Graphics™ 2.5 Developer’s Guide

Release 2.5

Volume I
Preface

This guide is one of two documents in the Graphics 2.5 documentation set. Before you use it, you should be familiar with the following topics:

- audience
- structure of this guide
- typographic conventions
- how to use this guide
- Graphics documentation and related information
- how to contact Oracle with your comments about the product or documentation

This section will prepare you for using this guide and direct you to the minimum information you need to get started.
This guide is one of two documents in the Graphics 2.5 documentation set. Before you use it, you should be familiar with the following topics:

• audience
• structure of this guide
• typographic conventions
• how to use this guide
• Graphics documentation and related information
• how to contact Oracle with your comments about the product or documentation

This section will prepare you for using this guide and direct you to the minimum information you need to get started.
Audience

The information presented here is intended primarily for application developers and readers who want to create displays. Readers should have a working knowledge of SQL, PL/SQL, and Oracle Server concepts.

Structure of this Guide

This guide provides comprehensive information about Graphics presented in concise modules.

Major Sections of this Guide

This document consists of the following major sections:

- **Part I**
  - Learning Graphics
    - This section contains introductory and conceptual information, including a comprehensive tutorial, to help you get started with Graphics.

- **Part II**
  - Using Graphics
    - This section contains step by step instructions for how to perform most major tasks in Graphics.

Typographic Conventions

Throughout this guide, we use typographic conventions to distinguish important elements from the body text of the document.

Menu Navigation Syntax

Menu navigation is represented by the name of the menu followed by an arrow and the menu item. For example:

**File**—>**New**

Means you should choose **New** from the **File** menu.

If the menu item leads to a submenu, another arrow and the submenu item is added. For example:

**File**—>**Open**—>**Database**

Means you should choose **Open** from the File menu, and then **Database** from the Open submenu.
Audience
The information presented here is intended primarily for application developers and readers who want to create displays. Readers should have a working knowledge of SQL, PL/SQL, and Oracle Server concepts.

Structure of this Guide
This guide provides comprehensive information about Graphics presented in concise modules.

This document consists of the following major sections:

- Learning Graphics
  This section contains introductory and conceptual information, including a comprehensive tutorial, to help you get started with Graphics.

- Using Graphics
  This section contains step by step instructions for how to perform most major tasks in Graphics.

Typographic Conventions
Throughout this guide, we use typographic conventions to distinguish important elements from the body text of the document.

Menu navigation is represented by the name of the menu followed by an arrow and the menu item. For example:

- File —> New
  Means you should choose New from the File menu.

If the menu item leads to a submenu, another arrow and the submenu item is added. For example:

- File —> Open —> Database
  Means you should choose Open from the File menu, and then Database from the Open submenu.
Throughout this guide, and in the online documentation, we use the following notational conventions:

- **Font Change** Denotes that you should enter text exactly as shown.
- **[]** Denotes that the enclosed item is optional. Do not enter the brackets.
- **{}** Denotes that one, and only one, of the enclosed items must be entered. If the braces are surrounded by brackets, then all of the enclosed items are optional. Do not enter the braces.
- **|** Separates options within brackets and braces. You must enter one, and only one, of the options separated by the vertical bar. Do not enter the vertical bars.
- **_** Denotes that the underlined text is the default.

Denotes values or options.

### How to Use this Guide

Don’t worry. You don’t have to read the whole guide to become a productive Graphics user. Below are suggested chapters to read to help you get started, based on your level of experience with Graphics.

#### New or Non–Technical Users

You need to gain a working knowledge of Graphics so you can start building displays. We suggest the following steps:

1. Read through Chapter 1, “Getting Started,” to get an overview of Graphics features and prepare for using Graphics.
2. Work through the Tutorial in Chapter 2 to build a display from start to finish, taking time to understand why you take certain steps.
3. Read Chapter 3, “Understanding Graphics,” for a description of Graphics components. Understanding how Graphics is structured will help you to get the most from this product.
4. Chapters 4 and beyond can help you learn how to perform a particular task.

#### Experienced or Technical Users

You may already have a working knowledge of Graphics. However, you need to learn about the new features in Release 2.5.
Throughout this guide, and in the online documentation, we use the following notational conventions:

- **Denotes that you should enter text exactly as shown.**

- **Denotes that the enclosed item is optional. Do not enter the brackets.**

- **Denotes that one, and only one, of the enclosed items must be entered. If the braces are surrounded by brackets, then all of the enclosed items are optional. Do not enter the braces.**

- **Separates options within brackets and braces. You must enter one, and only one, of the options separated by the vertical bar. Do not enter the vertical bars.**

- **Denotes that the underlined text is the default.**

- **Denotes values or options.**

---

**How to Use this Guide**

Don't worry. You don't have to read the whole guide to become a productive Graphics user. Below are suggested chapters to read to help you get started, based on your level of experience with Graphics.

1. Read through Chapter 1, "Getting Started," to get an overview of Graphics features and prepare for using Graphics.
2. Work through the Tutorial in Chapter 2 to build a display from start to finish, taking time to understand why you take certain steps.
3. Read Chapter 3, "Understanding Graphics," for a description of Graphics components. Understanding how Graphics is structured will help you to get the most from this product.
4. Chapters 4 and beyond can help you learn how to perform a particular task.

You may already have a working knowledge of Graphics. However, you need to learn about the new features in Release 2.5.
1. Read “New Features with the 2.5 Release,” in Chapter 1, “Getting Started,” to familiarize yourself with Release 2.5 features.

2. At this point, you may be ready to begin building displays. However, we recommend you work through the Tutorial in Chapter 2 to build a display from start to finish using Release 2.5.

3. Although some of the information may be familiar to you, we recommend that you read Chapter 3, “Understanding Graphics,” to ensure that you understand important new concepts in Release 2.5.

4. Refer to Chapters 4 and beyond whenever you need to review the steps involved in performing a particular task.

Graphics Documentation and Related Information

Graphics 2.5 enables you to perform a wide variety of data–related tasks while creating your displays, such as querying local or remote databases and importing files. Graphics can interact with several other products (such as the Oracle Server) and various networking tools. The following figure shows some of the major products you will probably use with Graphics and lists references. More information follows the figure.
1. Read "New Features with the 2.5 Release," in Chapter 1, "Getting Started," to familiarize yourself with Release 2.5 features.

2. At this point, you may be ready to begin building displays. However, we recommend you work through the Tutorial in Chapter 2 to build a display from start to finish using Release 2.5.

