Specifies the full path to the existing registry file

existing_registry_file_name

Specifies the name of the existing registry file

Note: Do not set the MWREGISTRY environment variable to the old registry file name because it is passed to mwregconv as a command line parameter.

Upgrade an Existing Registry File from OpenROAD 4.1 Service Pack 2

If you are upgrading from OpenROAD 4.1/0302, you must upgrade the registry file to complete the installation using the upgrade utility, orregconv.

The *orregconv utility* is a shell script that exports the OpenROAD Server root keys from an existing registry file and imports them into the new registry file. If necessary, you can modify this script to export and import other registry keys. Orregconv is located in the \$II_SYSTEM/ingres/bin directory.

To upgrade the default registry file

Run the following upgrade procedure:

```
%orregconv $HOME/windows/hk1m_$MWOS.bin:$HOME/windows/hkcu.bin
```

To upgrade a user-defined registry file

Run the following upgrade procedures:

```
%orregconv /full_path/existing_registry_name:$HOME/windows/hkcu.bin
```

full_path

Specifies the full path to the existing registry file

existing_registry_file_name

Specifies the name of the existing registry file

Note: Do not set the MWREGISTRY environment variable to the old registry file name because it is passed to orregconv as a command line parameter.

MainWin System Core Subsystem

The *MainWin System Core (MSC) subsystem* is a runtime component that supplies services to Visual MainWin-ported applications, such as registry access and COM or DCOM invocation (RPCSS). The MSC subsystem provides background information needed for post-installation customization.

The advantages of using the MSC subsystem include:

Services availability

The MSC services are always running. They are available to all OpenROAD applications.

Multi-user support

The MSC subsystem supports multiple users. This means that different users have access to the same registry, identical to the Windows registry. For example:

- Other users can use COM servers registered by one user.
- Printers set up on a machine by one user are available to all users.
- COM and DCOM support

The MSC subsystem supports multiple users and provides support for secure COM or DCOM interaction with other COM or DCOM hosts on both UNIX and Windows.

mwadm—MSC Administrator

The administration tool, *mwadm*, provides administrative support for the MSC services. The MSC administrator, designated when installing MSC, has special privileges including the exclusive right to execute certain Visual MainWin system operations, for example, starting and stopping MSC services.

Note: Only the MSC administrator can execute operations to modify system data, such as registry entries that control the local machine settings.

Install MSC in Interactive Mode

Executing the installation program `install.mwcore` with no parameter launches the MSC installation in interactive mode.

Note: Only a superuser (root) can install the MSC subsystem because the installation procedures include setting up the system services (daemon process) at system boot, making the MSC subsystem available to any application that need its services.

To start MSC installation in interactive mode

1. Enter the following commands:

```
%su root  
%cd $II_SYSTEM/ingres/mainwin/MS  
%./install.mwcore
```

2. Enter the path to installation directory:

```
default=/opt/mainsoft/mwcore:
```

3. Enter **2** to create a directory.

4. Enter the name of the user (for example, ingres) who you want to be the MSC administrator on this host.

A transcript of this setup is saved in the following file:

```
/opt/mainsoft.test/mwcoredata/node_name/logs/mwcore_install.log.1
```

Install MSC in Non-interactive Mode

To install MSC in non-interactive (“silent”) mode, you must first configure the response file to use the `install.mwcore` utility, and then run the installation commands.

To configure the response file to use the `install.mwcore` utility

1. Edit the response file.

This utility takes the following optional parameters:

-silent

Specifies that the utility should run in non-interactive mode

-options *response_file*

Specifies the name of the file (*response_file*), the response file for the `install.mwcore` utility. A sample file, named `msc_silent`, is located in the directory `$II_SYSTEM/ingres/mainwin/MS`.

2. Using the `msc_silent` file as a template, make any necessary changes to the settings.

For example, the following line specifies the MSC installation directory, where the default is `"/opt/mainsoft,"` although you can change this to another directory.

```
-W dest_dir.mwHomeDir="/opt/mainsoft"
```

The following line specifies the MSC administrator, where the default administrator name is `"ingres."`

```
-W msc_admin_setup.admin="ingres"
```

You can designate another user to be the MSC administrator by changing `"ingres"` to another user name.

3. Log in as root.
4. Enter the following commands:

```
%su root
#cd $II_SYSTEM/ingres/mainwin/MS
#./install.mwcore -silent -options msc_silent
```

5. Enter the root password when prompted.

Uninstall MSC

There are two ways to uninstall the MSC:

- Using unset.mwcore
- Using uninstall.mwcore

In the following procedures:

MSC_install_dir

Specifies the full path to the MSC installation

To uninstall the MSC subsystem using unset.mwcore

Enter the following commands:

```
%mwadm stop
%cd MSC_install_dir/mw/bin/unset.mwcore
%rm -f MSC_install_dir
```

To uninstall the MSC subsystem using uninstall.mwcore

Enter the following commands:

```
%mwadm stop
%MSC_install_dir/mw/bin/uninstall.mwcore
```

How an Administrator Can Administer the MainWin System Core

To administer MSC services on every UNIX host, the MSC administrator uses the mwadm utility. This utility is included with the MSC installation. It is a stand-alone utility, which does not require a Visual MainWin runtime environment (that is, MWHOME) to be set. The mwadm utility supports high-level commands, which in turn implement administration functions for specific areas, including security and registry.