3. Although some of the information may be familiar to you, we recommend that you read Chapter 3, "Understanding Graphics," to ensure that you understand important new concepts in Release 2.5.

4. Refer to Chapters 4 and beyond whenever you need to review the steps involved in performing a particular task.

Graphics Documentation and Related Information

Graphics 2.5 enables you to perform a wide variety of data–related tasks while creating your displays, such as querying local or remote databases and importing files. Graphics can interact with several other products (such as the Oracle Server) and various networking tools. The following figure shows some of the major products you will probably use with Graphics and lists references. More information follows the figure.
The following is a list of specific titles and part numbers of Graphics and related documentation:


- *Graphics Reference Manual*, Part No. A32483–1, provides detailed reference information about all aspects of Graphics. It is intended primarily to answer specific questions that you might have about using Graphics (e.g., what can you enter in a certain field).


Each Graphics manual listed above is also shipped as an Oracle Book document. For the exact location and names of the document files, see your *Developer/2000 Installation Guide*.

The online help system is intended to provide answers to questions you may have about Graphics as you work.

The following table briefly presents each piece of the set, its intended audience, and any suggested prerequisite reading.

<table>
<thead>
<tr>
<th>Document</th>
<th>Primary Audience</th>
<th>Prerequisite Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics Developer’s Guide</td>
<td>novice and advanced</td>
<td>none</td>
</tr>
<tr>
<td>Online Help</td>
<td>novice and advanced</td>
<td>Graphics Developer’s Guide</td>
</tr>
</tbody>
</table>

As an application designer using Graphics Release 2.5, you may need to refer to some or all of the documents listed below. Please see the document in question for information on its specific purpose and usage.

- *Developer/2000 Installation Guide* or *System Release Bulletin*. This document is different for each hardware/software platform. Ask your sales representative for the appropriate part number.

- *Oracle7 Server Concepts Manual*

- *Oracle7 Server Administrator’s Guide*
The following is a list of specific titles and part numbers of Graphics and related documentation:

- **Graphics Documentation Set**, Part No. A32481–2, is a documentation bundle that contains the following:

- **Graphics Reference Manual**, Part No. A32483–1, provides detailed reference information about all aspects of Graphics. It is intended primarily to answer specific questions that you might have about using Graphics (e.g., what can you enter in a certain field).


Each Graphics manual listed above is also shipped as an Oracle Book document. For the exact location and names of the document files, see your **Developer/2000 Installation Guide**.

The online help system is intended to provide answers to questions you may have about Graphics as you work.

The following table briefly presents each piece of the set, its intended audience, and any suggested prerequisite reading:

<table>
<thead>
<tr>
<th>Document</th>
<th>Primary Audience</th>
<th>Prerequisite Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics Developer's Guide</td>
<td>novice and advanced designers</td>
<td>none</td>
</tr>
<tr>
<td>Online Help</td>
<td>novice and advanced designers</td>
<td>Graphics Developer's Guide</td>
</tr>
</tbody>
</table>

As an application designer using Graphics Release 2.5, you may need to refer to some or all of the documents listed below. Please see the document in question for information on its specific purpose and usage.

- **Developer/2000 Installation Guide** or **System Release Bulletin**. This document is different for each hardware/software platform. Ask your sales representative for the appropriate part number.
- **Oracle7 Server Concepts Manual**
- **Oracle7 Server Administrator's Guide**
• Oracle7 Server Application Developer’s Guide
• Oracle7 Server Messages and Codes Manual
• Oracle7 Server SQL Language Reference Manual
• PL/SQL User’s Guide and Reference
• Programmer’s Guide to the ORACLE Precompilers
• the precompiler supplement for your programming language
• Oracle Network Manager Administrator’s Guide
Your Comments Are Welcome

We value and appreciate your comments as an Oracle user. As we write, revise, and evaluate our work, your opinions are the most important input we receive. At the back of this guide is a Reader’s Comment Form. We encourage you to use this form to tell us both what you like and dislike about this (or other) Oracle documentation. If the form is missing, or you would like to contact us, please use the following addresses and phone numbers.

For documentation questions/comments, contact:

Graphics Documentation Manager
Oracle Corporation
Box 659409
500 Oracle Parkway
Redwood Shores, California 94065–5028
USA
Voice: 415–506–7000
Fax: 415–506–7200

For product questions/comments, contact:

Graphics Product Manager
Oracle Corporation
Box 659409
500 Oracle Parkway
Redwood Shores, California 94065–5028
USA
Voice: 415–506–7000
Fax: 415–506–7200
Your Comments Are Welcome

We value and appreciate your comments as an Oracle user. As we write, revise, and evaluate our work, your opinions are the most important input we receive. At the back of this guide is a Reader's Comment Form. We encourage you to use this form to tell us both what you like and dislike about this (or other) Oracle documentation. If the form is missing, or you would like to contact us, please use the following addresses and phone numbers.

For documentation questions/comments, contact:
Graphics Documentation Manager
Oracle Corporation
Box 659409
500 Oracle Parkway
Redwood Shores, California 94065–5028
USA
Voice: 415–506–7000
Fax: 415–506–7200

For product questions/comments, contact:
Graphics Product Manager
Oracle Corporation
Box 659409
500 Oracle Parkway
Redwood Shores, California 94065–5028
USA
Voice: 415–506–7000
Fax: 415–506–7200
# Contents

## PART I LEARNING GRAPHICS

### Chapter 1 Getting Started ............................................ 1 – 1
- An Overview of Graphics ............................................ 1 – 1
- New Features with this Release ................................. 1 – 4
- Executables ............................................................. 1 – 6
- Installing Graphics on Your System ......................... 1 – 7
- Using the Help System .............................................. 1 – 8
- Resolving Errors ..................................................... 1 – 9

### Chapter 2 Tutorial ....................................................... 2 – 1
- Before You Begin .................................................... 2 – 2
- Tutorial Scenario ..................................................... 2 – 2
- Lesson 1: Create a Pie Chart .................................... 2 – 3
- Lesson 2: Create a Column Chart ............................ 2 – 8
- Lesson 3: Modify the Charts .................................... 2 – 11
- Lesson 4: Create a Drill-down Relationship ............. 2 – 13
- Lesson 5: Create a Format Trigger ......................... 2 – 18
- Lesson 6: Add a Text Object ................................... 2 – 22
- Lesson 7: Create a Break Chart ......................... 2 – 24
- Lesson 8: Create a Query Filter ......................... 2 – 26
- Lesson 9: Create a Second Layer ......................... 2 – 31
- Lesson 10: Create a Button Procedure ................. 2 – 34
- Lesson 11: View The Results ................................ 2 – 39
Sharing Information Among Displays .......................... 15 – 4
Managing Windows ............................................. 15 – 6

Chapter 16

Integrating Displays ............................................. 16 – 1
Preparing a Display for Embedding ......................... 16 – 2
Including a Display in Forms ................................. 16 – 4
Including a Display in Reports ............................... 16 – 15
Including a Display in OLE Container Applications ....... 16 – 15
Calling Other Products From Graphics ...................... 16 – 16

Index
Learning Graphics
Getting Started

This chapter contains initial information about Graphics that you’ll need to know before you begin the Tutorial in Chapter 2.