How You Can Display mwadm Usage Information

If you invoke mwadm with no parameters, it displays the usage information for the utility as shown:

```
%mwadm
Usage: mwadm command [parameters]
```

Following is the list of commands supported by the mwadm utility:

| | |
|---------|--|
| version | displays the version number of the MSC. |
| help | displays help on using mwadm and its commands. |
| stop | stops all MSC services in an orderly manner. |
| start | manually starts all MSC services. |
| status | provides vital signs of the MSC services. |
| log | administers the MSC logs. |
| sec | administers MSC DCOM security settings. |
| reg | administers the MSC registry service. |

The mwadm utility also supports the following commands not listed in the usage section:

chadm

Changes the MSC administrator

port

Administers the MSC RPCSS port allocation

MSC Administrator Commands

This section describes the syntax, parameters, descriptions, and examples for each of the mwadm commands.

chadm Command—Change the MSC Administrator

This command changes the MSC administrator. When you enter the new user name, you are prompted to confirm it.

This command has the following format:

```
mwadm chadm [new_administrator_name]
```

new_administrator_name

Specifies the user name of the new administrator

Examples—chadm command:

Change the MSC administrator to myuser:

```
% mwadm chadm myuser
% Please re-enter user name:myuser
```

Change the MSC administrator with a prompt to enter a new user name:

```
% mwadm chadm
% Please enter the user name of the new MSC administrator:myuser
% Please re-enter user name:myuser
```

help Command—Display Help

The help command explains the purpose of the mwadm utility and lists all available mwadm commands along with a short description of each. You can also get detailed help for a particular mwadm command by specifying the command name as a parameter to the help command.

This command has the following format:

```
mwadm help [command]
```

command

Specifies the command you want help for

Examples—help command:

Display help for using the mwadm utility:

```
% mwadm help
```

Display help for using the mwadm start command:

```
% mwadm help start
```

log Command—Administer MSC Logs

This command lets you administer the logs created and maintained by the MSC. You can limit the size of the logs, set the level of detail that should be registered by the MSC services into these logs, or display the logs' contents.

Note: Only the MSC administrator can execute the log command.

This command has the following format:

```
mwadm log [-view | -level {1|2} | -maxsize kb | -basesize kb ] logname ]
```

-view

Prints the contents of the log to the standard output

-level {1|2}

Determines the level of detail for the messages that will be logged into the log files. You must specify one of the following types of logging:

1

Specifies basic logging

2

Specifies detailed logging

Default: 1

-maxsize *kb*

Limits the size of the log file to the number of kilobytes specified in *kb*. The MSC monitors and truncates the size of the log file when it reaches maxsize. The log is truncated to the size set by the -basesize option.

-basesize *kb*

Sets the size of the truncated log to the number of kilobytes specified in *kb* when the log file reaches maxsize.

Note: The value set with the -basesize option must be smaller than the value set with the -maxsize option.

logname

Specifies the name of the log, which must be one of the following logs that the MSC maintains:

regss

Stores messages generated by the registry server

rpcss

Stores messages generated by the RPCSS server

Examples—log command:

Display the settings for the registry services log:

```
% mwadm log regss
```

Set the level of log details to 2 (detailed) for the REGSS log:

```
% mwadm log -level 2 regss
```

Print the contents of the REGSS log to the standard output:

```
% mwadm log -view regss
```

Limit the size for the RPCSS log to 100 KB:

```
% mwadm log -maxsize 100 rpcss
```

Set the base size for the RPCSS log to 20 KB:

```
% mwadm log -basesize 20 rpcss
```

Display the list of the log settings in effect for a specific MSC log *logname* (execute the log command with no options):

```
% mwadm log logname
```

reg Command—Administer the MSC Registry

This command lets you administer various aspects of the MSC registry services.

This command has the following format:

```
mwadm reg -quota {-d [sizekb] | -u user [sizekb] | -r user | -a}
mwadm reg -chmod [-r] mode key
mwadm reg -chown [-r] user key
mwadm reg -chgrp [-r] group key
mwadm reg -ls key
mwadm reg -ps
mwadm reg -backup filename
mwadm reg -auto_backup [ -on | -off | -n #versions -i #hours]
mwadm reg -restore [ filename | -v # ]
```

-quota {-d [*sizekb*] | -u *user* [*sizekb*] | -r *user* | -a}

Sets or displays a quota for data that can be allocated under the HKEY_CURRENT_USER registry entry. The quota size is measured in kilobytes (KB). You must specify one of the following settings:

-d [*sizekb*]

Displays or sets the default user quota

-u *user* [*sizekb*]

Displays or sets the quota for a specific user

-r *user*

Removes a specific quota (set with the -u option) for a specific user, whereby the user acquires the default quota again

-a

Displays all the quota settings

-chmod [-r] *mode key*

Changes a key's protection level in UNIX chmod format

Note: Only the key's owner or the MSC administrator can execute the chmod option. This option has no effect on any of the HKEY_CURRENT_USER registry keys.

-r

Causes the change to occur recursively

mode

Represents the new mode in octal format

key

Represents the full path to the key

-chown [-r] *user key*

Changes a key's owner in UNIX chown format

Note: Only the MSC administrator can execute this option. This option has no effect on any of the HKEY_CURRENT_USER registry keys.

-r

Causes the changes to occur recursively

user

Represents the new owner's name

key

Represents the full path to the key

-chgrp [-r] *group key*

Changes a key's group in UNIX chgrp format

Note: Only the MSC administrator can execute this option to change the key to any group. In addition, owners can use this option to change one of their keys to a different group, provided they belong to the group. This option has no effect on any of the HKEY_CURRENT_USER registry keys.

-r

Causes changes to occur recursively

group

Represents the new group's name

key

Represents the full path to the key

-ls *key*

Prints information about a specific key using the UNIX “ls -l” command

Note: This option has no effect on any of the HKEY_CURRENT_USER registry keys.

key

Represents the full path to the key

-ps

Lists all the processes currently connected to the registry service. For each connected process, it lists the process ID, user ID, and the path to the HKEY_CURRENT_USER file.

-backup *filename*

Saves the registry database (excluding the HKEY_CURRENT_USER) to a specified file. There is no need to stop the system services to use this option.

Note: Only the MSC administrator can execute this option.

filename

Represents the location of the file to be saved. In case the specified file already exists, it will be overwritten.

-auto_backup [-on | -off | -n *#versions* -i *#hours_interval*]

Sets or displays the automatic backup settings of the registry. When on, the automatic backup periodically backs up the registry database (excluding HKEY_CURRENT_USER). Later, you can restore the backed up versions using the -restore option.

Note: Only the MSC administrator can execute this option.

-on

Turns on the automatic backup

Default: on

-off

Turns off the automatic backup

-n *#versions*

Sets the number of backup versions to be stored

Default: 1

-i *#hours_interval*

Time (in hours) between backups

Default: 24

-restore [*filename* | -v #]

Restores the registry database from a backup. If automatic backup is on, then calling the -restore option with no parameters displays a list of the backup versions that were saved by the automatic backup. Each version is identified by a number (1, 2, 3, and so on) and by the date on which the registry database was backed up. You can restore a specific version by identifying its number using the -v option.

Note: Only the MSC administrator can execute this option. The restore operation requires the system services to be shut down.

filename

Represents the location of a backup file previously saved using the -backup option. Files backed up on one host cannot be restored on another host.

-v #

Identifies a specific backup version (#) previously backed up by automatic backup

Examples—reg command:

Display a list of all quota values including the default:

```
% mwadm reg -quota -a
```

Display the quota for the user joe:

```
% mwadm reg -quota -u joe
```

Display the default quota:

```
% mwadm reg -quota -d
```

Set the quota for the user joe to 280 KB:

```
% mwadm reg -quota -u joe 280
```

Set the default quota to 200 KB:

```
% mwadm reg -quota -d 200
```

Change the owner for a specific key:

```
% mwadm reg -chown root 'HKEY_LOCAL_MACHINE\\Software\\Microsoft'
```

Back up the registry data (excluding the HKEY_CURRENT_USER data) to a file:

```
% mwadm reg -backup /backups/regbu-28-oct-06
```

Restore a specific version backed up by automatic backup:

```
% mwadm reg -restore 2
```

sec Command—Administer MSC Security

This command administers MSC security settings affecting DCOM communication between the local host and any other UNIX or Windows host.

This command has the following format:

```
mwadm sec -passwd  
mwadm sec -type [unix | win]  
mwadm sec -port [port_id]  
mwadm sec -domain_name [domain_name]  
mwadm sec -domain_server [domain_server]
```

-passwd

Interactively sets the DCOM security activation password. To facilitate secure DCOM communication between the machines set up with MSC services, all machines must be configured with an identical DCOM security activation password.

-type [unix | win]

Displays or sets the security authentication type. When called without parameters, this command displays the security authentication type currently in effect. Options for the security authentication type are:

unix

Specifies that the MSC security mechanism should rely on UNIX security for authentication services. This authentication mode does not support DCOM communication with a Windows machine.

win

Specifies that the MSC security mechanism should rely on a Windows domain server for authentication services. You must use this mode when you expect DCOM communication to take place between Windows and UNIX hosts.

Default: unix

-port [*port_id*]

Displays or sets the port number used for negotiating authentication information with the remote MainWin utility *remotesa*. This option is only relevant when the security type is set to win.