The following topics are covered in this chapter:

- an overview of Graphics – 1 – 1
- new features with the 2.5 release – 1 – 4
- executables – 1 – 6
- installing Graphics on your system – 1 – 7
- using the help system – 1 – 8
- resolving errors – 1 – 9

An Overview of Graphics

Corporations today are flooded with information. The challenge is to interpret that information and relay it to decision makers in a form they can readily understand. Graphics opens up the next era in corporate decision-making by integrating an end-user graphics system into the existing information network. It’s a tool that allows everyone in an organization to visually analyze information online.
Getting Started

This chapter contains initial information about Graphics that you’ll need to know before you begin the Tutorial in Chapter 2.

The following topics are covered in this chapter:

• an overview of Graphics
• new features with the 2.5 release
• executables
• installing Graphics on your system
• using the help system
• resolving errors

An Overview of Graphics

Corporations today are flooded with information. The challenge is to interpret that information and relay it to decision makers in a form they can readily understand. Graphics opens up the next era in corporate decision-making by integrating an end-user graphics system into the existing information network. It’s a tool that allows everyone in an organization to visually analyze information online.
Uniting both information and presentation resources, Graphics enables you to create multimedia graphical displays that can be dynamically linked to data sources. You can create Graphics displays as stand-alone modules, or you can integrate them with modules in Forms and Reports to create larger business applications.

Functionality of Graphics

Graphics provides many options for graphically portraying data. Here’s what you can do:

- Retrieve data from a wide variety of sources and display it graphically, using charts, line drawings, bit-mapped images, text, and sound.
- Draw artwork such as lines, polygons, and freehand shapes using a full set of editing tools.
- Use PL/SQL to control flow, add interactive features, and animate artwork. Write, edit, and debug your programs within Graphics.
- Make charts and drawings dynamic, by linking them to data sources. Write procedures that update the display to reflect changes in the data.
- Include interactive features for end-users, such as mouse-activated buttons, or use timers to trigger certain operations automatically.
- Choose from 56 pre-defined chart templates that you can customize and reuse for multiple displays.
- Open multiple windows for working on more than one display at a time.
- Store displays in either your file system or the Oracle Server.
- Import and export CGM (ISO, ANSI, or GKS) drawings; BMP, JPEG, and TIFF images; and AIFF–c sound files. Also import PCX and PICT images.
- Provide end-users with a simplified runtime version of the interface for executing Graphics displays.

Features of Graphics

Graphics provides the following major features to support the functionality described above:

Point-and-Click Interface

With its bit-mapped point-and-click interface, Graphics enables you to create database-driven graphics without programming. You can create displays using a combination of pull-down menus, dialog boxes, and object-oriented tools, accessing data from a variety of sources.
Uniting both information and presentation resources, Graphics enables you to create multimedia graphical displays that can be dynamically linked to data sources. You can create Graphics displays as stand-alone modules, or you can integrate them with modules in Forms and Reports to create larger business applications.

Graphics provides many options for graphically portraying data. Here's what you can do:

- Retrieve data from a wide variety of sources and display it graphically, using charts, line drawings, bit-mapped images, text, and sound.
- Draw artwork such as lines, polygons, and freehand shapes using a full set of editing tools.
- Use PL/SQL to control flow, add interactive features, and animate artwork. Write, edit, and debug your programs within Graphics.
- Make charts and drawings dynamic, by linking them to data sources. Write procedures that update the display to reflect changes in the data.
- Include interactive features for end-users, such as mouse-activated buttons, or use timers to trigger certain operations automatically.
- Choose from 56 pre-defined chart templates that you can customize and reuse for multiple displays.
- Open multiple windows for working on more than one display at a time.
- Store displays in either your file system or the Oracle Server.
- Import and export CGM (ISO, ANSI, or GKS) drawings; BMP, JPEG, and TIFF images; and AIFF–c sound files. Also import PCX and PICT images.
- Provide end-users with a simplified runtime version of the interface for executing Graphics displays.

Graphics provides the following major features to support the functionality described above:

With its bit-mapped point-and-click interface, Graphics enables you to create database-driven graphics without programming. You can create displays using a combination of pull-down menus, dialog boxes, and object-oriented tools, accessing data from a variety of sources.
| Layout Editor | Graphics provides a layout editor with a complete set of design tools for creating drawings, charts, and text. You can import images such as scanned photographs from other sources and include them in your displays, or export drawings to include in written documents prepared with popular desktop publishing systems. Graphics supports many standard graphics formats. |
| Intelligent Defaulting | The intelligent defaulting functionality within Graphics enables you to quickly develop graphics modules for your applications. To build a chart, you simply create a query, choose one of 56 pre-defined chart templates, and position the chart on the layout. Graphics draws the chart for you and automatically labels it according to the data you selected. |
| Extensible Architecture | If a display requires special logic and flow control, you can add it using Oracle’s procedural language, PL/SQL. An embedded PL/SQL editor lets you create, debug, and reuse your program units without leaving Graphics. Program units you create with Graphics can be shared by other products such as Forms and Reports, and integrated into larger applications. |
| Portability Across Platforms | Since Graphics is built using Oracle Toolkit, it will conform to the native look-and-feel of your host environment. You can create and run displays on bit-mapped environments including Microsoft Windows, OSF/Motif, and Macintosh, with identical functionality and compatibility across platforms. End-users working on character or block mode devices can use a Batch executable to direct the output of an Graphics display to a printer or file. Modules you create with Graphics are hardware-independent, which means you can design them on any platform and then distribute them over a wide range of hardware and operating systems. A separate Runtime executable provides end-users with a simplified environment for executing Graphics displays. |
| Integration with Other Oracle Products | A database graphics system is one component of the total information strategy. Graphics is part of Oracle’s Developer/2000, a suite of tools, each designed to address a specific information need. Thus, displays you create with Graphics can be integrated with modules from Forms and Reports to create larger database applications. |
| Support for Non-Oracle Data Sources | Graphics works with SQL*Connect to support the use of IBM’s DB2 and SQL/DS, as well as many other databases. It also reads data from Microsoft Excel (SYLK), Lotus 1–2–3 (WKS), and ASCII–delimited (PRN) files. |
Graphics provides a layout editor with a complete set of design tools for creating drawings, charts, and text. You can import images such as scanned photographs from other sources and include them in your displays, or export drawings to include in written documents prepared with popular desktop publishing systems. Graphics supports many standard graphics formats.