Default: 667

-domain_name [*domain_name*]

Displays or sets the domain name used by the MSC security mechanism when using a Windows domain server for authentication services

Note: This option is only relevant when the security type is set to win.

-domain_server [*domain_server*]

Displays or sets the domain server used by the MSC security mechanism when using a Windows domain server for authentication services. The *domain_server* can be a host name (recognized by the local UNIX machine) or an IP address.

Note: This option is relevant only when the security type is set to win.

Examples—sec command:

Display the list of all security settings in effect (execute the sec command without no options):

```
% mwadm sec
```

Set the DCOM security activation password:

```
%mwadm sec -passwd
```

Display the security authentication type in effect:

```
%mwadm sec -type
```

Set the security authentication type to win:

```
%mwadm sec -type win
```

Display the security authentication port number in effect:

```
% mwadm sec -port
```

Set the security authentication port number:

```
% mwadm sec -port 685
```

Display the domain name used by the MSC security mechanism:

```
% mwadm sec -domain_name
```

Set the domain name used by the MSC security mechanism to ACME:

```
% mwadm sec -domain_name ACME
```


Display the name of the domain server used by the MSC security mechanism:

```
% mwadm sec -domain_server
```

Set the domain server used by the MSC security mechanism to a machine called purple:

```
% mwadm sec -domain_server purple
```

Set the domain server used by the MSC security mechanism to a machine whose IP address is 172.17.1.97:

```
% mwadm sec -domain_server 172.17.1.97
```

start Command—Start All MSC Services

The MSC services should start automatically on system boot. If the services are stopped at any time, you can use this command to restart all MSC services.

Note: Only the MSC administrator or the superuser (root) can execute this command.

This command has the following format:

```
mwadm start
```

Example—start command:

Start the MSC services:

```
% mwadm start
```

status Command—Verify MSC Status

This command verifies whether the MSC is functioning properly.

This command has the following format:

```
mwadm status
```

Example—status command:

Display the MSC status:

```
% mwadm status
```

stop Command—Shut Down All MSC Services

This command shuts down all MSC services in an orderly manner. The MSC services shut down only if there is no MainWin process running. If a MainWin process is running, an error message displays and no action is taken. You can force a shutdown of the MSC services by using the `-f` option.

Note: Only the MSC administrator or the superuser (root) can execute this command.

This command has the following format:

```
mwadm stop [-f]
```

-f

Forcefully shuts down all MSC services. Processes attached to the MSC services are terminated without warning.

Examples—stop command:

Stop MSC services:

```
% mwadm stop
```

Forcefully stop the MSC (even if the processes are attached to the services):

```
% mwadm stop -f
```

version Command—Display the MSC Version

This command displays the version number of MSC.

This command has the following format:

```
mwadm version
```

Example—version command:

Display the MSC version number:

```
% mwadm version
```

port Command—Control the RPCSS Service Port

This command lets you control the port that the MSC RPCSS service uses. To change the port setting with this command, you must first shut down the MSC services (see stop command). If you want to view the settings only, then there is no need to stop the MSC services. When executed with no parameters, the port command prints the current setting.

Note: Distributed COM is not supported for the auto mode. To enable DCOM with other UNIX hosts, you must change the mode to fixed and set the same port number on all UNIX hosts participating in the DCOM interaction. To enable DCOM interaction with Windows, set the fixed port number to 135 on all relevant UNIX hosts. For the new setting to take effect, you must restart the MSC services (see start command).

This command has the following format:

```
mwadm port [-amode {auto | {fixed port_no}} ]
```

-amode { auto | {fixed *port_no*}}

Defines how MSC RPCSS service allocates the number for the port it attempts to attach to when it starts. You must specify one of the following parameters:

auto

Instructs the RPCSS service to use the first available port

fixed *port_no*

Instructs the RPCSS service to attach itself to a specific, pre-defined port, *port_no*

Default: auto

Examples—port command:

Instruct the RPCSS server to use port 135 always:

```
% mwadm port -amode fixed 135
```

Display the port allocation setting currently in effect:

```
% mwadm port
```

The previous command gives the following result:

```
% fixed (135)
```

How You Can Configure COM and DCOM

This section describes how to configure COM and DCOM to enable access to the OpenROAD Server.

How You Can Use the DCOM Configuration Utility on UNIX

OpenROAD includes the `dcomcnfg` utility, which only the MSC administrator can use. On Windows, you can use the `dcomcnfg` utility to modify various security settings such as the default DCOM communication properties and the default security settings for specific COM servers.

After changing the authentication or the impersonation levels, you must restart MSC for these changes to be effective. You can manually restart MSC by issuing the following commands:

```
%mwadm stop  
%mwadm start
```

How You Can Set the Identity of an Out-of-process COM Server

On Windows, you can set the identity of the user who launches the COM server using the `dcomcnfg` utility, or by directly setting the appropriate registry keys relevant to this server. However, further operations are required in certain cases:

Running as an interactive user

There is no support for an interactive user on UNIX because there is no user equivalent to the Windows term for an interactive user.