The intelligent defaulting functionality within Graphics enables you to quickly develop graphics modules for your applications. To build a chart, you simply create a query, choose one of 56 pre-defined chart templates, and position the chart on the layout. Graphics draws the chart for you and automatically labels it according to the data you selected.

If a display requires special logic and flow control, you can add it using Oracle's procedural language, PL/SQL. An embedded PL/SQL editor lets you create, debug, and reuse your program units without leaving Graphics. Program units you create with Graphics can be shared by other products such as Forms and Reports, and integrated into larger applications.

Since Graphics is built using Oracle Toolkit, it will conform to the native look-and-feel of your host environment. You can create and run displays on bit-mapped environments including Microsoft Windows, OSF/Motif, and Macintosh, with identical functionality and compatibility across platforms. End-users working on character or block mode devices can use a Batch executable to direct the output of a Graphics display to a printer or file.

Modules you create with Graphics are hardware-independent, which means you can design them on any platform and then distribute them over a wide range of hardware and operating systems. A separate Runtime executable provides end-users with a simplified environment for executing Graphics displays.

A database graphics system is one component of the total information strategy. Graphics is part of Oracle's Developer/2000, a suite of tools, each designed to address a specific information need. Thus, displays you create with Graphics can be integrated with modules from Forms and Reports to create larger database applications.

Graphics works with SQL*Connect to support the use of IBM's DB2 and SQL/DS, as well as many other databases. It also reads data from Microsoft Excel (SYLK), Lotus 1–2–3 (WKS), and ASCII–delimited (PRN) files.
New Features with this Release

With the 2.5 release of Graphics, you’ll find the following new features:

• Designer User Interface
  – object navigator
  – direct editing of chart properties on the layout
  – support for tabbed property sheets
  – object-specific menus activated by right mouse button
  – new color and pattern palettes

• Chart Capabilities
  – quick interface for creating drill-down charts
  – ability to plot subcategory data in a break chart
  – dynamic chart updating
  – data–driven formatting

• Runtime Component
  – integration

Object Navigator

The new Object Navigator provides a hierarchical, icon-based display of all Graphics objects, such as displays, libraries, program units, queries, etc.

The following functions are supported in the Object Navigator:

• expand or collapse nodes in the hierarchy to view objects and object relationships
• copy or move objects from one node to another by dragging them with the mouse
• edit object properties
• rename, create, and delete objects

For more information about the Object Navigator, see chapter “Using the Object Navigator” on page 5–1.

Direct Editing of Charts

In addition to the Chart Template Editor, which can be used to modify the format of a chart or set of charts, you can modify a chart’s appearance directly in the Layout Editor.

Tabbed Property Sheets

Graphics 2.5 supports Microsoft Windows Chicago–style property sheets with tabbed “property pages.”
New Features with this Release

With the 2.5 release of Graphics, you’ll find the following new features:

- Designer User Interface
  - object navigator
  - direct editing of chart properties on the layout
  - support for tabbed property sheets
  - object-specific menus activated by right mouse button
  - new color and pattern palettes

- Chart Capabilities
  - quick interface for creating drill-down charts
  - ability to plot subcategory data in a break chart
  - dynamic chart updating
  - data–driven formatting

- Runtime Component
  - integration

The new Object Navigator provides a hierarchical, icon-based display of all Graphics objects, such as displays, libraries, program units, queries, etc.

The following functions are supported in the Object Navigator:

- expand or collapse nodes in the hierarchy to view objects and object relationships
- copy or move objects from one node to another by dragging them with the mouse
- edit object properties
- rename, create, and delete objects

For more information about the Object Navigator, see chapter “Using the Object Navigator” on page 5–1.

In addition to the Chart Template Editor, which can be used to modify the format of a chart or set of charts, you can modify a chart’s appearance directly in the Layout Editor.

Graphics 2.5 supports Microsoft Windows Chicago–style property sheets with tabbed “property pages.”
Pop-up Menus
In addition to the main menu, you can display a pop-up menu for a selected object using the right mouse button. For example, selecting a chart in the Layout Editor and then pressing the right mouse button will display a menu of Undo, Cut, Copy, Paste, Clear, Properties, Axes, Frame, etc.

The pop-up menus can be used in both the Layout editor and the Object Navigator.

Note: Not all systems provide support for a multiple–button mouse. See your Graphics operating system documentation for more information on your options for a single–button mouse.

New Color and Pattern Palettes
The Color and Pattern palettes have been modified for ease of use. In addition to a new format for selecting colors and patterns for edge, fill, and text, you can now “tear off” the palettes and move them around the Layout Editor.

For more information about using the color and pattern palettes, see “Setting Patterns and Colors for Objects” on page 4 – 25

Drill-down Charts
You can create dynamic master–detail relationships between charts using a simple property sheet. No programming or direct entry of PL/SQL code is necessary.

For more information about drill-down charts, see “Using Drill-down Charts” on page 8 – 26.

Plotting Subcategory Data
You can now create a chart, known as a “break chart,” which plots two categories of data along the chart’s axes. The category sets are grouped along the axes to provide a clear delineation between them.

For more information about break charts, see “Creating a Break Chart” on page 8 – 13.

Event-driven or Timed Update of Charts
You can use timers to update a chart at regular intervals. Such a chart might plot the value of a given stock over a course of time. You can further define the chart to scroll as the data is updated so it retains the same size on the display. This type of chart is referred to as a “strip chart.”

In addition, you can specify that a chart is updated based on an event, such as when the display is opened, when the display is closed, when the associated query is executed, or upon demand.

For more information about timers, see “Using Timer Triggers” on page 11 – 9 in chapter “Using Triggers”.
In addition to the main menu, you can display a pop-up menu for a selected object using the right mouse button. For example, selecting a chart in the Layout Editor and then pressing the right mouse button will display a menu of Undo, Cut, Copy, Paste, Clear, Properties, Axes, Frame, etc.

The pop-up menus can be used in both the Layout editor and the Object Navigator.