Running as a launching user

This is the default mode. On UNIX, the user who starts the COM server from the command line or from a daemon process script is the owner of the process.

Note: When starting the COM server as a daemon process, ensure that the user name specified in the daemon process script is actually the user that owns the daemon process.

Running as a specific user

To run as a specific user, you must complete the following steps:

1. Set the COM server executable file ownership and permissions. For more information about setting the permissions, see Set the COM Server Executable File Ownership and Permissions to Run as a Specific User in this chapter.
2. Provide a server initialization script to ensure the proper environment is set before launching the COM server. For information on setting up this script, see Server Initialization Script in this chapter.

Set the COM Server Executable File Ownership and Permissions to Run as a Specific User

You can set the COM server executable file ownership and permissions to ensure that it always runs as a specific user.

To set file ownership and permissions

1. Set the owner of the COM server executable file to the specific user that is to own the process, no matter who launches the process.
2. Set the COM server's executable file permission to Set user ID on execution (also known as sticky bit).

Combined with the previous step, this step ensures that the running process is always be owned by the intended user.

To set the file permissions manually, enter the following command at the command prompt:

```
%chmod u+s filename
```

3. Use dcomcnfg to set the COM server identity to this user and provide the user name.

Note: The UNIX version of dcomcnfg does not require the user password; the previous steps ensure the user's identity.

If you intend to expose this service to a remote client (a client process that does not run on the same host as the server), you must set up the Visual Main Win DCOM security (see How You Can Set Up Distributed COM in this chapter).

Server Initialization Script

When Visual MainWin launches an out-of-process COM server with the identity of the launching user, it ensures that the COM server inherits its environment (MWHOME, LD_LIBRARY_PATH, and so on) from the calling client process. In some situations, however, Visual MainWin cannot automatically pass the client environment to the server. For example:

- The client is on a remote host.
- The server is set up to run as a specific user.

In both of these cases, a script must be provided to Visual MainWin to set the proper environment for the COM server and launch it. In case the environment cannot be copied from the client process, the Visual MainWin DCOM service control manager (RPCSS) uses the script to launch the COM server.

The following points explain how to write a server initialization script, its format, file permissions for the script, and where the script is located at runtime.

Writing a server initialization script

There are no special requirements for the language in which the script is written. The only requirement is that it be executable and it achieves the remaining objectives, following.

Contents of the script

The script should contain statements that:

- Set up the Visual MainWin runtime environment
- Set up any additional environment variables
- Launch the server executable with proper parameters

File permissions for the script

The script should enable read and execute permissions for all users authorized to start the COM server.

The following is an example of an initialization script:

```
#!/bin/tcsh -f
setenv II_SYSTEM /usr/install
setenv MWHOME /usr/install/ingres/mainwin/mw
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:%II_SYSTEM/ingres/lib
source $MWHOME/setmwruntime.csh
set path=
(
$path \
$II_SYSTEM/ingres/bin \
$II_SYSTEM/ingres/utility \
)
setenv DISPLAY {hostname | IP_address}:0
exec $*
```

Location of the initialization script for Visual MainWin at runtime

Visual MainWin searches for an initialization script in the directory where the server executable resides. It performs the search according to the following search rules:

3. Look for a file named *server.init*, where *server* is the server name. For example, if the server is registered by the name *oraso*, the init script file name is expected to be *oraso.init*.

If no script is found according to this rule, go to the next rule.

4. Search for a script named *init*.

If Visual MainWin finds no script based on the above rules, RPCSS returns a "Server execution failed" error code to the calling process.

How You Can Set Up Distributed COM

This release of OpenROAD supports Distributed COM communication between Windows and UNIX hosts, letting you deploy DCOM services on UNIX servers or use DCOM services from a UNIX client. Security is set up to authenticate the users requesting a COM service and to limit their access to system resources such as launching specific COM servers, accessing specific registry keys, and accessing files.

The following steps describe how you can set up Distributed COM communication to achieve this heterogeneous communication.

1. Decide the port to be used for RPCSS.

For DCOM communication to take place, ensure that the MSC RPCSS services use the same port on all the hosts participating in the DCOM interaction.

- If the interaction will take place between UNIX hosts only, decide on an arbitrary port number (as long as it is available on all the hosts).
- If the interaction will include Windows hosts, the only option available is port 135. Before assigning the port, ensure that the port is not already assigned. You can check in */etc/services* or use *netstat -a*.

Note: Port numbers below 1024 are privileged numbers, and you must have root privileges to assign them.

On some HP-UX and IBM AIX systems, port 135 is already used by the DCE daemon. If you need DCOM between Windows and HP-UX or IBM AIX, ask your system administrator to reconfigure the system or remove the DCE daemon.

Note: To change the port setting with *mwadm port* command, the MSC services must be stopped if they are running. If you only want to view the settings, there is no need to stop the MSC services.