Note: Not all systems provide support for a multiple–button mouse. See your Graphics operating system documentation for more information on your options for a single–button mouse.

The Color and Pattern palettes have been modified for ease of use. In addition to a new format for selecting colors and patterns for edge, fill, and text, you can now “tear off” the palettes and move them around the Layout Editor.

For more information about using the color and pattern palettes, see “Setting Patterns and Colors for Objects” on page 4 – 25

You can create dynamic master–detail relationships between charts using a simple property sheet. No programming or direct entry of PL/SQL code is necessary.

For more information about drill-down charts, see “Using Drill-down Charts” on page 8 – 26.

You can now create a chart, known as a “break chart,” which plots two categories of data along the chart’s axes. The category sets are grouped along the axes to provide a clear delineation between them.

For more information about break charts, see “Creating a Break Chart” on page 8 – 13.

You can use timers to update a chart at regular intervals. Such a chart might plot the value of a given stock over a course of time. You can further define the chart to scroll as the data is updated so it retains the same size on the display. This type of chart is referred to as a “strip chart.”

In addition, you can specify that a chart is updated based on an event, such as when the display is opened, when the display is closed, when the associated query is executed, or upon demand.

For more information about timers, see “Using Timer Triggers” on page 11 – 9 in chapter “Using Triggers.”
For more information about query triggers, see “Query Execution” on page 11 – 2 in chapter “Using Triggers”.

Data Driven Formatting

Using format triggers, you can create data conditions that control the appearance of specified elements in your chart. For example, you could set a condition for a bar chart such that if a value falls below a certain point, that bar should be displayed in a different color than the rest.

For more information about format triggers, see “Using Format Triggers” on page 11 – 8 in chapter “Using Triggers”.

Product Integration

Graphics Runtime displays can be embedded within other Developer/2000 components such as Forms and Reports. An embedded Graphics display can pass data to its container application as well as receive data from that container application.

An embedded Graphics display is dynamic and can be manipulated with the full functionality of Graphics Runtime. For instance, it will respond to mouse events.

In addition to Oracle’s own internal product integration, Graphics acts as a Microsoft Windows OLE2 server. This allows you to embed a display within an OLE2 container application, such as Visual Basic and Microsoft Excel.

For more information about product integration, see “Integration” on page 16 – 1.

Executables

Executables are binary files that you use to invoke the Graphics Designer, Runtime, and Batch mode components. You can run executables in two ways: from a bit-mapped environment by invoking their icons, or from a command line prompt using one word names along with arguments.


The three Graphics executables are described below:

- Designer
- Runtime
- Batch
For more information about query triggers, see "Query Execution" on page 11–2 in chapter "Using Triggers".

Using format triggers, you can create data conditions that control the appearance of specified elements in your chart. For example, you could set a condition for a bar chart such that if a value falls below a certain point, that bar should be displayed in a different color than the rest.

For more information about format triggers, see "Using Format Triggers" on page 11–8 in chapter "Using Triggers".

Graphics Runtime displays can be embedded within other Developer/2000 components such as Forms and Reports. An embedded Graphics display can pass data to its container application as well as receive data from that container application.

An embedded Graphics display is dynamic and can be manipulated with the full functionality of Graphics Runtime. For instance, it will respond to mouse events.

In addition to Oracle's own internal product integration, Graphics acts as a Microsoft Windows OLE2 server. This allows you to embed a display within an OLE2 container application, such as Visual Basic and Microsoft Excel.

For more information about product integration, see "Integration" on page 16–1.

Executables are binary files that you use to invoke the Graphics Designer, Runtime, and Batch mode components. You can run executables in two ways: from a bit-mapped environment by invoking their icons, or from a command line prompt using one word names along with arguments.


The three Graphics executables are described below:

- Designer
- Runtime
- Batch
### Designer Executable
You’ll use the Designer executable to invoke the Graphics Designer to create and work with displays. As an application developer, you’ll use this executable most often. Within the Designer, you can use a debugger that simulates the runtime environment for testing and debugging your displays.

### Runtime Executable
Intended for end-users, the Runtime executable invokes the Graphics runtime environment, which is used for running displays only.

### Batch Executable
You’ll use the Batch executable to invoke the Graphics Batch mode to direct the output of a display to a printer, file, or database.

### Installing Graphics on Your System
Before you begin the Tutorial, you must have the following items installed on your system:
- Oracle Server (Version 6.0 or later) or another database
- Graphics Release 2.5
- demo database tables and sample files

The following information for you and your database administrator is divided into two sections:
- questions to ask your database administrator
- guidelines for database administrators

### Questions to Ask Your Database Administrator
1. Is the Oracle Server (Version 6.0 or later) or another database installed?
2. What username, password, and database specification should I use to connect to the database?
3. Is Graphics 2.5 installed?
4. Are the demo database tables and sample files loaded?

### Guidelines for Database Administrators
1. Install the database, if applicable. For more information about installing the Oracle Server, see the Oracle Server documentation for your operating system.
You'll use the Designer executable to invoke the Graphics Designer to create and work with displays. As an application developer, you'll use this executable most often. Within the Designer, you can use a debugger that simulates the runtime environment for testing and debugging your displays.

Intended for end-users, the Runtime executable invokes the Graphics runtime environment, which is used for running displays only. You'll use the Batch executable to invoke the Graphics Batch mode to direct the output of a display to a printer, file, or database.

Before you begin the Tutorial, you must have the following items installed on your system:

- Oracle Server (Version 6.0 or later) or another database
- Graphics Release 2.5
- demo database tables and sample files

The following information for you and your database administrator is divided into two sections:

- questions to ask your database administrator
- guidelines for database administrators

1. Is the Oracle Server (Version 6.0 or later) or another database installed?
2. What username, password, and database specification should I use to connect to the database?
3. Is Graphics 2.5 installed?
4. Are the demo database tables and sample files loaded?

1. Install the database, if applicable. For more information about installing the Oracle Server, see the Oracle Server documentation for your operating system.
2. Set up logon accounts for users. For more information about logon accounts, see the Oracle7 Server Administrator’s Guide.

3. Install Graphics 2.5. For more information about installing Graphics, see the Graphics documentation for your operating system.

4. Install the demo database tables. For more information about installing the demo tables, see the Graphics documentation for your operating system.

5. Load the sample files in the appropriate directory. For more information about loading sample files, see the Graphics documentation for your operating system.

Using the Help System

The help system is an online document facility you can access while you’re working. To use this feature, the help system files need to be installed in advance. Also, the way in which you access online help and the online help system used will depend on your operating system. If you need assistance, consult your system administrator.