2. Decide the authentication type to use.

To ensure high security standards, the DCOM security mechanism uses the local authentication services to verify the identity of remote users requesting DCOM services. The DCOM security mechanism supports two types of authentication services:

UNIX-based authentication

Relies on the local UNIX security system. It can be used in a homogeneous environment where all hosts are UNIX. This type of authentication is easy to use and provides better performance than Windows-based authentication.

Windows-based authentication

Relies on authentication from the Windows domain controller. It is a requirement for heterogeneous UNIX-Windows environments. This type of authentication requires additional settings on the primary Windows domain controller for the domain where authentication is required.

Note: UNIX-based authentication is the default and requires no additional setup.

Set Up DCOM for a Group of UNIX Hosts

You can enable Distributed COM interaction between UNIX hosts by setting up the Visual MainWin DCOM security.

To set up every UNIX host that uses DCOM

1. Use MSC to configure DCOM on every UNIX host that uses DCOM.
2. Set up the MSC RPCSS service to use a specific fixed port.

For more information about using the fixed port, see How You Can Set Up Distributed COM in this chapter.

3. Have an MSC administrator set up the relevant DCOM security parameters:

- Set up the DCOM security activation password.

The DCOM security activation password is used by Visual MainWin DCOM to deny DCOM services from untrusted hosts. Only hosts configured with an identical DCOM security activation password can participate in DCOM interaction with each other.

To set up the DCOM security activation password, log in as the MSC administrator and execute the following command:

```
%mwadm sec -passwd
```

A prompt appears for you to enter the new password and verify it.

Note: You must enter the same password-including capitalization-on all hosts. The Visual MainWin DCOM authentication type is by default UNIX.

- Enter the following command to check the authentication type in use:

```
%mwadm sec -type
```

- Enter the following command to set up the DCOM security authentication type:

```
%mwadm sec -type unix
```

How You Can Set Up DCOM for a Mixed Group of UNIX and Windows Hosts

You can enable DCOM interaction between a mixed group of Windows and UNIX hosts. This section first provides a high-level list of steps required on Windows and UNIX, and then provides detailed instructions for each of these high-level steps.

For Windows workstations

For every Windows workstation to participate in the DCOM interaction, the following preparations are required:

- The minimum O/S supported is Windows NT 4.0 Service Pack 3.
- Connection-oriented TCP/IP must be the first protocol in the list of Windows DCOM protocols.

Note: On Windows NT 4.0, you can launch a COM server on a remote machine only with the identity of the launching user. On Windows 2000 and Windows XP, you can launch a COM server on a remote machine only with the identity of the specific user.

For UNIX hosts

For every UNIX host to participate in the DCOM interaction, the MSC RPCSS service must be set up to use port number 135. For more information about setting up the service, see How You Can Set Up Distributed COM in this chapter.

Place TCP/IP First on Windows

Because TCP/IP is the only protocol that UNIX DCOM currently supports, Connection-oriented TCP/IP must be the first protocol in the list of Windows DCOM protocols.

You can place Connection-oriented TCP/IP first in the list of Windows DCOM protocols by using the dcomcnfg utility. If you have Windows NT 4.0 Service Pack 4 or later, perform the following procedure.

To place TCP/IP first in the list

1. Ensure that you have administrator privileges, and then run the dcomcnfg utility.
2. Select the Default Protocols tab.
3. If Connection-oriented TCP/IP is not the first entry in the list, select it and use the Move Up button to move it to the top of the list.
4. Click Apply, and then click OK.

You can also place Connection-oriented TCP/IP first in the list of Windows NT DCOM protocols using regedit if you have Windows NT 4.0 Service Pack 3 or later.

To place TCP/IP first in the list using regedit

1. Open the Windows regedit utility.
2. Manually edit the Registry key, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\DCOM Protocols, so that ncacn_ip_tcp protocol appears first in the list of DCOM protocols.

How You Can Place TCP/IP First on UNIX

Set up DCOM on all the hosts. All UNIX hosts participating in DCOM interaction must be set up with MSC. By default, the Visual MainWin DCOM security authentication type is unix.

Chapter 5: Installing and Configuring OpenROAD on HP Tru64

This section contains the following topics:

[OpenROAD Installation on HP Tru64](#) (see page 59)

OpenROAD provides runtime support on HP Tru64 for COM, DCOM, and Registry Services, which allow cross-platform communication between hosts running HP Tru64, IBM AIX, Sun Solaris, HP-UX, Linux, and various Windows-based operating systems. The OpenROAD Server makes extensive use of these technologies to develop multi-tiered distributed applications.

OpenROAD Installation on HP Tru64

OpenROAD installs either as a new client installation or into an existing Ingres installation. For instructions, see the appropriate section, following. If you plan to install this release into an existing Ingres installation, back up your existing Ingres installation before starting the install.

Before starting the installation process, review the information in this chapter. It contains pertinent information regarding registry upgrades and DCOM support.

Note: You must have superuser (root) privileges to complete the installation.

Install OpenROAD as a New Installation

You can install OpenROAD as a new installation using the following procedure.