For systems running Microsoft Windows, Microsoft Windows Help will be used as the help system. On all other systems, Oracle Help will be used as the help system. This section describes how to use Oracle Help. For information on how to use Microsoft Windows Help, see your Microsoft Windows documentation.

When the help system window appears, you can scroll the text line by line, or move between subtopics using the Previous and Next buttons. You can also view and select topics or subtopics from the Table of Contents and Index. If you aren’t sure which topic you want, you can enter a word or phrase to search for the locations of that string.

1. Choose the Find button to show the Help Search window.

2. Place your cursor in the Find field and type mode.

3. Choose the Find Word button to execute the search.

   A list of the word’s locations appears. If the word appears more than once within a chapter, each occurrence will be listed.

4. Select one of the locations and choose the Go To button.

   The part of the chapter containing the word is shown.

5. Choose the Cancel button to close the help system.
2. Set up logon accounts for users. For more information about logon accounts, see the Oracle7 Server Administrator's Guide.

3. Install Graphics 2.5. For more information about installing Graphics, see the Graphics documentation for your operating system.

4. Install the demo database tables. For more information about installing the demo tables, see the Graphics documentation for your operating system.

5. Load the sample files in the appropriate directory. For more information about loading sample files, see the Graphics documentation for your operating system.

Using the Help System

The help system is an online document facility you can access while you're working. To use this feature, the help system files need to be installed in advance. Also, the way in which you access online help and the online help system used will depend on your operating system. If you need assistance, consult your system administrator.

For systems running Microsoft Windows, Microsoft Windows Help will be used as the help system. On all other systems, Oracle Help will be used as the help system. This section describes how to use Oracle Help.

For information on how to use Microsoft Windows Help, see your Microsoft Windows documentation.

When the help system window appears, you can scroll the text line by line, or move between subtopics using the Previous and Next buttons. You can also view and select topics or subtopics from the Table of Contents and Index. If you aren't sure which topic you want, you can enter a word or phrase to search for the locations of that string.

1. Choose the Find button to show the Help Search window.
2. Place your cursor in the Find field and type mode.
3. Choose the Find Word button to execute the search. A list of the word's locations appears. If the word appears more than once within a chapter, each occurrence will be listed.
4. Select one of the locations and choose the Go To button. The part of the chapter containing the word is shown.
5. Choose the Cancel button to close the help system.
Also available via the Help menu are two instructional aids: a Quick Tour, and Cue Cards.

Selecting Help—>Quick Tour invokes a series of screens that provide a brief introduction to Graphics, including explanations of basic concepts and instructions on how to accomplish basic tasks.

Selecting Help—>Cue Cards displays a menu of commonly–performed tasks. Choose what you want to accomplish, and you are shown a series of steps that guide you through your selected task.

Resolving Errors

If you get an error message in Graphics, you need to find out which product or component generated the error so that you can locate the appropriate instructions on how to resolve it. You do this by noting the prefix and code number of the error message.

The prefix of the error message tells you the product or component that generated the error. For example, the prefix PLS tells you it’s a PL/SQL error. Once you know the prefix, you can look in the document that lists error messages for that product or component. By looking up the code number of the error message, you can find a description of what caused the error and how to resolve it.

Also available via the Help menu are two instructional aids: a Quick Tour, and Cue Cards.

Selecting Help —> Quick Tour invokes a series of screens that provide a brief introduction to Graphics, including explanations of basic concepts and instructions on how to accomplish basic tasks.

Selecting Help —> Cue Cards displays a menu of commonly–performed tasks. Choose what you want to accomplish, and you are shown a series of steps that guide you through your selected task.

Resolving Errors

If you get an error message in Graphics, you need to find out which product or component generated the error so that you can locate the appropriate instructions on how to resolve it. You do this by noting the prefix and code number of the error message.

The prefix of the error message tells you the product or component that generated the error. For example, the prefix PLS tells you it's a PL/SQL error. Once you know the prefix, you can look in the document that lists error messages for that product or component. By looking up the code number of the error message, you can find a description of what caused the error and how to resolve it.

his chapter is designed to introduce you to the basic operations of Graphics and includes the following lessons:

Lesson 1  Create a pie chart – 2 – 3
Lesson 2  Create a column chart – 2 – 8
Lesson 3  Modify the charts – 2 – 11
Lesson 4  Create a drill-down relationship – 2 – 13
Lesson 5  Create a format trigger – 2 – 18
Lesson 6  Add a text object – 2 – 22
Lesson 7  Create a break chart – 2 – 24
Lesson 8  Create a query filter – 2 – 26
Lesson 9  Create a second layer – 2 – 31
Lesson 10 Create a button procedure – 2 – 34
Lesson 11 View the Results – 2 – 39
This chapter is designed to introduce you to the basic operations of Graphics and includes the following lessons:

- Create a pie chart – 2 – 3
- Create a column chart – 2 – 8
- Modify the charts – 2 – 11
- Create a drill-down relationship – 2 – 13
- Create a format trigger – 2 – 18
- Add a text object – 2 – 22
- Create a break chart – 2 – 24
- Create a query filter – 2 – 26
- Create a second layer – 2 – 31
- Create a button procedure – 2 – 34

Lesson 1
Lesson 2
Lesson 3
Lesson 4
Lesson 5
Lesson 6
Lesson 7
Lesson 8
Lesson 9
Lesson 10
Lesson 11
Before You Begin

The lessons in this tutorial use the sample database tables for the Summit Sporting Goods company. These tables are created with the SUMMIT2.SQL script that is shipped on your distribution media. You may install these tables locally if you have a local Oracle database. See your Developer/2000 Installation Guide for instructions on running this script.

If you do not have a local Oracle database, contact your DBA to make sure these tables have been installed on a server, and to find out how you can access them.

For this tutorial, you’ll need to supply a userid and password as well as the connect string for the sample database (whether local or remote). Your DBA should be able to supply you with this information for the remote database.

Additional Information: See your Developer/2000 Installation Guide for system specific information such as installing and starting up Graphics on your platform, and how to set up the Graphics sample database tables.

Tutorial Scenario

In this tutorial, you will create a display that graphically charts inventory levels for products sold by Summit Sporting Goods. Summit maintains five warehouses worldwide. Each warehouse stores inventory of every product sold by Summit.

Once you have created and generated the display in the Graphics Designer, the display will be used in conjunction with Graphics Runtime by Summit’s sales and manufacturing divisions to help track and maintain inventory levels worldwide.