To start the installation

1. Log in as root.
2. Mount the CD-ROM drive, and insert the OpenROAD installation media into the drive.

3. Change the directory to the CD-ROM drive and then enter the following command:

```
#sh ./install.sh
```
4. Select PackageInstall from the menu, and then select one of the following packages:
 - OpenROAD Runtime
 - OpenROAD Runtime and Development
 - Respond appropriately to the prompts to complete the installation.

Install OpenROAD into an Existing Ingres Installation

You must use ORINSTALL.ING to install OpenROAD into an existing Ingres installation.

The registry file created prior to OpenROAD 4.1 SP3 is incompatible with this release. The installer can upgrade your existing registry file to the new format, or you can perform the upgrade after completing the installation. For more information on upgrading after installing, see Update Your Registry on HP Tru64 in this chapter. If you are upgrading from OpenROAD 4.1 SP3, you do not need to upgrade your existing registry file.

Prepare for Installation

Before you install OpenROAD into an existing installation, you must perform the following procedure.

To prepare for installation

1. Shut down your existing Ingres installation.
2. Shut down the OpenROAD Server.
3. Back up your installation.
4. Set the II_SYSTEM environment variable to the existing installation you want to upgrade.
5. Decide the OpenROAD package you want to install:
 - ordev for Runtime and Development
 - orrun for Runtime only

Note: The ordev and orrun products do not include the Ingres Net 2.6 components. The installer does not allow an upgrade because you are installing into an existing Ingres installation. To upgrade your Ingres Net in this situation, you must acquire and use an Ingres installer.

6. If you plan to upgrade an existing registry file at the time of installation, ensure that you have set MWREGISTRY to your registry file using the full path name. This applies even if you are using the default name for the registry file.

If you are using the default registry name, set MWREGISTRY as follows:

```
%setenv MWREGISTRY $HOME/windows/registry.bin
```

If you use a user-defined registry file name, then set MWREGISTRY to your registry file name with full path as follows:

```
%setenv MWREGISTRY /full_path/existing_registry_name
```

Note: The installer skips the registry conversion step if the MWREGISTRY environment variable is not set. In addition, an X Server must be available when an OpenROAD application performs the conversion.

Start the Installation

You install OpenROAD from the installation media using ORINSTALL.ING.

To start the installation

1. Mount the CD-ROM drive and then insert the OpenROAD installation media into the drive.
2. Enter the following commands:

```
%cd $II_SYSTEM/ingres
%tar -xvf /CD-ROM/axp_osf/orr5.tar ORINSTALL.ING
%./ORINSTALL.ING /CD-ROM/axp_osf/orr5.tar {ordev | orrun}
```

CD-ROM

Specifies the mount point for your CD-ROM drive

During installation, the installer prompts you to confirm the upgrade of the existing registry to the new format.

Update Your Registry on HP Tru64

If you skipped the registry upgrade during installation, you can convert an existing registry file to the new format or create a new registry file. For instructions, see the appropriate section, following.

Note: We recommend that you create a new registry file for this release. To do so, first remove the existing registry files.

Create a New Registry File for Each User

To create a registry file for a specific user, set the environment variable MWREGISTRY to point to a path that is unique for that user, and then create the registry.

To create a new registry file for each user

1. Enter the following commands:

```
%setenv MWREGISTRY unique_path_for_specific_user/registry.bin  
%mwregedit -new
```

unique_path_for_specific_user

Specifies the full path to the specific user's registry file

2. Re-register the OpenROAD Server using the following commands:

```
%setenv USERNAME host_login  
%setenv COMPUTERNAME host_name  
%w4glrun asreg.img
```

host_login

Specifies a valid host login, such as "ingres"

host_name

Specifies the name of your UNIX machine

Upgrade an Existing Registry File from OpenROAD 4.1

If you are upgrading from OpenROAD 4.1/0003 or 4.1/0109, you must upgrade the registry file using the utility ru41sp2.img to complete the installation. The purpose of this utility is to update existing OpenROAD Server keys.

Note: Before you upgrade, verify that your OpenROAD runtime environment is set up correctly.

To upgrade an existing registry file

Run the upgrade utility using the following command:

```
%w4glrun ru41sp2.img
```

OpenROAD Utilities orspostart and orspostop

The utilities orspostart and orspostop require no GUI and can run without an X Server graphic display. You can use the X Virtual Frame Buffer (Xvfb) to run these utilities in the absence of an X Server graphics display device.

Start the OpenROAD Server from a Daemon Process on UNIX

On a Windows platform, you can start the OpenROAD Server (orspo) using orsposvc as system service after Windows NT boots up and is running.

On UNIX platforms, two utilities, orspostart and orspostop, are used to start and stop the orspo process.

orspostart

Orspostart is a daemon process that can be run during operating system startup or started from the command line, with or without the existence of an X Server graphics display device (for instructions, see the appropriate section, following). When orspostart runs, it launches orspo and keeps orspo running. When orspostart receives a kernel event sent by orspostop, it shuts down the orspo gracefully.

orspostop

Orspostop is a utility that simply sends a shutdown event to the daemon process orspostart.