These lessons are designed to be performed in order, but you do not have to perform all lessons. If you decide to skip a particular lesson, you may experience inconsistencies in your results. You can perform all the lessons in one session, or set your own pace. At any time, you may skip to Lesson 11 on page 2–39 to view the display in runtime simulation.
OS 2 – 2 Graphics Developer's Guide

Before You Begin

The lessons in this tutorial use the sample database tables for the Summit Sporting Goods company. These tables are created with the SUMMIT2.SQL script that is shipped on your distribution media. You may install these tables locally if you have a local Oracle database. See your Developer/2000 Installation Guide for instructions on running this script.

If you do not have a local Oracle database, contact your DBA to make sure these tables have been installed on a server, and to find out how you can access them.

For this tutorial, you'll need to supply a userid and password as well as the connect string for the sample database (whether local or remote). Your DBA should be able to supply you with this information for the remote database.

Additional Information:
See your Developer/2000 Installation Guide for system specific information such as installing and starting up Graphics on your platform, and how to set up the Graphics sample database tables.

Tutorial Scenario

In this tutorial, you will create a display that graphically charts inventory levels for products sold by Summit Sporting Goods. Summit maintains five warehouses worldwide. Each warehouse stores inventory of every product sold by Summit.

Once you have created and generated the display in the Graphics Designer, the display will be used in conjunction with Graphics Runtime by Summit's sales and manufacturing divisions to help track and maintain inventory levels worldwide.

These lessons are designed to be performed in order, but you do not have to perform all lessons. If you decide to skip a particular lesson, you may experience inconsistencies in your results. You can perform all the lessons in one session, or set your own pace. At any time, you may skip to Lesson 11 on page 2 – 39 to view the display in runtime simulation.
Lesson 1: Create a Pie Chart

This lesson describes how to create a pie chart that shows the distribution of inventory among Summit Sporting Goods’ five warehouses. This chart will present an overview of total inventory distribution for Summit. In subsequent lessons, you’ll provide more detailed product inventory information.

Pie charts are optimal for comparing percentages or ratios.

Start Graphics

Double-click on the Graphics icon on your desktop, or type the appropriate command at your operating system prompt to start Graphics.

See your *Developer/2000 Installation Guide* for more information on how to start Graphics on your platform.

Notice that an empty display is automatically opened for you when you start Graphics. This is where you will create your chart.

Connect to the Database

Although you can create charts from spreadsheet files such as PRN, SYLK, or WKS format, in most cases your charts will be based on data stored in database tables. You must be connected to the appropriate database in order to query it.

1. Choose **File**—>**Connect** to show the Connect dialog box.

   Check with your database administrator if you are unsure of your database userid, password, or database connect string for the database that contains the Summit Sporting Goods sample tables.

2. Type your username in the Username field.

3. Type your database password in the Password field. Your entry will not be echoed on the screen.

4. Type the database connect string in the Database field. Be sure to include the network protocol, network node, and database name as in the following example:

   \[t:boston:payroll\]

   In the above example, \(t:\) represents the network protocol, \(boston:\) is the network node name, and \(payroll\) is the name of the database.

5. Choose the Connect button to proceed with the connection.

   Unless there are errors, the Connect dialog will close when your connection to the database is established.
Lesson 1: Create a Pie Chart

This lesson describes how to create a pie chart that shows the distribution of inventory among Summit Sporting Goods' five warehouses. This chart will present an overview of total inventory distribution for Summit. In subsequent lessons, you’ll provide more detailed product inventory information.

Pie charts are optimal for comparing percentages or ratios. Double-click on the Graphics icon on your desktop, or type the appropriate command at your operating system prompt to start Graphics.

See your Developer/2000 Installation Guide for more information on how to start Graphics on your platform.

Notice that an empty display is automatically opened for you when you start Graphics. This is where you will create your chart.

Although you can create charts from spreadsheet files such as PRN, SYLK, or WKS format, in most cases your charts will be based on data stored in database tables. You must be connected to the appropriate database in order to query it.

1. Choose File —> Connect to show the Connect dialog box.
2. Type your username in the Username field.
3. Type your database password in the Password field. Your entry will not be echoed on the screen.
4. Type the database connect string in the Database field. Be sure to include the network protocol, network node, and database name as in the following example:

   t:boston:payroll

   In the above example, t: represents the network protocol, boston: is the network node name, and payroll is the name of the database.
5. Choose the Connect button to proceed with the connection.

Unless there are errors, the Connect dialog will close when your connection to the database is established.
Note: You can specify your connect string on the command line using the USERID keyword. See the “Executables” chapter in the Graphics Reference Manual for more information.

Setting Preferences

The charts we will create in this tutorial all contain numbers. Graphics uses a default number format unless you specify a different format for the current session in your preferences.

1. Select Tools—>Tools Options to show the Tools Options property sheet.

2. Choose the Number button to show the Number Format dialog.
   You can enter a variety of number formats and save them to your preferences file for future use.

3. If there is a number format already in the list that does not use decimals or commas, select it from the list so it appears in the Format field.

4. If there are no number formats in the list, type 9999990 in the Format field and choose the Add button to add the format to your preferences list.
   The number format you typed should still appear in the Format field.

5. Select the OK button to accept the new number format selection.

6. Select the Save button to save this setting to your preferences file.
Note:

You can specify your connect string on the command line using the USERID keyword. See the "Executables" chapter in the Graphics Reference Manual for more information.

The charts we will create in this tutorial all contain numbers. Graphics uses a default number format unless you specify a different format for the current session in your preferences.

1. Select Tools —> Tools Options to show the Tools Options property sheet.

2. Choose the Number button to show the Number Format dialog. You can enter a variety of number formats and save them to your preferences file for future use.

3. If there is a number format already in the list that does not use decimals or commas, select it from the list so it appears in the Format field.

4. If there are no number formats in the list, type 9999990 in the Format field and choose the Add button to add the format to your preferences list. The number format you typed should still appear in the Format field.

5. Select the OK button to accept the new number format selection.

6. Select the Save button to save this setting to your preferences file.
Create the Chart

Creating a chart consists of two basic tasks: creating the query, and defining how the chart will be drawn. In Graphics, you can create your queries independently of your charts, or you can create both in one process. In this lesson, you’ll create them both in one process.

Create the Query

When you create the query as part of the chart creation process, you will use the New Query dialog to specify the query. You can also use this dialog to name the query and view the data returned by it before creating the chart.

1. Choose Chart—>Create Chart to show the New Query dialog.

   If you do not see the Chart menu, click in the Layout Editor to activate it. The Chart menu should appear.