Launch orspo with an X Server Graphics Display Device

To enable orspostart to run as a UNIX daemon at operating system startup, you must create a startup script to initialize the OpenROAD runtime environment and execute the binary program, orspostart. For example, on a Solaris system, you may need to log in as root and create a script S99orspo in /etc/rc2.d. This script accepts start and stop command line options. The script initializes your OpenROAD Server runtime environment and sets up the correct "run as user id" (for example, run as ingres or any other user). The script then executes orspostart in the background for the start option, and executes orspostop for the stop option.

To start orspostart from the command line

Set the runtime environment, and then execute the following command:

```
%orspostart &
```

To run orspostop from the command line

Set the runtime environment, and then execute the following command:

```
%orspostop
```

Launch orspo Without an X Server Graphics Display Device

Orspo can be launched without an X Server graphics display device. However, it still requires some kind of virtual server to provide a DISPLAY to OpenROAD applications. You can use Xvfb in the absence of an X Server graphics display device.

You can run orspostart and orspostop with Xvfb server in the same way as with (video) X Server, with the following exceptions:

- The Xvfb server must be started with its unique DISPLAY number before orspostart is executed.
- The OpenROAD runtime environment should set the DISPLAY to point to the display number associated with Xvfb server.
- The II_SYSTEM environment variable should be set to the existing installation you want to upgrade.

Configure COM and DCOM

You must configure COM and DCOM to allow access to the OpenROAD Server.

Because Windows NT uses port 135 exclusively for DCOM communications, UNIX RPCSS must be directed explicitly to use port 135 for DCOM communication with Windows NT. However, port number 135 is a privileged port on UNIX and only a root user can bind to it and accept connections. You must be a root user to set up RPCSS and bind to port 135.

To configure COM and DCOM for all UNIX platforms

Issue the following commands:

```
%cd $MWHOME/bin-osf1v4_optimized
%chown root rpcss
%chmod agu+s rpcss
```

Note: We recommend that you dedicate only one port to enable DCOM communications. That is, on each UNIX machine, only one MWRPC_ENDPOINT should be used to set up the OpenROAD Server runtime environment. Setting up MWPRC_ENDPOINT to use port 135 enables DCOM communications from both Windows NT and UNIX environments.

If you use the orspostart utility, there is no need to start RPCSS as a background process before launching orspo or oraso.

Index

A

- active repository • 22
- Application mode • 16
- Application Server • 8, 24
 - generalized signature • 24
- Application Workbench • 17
- applications
 - deployment in n-tier environment • 8
 - development in n-tier environment • 8
 - dynamic • 22
 - scalable • 22
- automatic generation
 - field • 21
 - frame • 21

C

- class browser • 21
- component libraries • 20
- Component mode • 16
- component sharing • 20

D

- debugging
 - interactive • 16
- development environment • 17
 - interactive (IDE) • 15, 16
- documentation • 9
 - Enterprise Access Installation and Operations Guide • 10
 - Ingres Embedded SQL for C Companion Guide • 10
 - Ingres OpenSQL Reference Guide • 10
 - Ingres SQL Reference Guide • 10
 - OpenROAD Application Server User Guide • 9
 - OpenROAD Getting Started • 9
 - OpenROAD Language Reference Guide • 9
 - OpenROAD Readme • 9
 - OpenROAD Release Summary • 9
 - OpenROAD Transformatin Runtime Readme • 10
 - OpenROAD User Guide • 9
 - Programming Guide • 9

E

- eClient • 8
 - deployment • 25
- editors
 - visual • 17

F

- features
 - active repository • 22
 - application management utilities • 19
 - application server • 24
 - Application Workbench • 17
 - automatic field generation • 21
 - automatic frame generation • 21
 - class browser • 21
 - component libraries • 20
 - component sharing • 20
 - dynamic applications • 22
 - eClient • 25
 - field templates • 20
 - frame templates • 20
 - interactive debugging • 16
 - interactive testing • 16
 - object-oriented language • 21
 - open architecture • 20
 - report writing • 19
 - scalable applications • 22
 - visual editors • 17
- field templates • 20
- frame templates • 20

I

- Ingres
 - stopping Net • 28

M

- management facilities • 22
- modes • 16
- multiple platforms • 7

N

networking

- Ingres/Enterprise Access • 14

- intelligent gateways • 14

- n-tier environment • 8

O

- object-oriented language • 21

- open architecture • 20

- open database access • 13

OpenROAD

- Application Server • 8, 24

- documentation • 9

- eClient • 8, 25

- features • 13

- multiple platforms • 7

- overview • 7

- Reporter • 19

- structured data • 24

- upgrading from earlier release • 27

R

- Reporter • 19

repository

- active • 22

- management facilities • 22

T

templates

- field • 20

- frame • 20

testing

- interactive • 16

tools

- visual • 16

U

utilities

- application management • 19

V

- visual editors • 17