   The new query is named query0 by default, where x is a number corresponding to the order in which you create the queries. Because this is the first query you have created for this display, it is named query0.

2. Double-click in the name field to select the default name.
Create the Chart

Create the Query

This format will be used for all charts you create until you specify another format (either in the Tools Options property sheet, or directly on the Layout editor).

Creating a chart consists of two basic tasks: creating the query, and defining how the chart will be drawn. In Graphics, you can create your queries independently of your charts, or you can create both in one process. In this lesson, you'll create them both in one process.

When you create the query as part of the chart creation process, you will use the New Query dialog to specify the query. You can also use this dialog to name the query and view the data returned by it before creating the chart.

1. Choose Chart —> Create Chart to show the New Query dialog.

If you do not see the Chart menu, click in the Layout Editor to activate it. The Chart menu should appear.

The new query is named queryx by default, where x is a number corresponding to the order in which you create the queries. Because this is the first query you have created for this display, it is named query0.

2. Double-click in the name field to select the default name.
3. Type the new name *pie* to rename the query.
   If you will be creating multiple queries, as you will in this tutorial, it is a good idea to give each of them a name indicative of their purpose. This makes it easier to remember which query performs which action if you need to reference them later.

4. Click in the SQL Statement field and type the following SQL SELECT statement:

   ```sql
   SELECT warehouse_id, sum(amount_in_stock) "Inventory"
   FROM s_inventory
   GROUP BY warehouse_id
   ```

   Notice that you can also specify a query from a file such as PRN, SYLK, or WKS format.

5. Choose the Execute button to execute the query.

   The query is executed and the Data property page is presented so you can view the results in a table format.

   This query has returned five rows and two columns of data. One column represents the warehouse identification number, the other represents the inventory level totals for a given warehouse.

6. Choose the OK button to close the New Query dialog.

   The Chart property sheet appears.

**Define the Chart**

Once you have created a query, you will use the Chart property sheet to define how your query data is charted. The Chart property sheet also allows you to assign a name and title to the chart you are creating.

1. Type `compare_inv` in the Name field.

   This name will be assigned to the chart object in the Object Navigator, and will also be used if you reference the chart in any PL/SQL program units.

2. Type `Total Inventory/Warehouse` in the Title field.

   This title will appear above the chart when the chart is drawn in the layout.

3. Select the pie chart button in the Type field.

   Notice that the available subtypes in the Subtypes field change to reflect the type of chart you have selected. The subtypes allow you to select such options as shadowing.

4. Choose the OK button to create the chart.
3. Type the new name pie to rename the query.

If you will be creating multiple queries, as you will in this tutorial, it is a good idea to give each of them a name indicative of their purpose. This makes it easier to remember which query performs which action if you need to reference them later.

4. Click in the SQL Statement field and type the following SQL SELECT statement:

   `SELECT warehouse_id, sum(amount_in_stock) AS "Inventory" FROM s_inventory GROUP BY warehouse_id`

   Notice that you can also specify a query from a file such as PRN, SYLK, or WKS format.

5. Choose the Execute button to execute the query.

   The query is executed and the Data property page is presented so you can view the results in a table format.

   This query has returned five rows and two columns of data. One column represents the warehouse identification number; the other represents the inventory level totals for a given warehouse.

6. Choose the OK button to close the New Query dialog.

   The Chart property sheet appears.

   Once you have created a query, you will use the Chart property sheet to define how your query data is charted. The Chart property sheet also allows you to assign a name and title to the chart you are creating.

   1. Type `compare_inv` in the Name field.

      This name will be assigned to the chart object in the Object Navigator, and will also be used if you reference the chart in any PL/SQL program units.

   2. Type `Total Inventory/Warehouse` in the Title field.

      This title will appear above the chart when the chart is drawn in the layout.

   3. Select the pie chart button in the Type field.

      Notice that the available subtypes in the Subtypes field change to reflect the type of chart you have selected. The subtypes allow you to select such options as shadowing.

   4. Choose the OK button to create the chart.
Your pie chart appears in the display area. The chart is surrounded by control points that you can use to resize the chart. When you see the control points around any object in your display, that object is in “select” mode. You can move it, resize it, or show its property sheet. You can deselect an object by clicking elsewhere in the Layout editor.

Your chart will look something like this:

![Pie Chart](image)

Notice that Graphics randomly selects different colors or patterns for each pie slice. Each pie slice is labeled according to the warehouse identification number it represents.

**Move and Resize the Chart**

Unless you specify a chart area before creating the chart, Graphics uses its own default values for the chart size and location. Since you’ll be creating more charts and objects in this display, resize and move this one to make room for the rest.

1. Click and drag the selected chart to the upper left corner of the display.
   
   Be sure to drag the chart from inside the bounding box created by the control points. If you drag one of the control points, you may resize the chart instead of moving it.
   
   If this happens inadvertently, choose **Undo** from the pop-up menu (or **Edit—>Undo**) to reverse the change.

2. Click on the lower right corner control point and drag diagonally to resize the pie chart to approximately two inches wide by two inches long.
Your pie chart appears in the display area. The chart is surrounded by control points that you can use to resize the chart. When you see the control points around any object in your display, that object is in "select" mode. You can move it, resize it, or show its property sheet. You can deselect an object by clicking elsewhere in the Layout editor.

Your chart will look something like this:

Notice that Graphics randomly selects different colors or patterns for each pie slice. Each pie slice is labeled according to the warehouse identification number it represents.

Unless you specify a chart area before creating the chart, Graphics uses its own default values for the chart size and location. Since you'll be creating more charts and objects in this display, resize and move this one to make room for the rest.

1. Click and drag the selected chart to the upper left corner of the display. Be sure to drag the chart from inside the bounding box created by the control points. If you drag one of the control points, you may resize the chart instead of moving it.

2. Click on the lower right corner control point and drag diagonally to resize the pie chart to approximately two inches wide by two inches long.
Use the rulers to help you measure the chart size.

3. Choose **Update** from the pop-up menu (or **Chart—>Update Chart**) to redraw the chart after resizing it.

   Sometimes after resizing, a chart can become distorted. Forcing an update of it can refresh its appearance.

**Save the Display**

Choose **File—>Save As—>File System** to save the display to a file. Give the display a name other than the default **DISP1**.

**Investigate the Object Navigator**

Now that you’ve created some objects in your display, try expanding some of the nodes in the Object Navigator to examine the object hierarchy. The Object Navigator is also useful for accessing an object’s property sheet, or for editing an object’s name.

Each node in the Navigator has a “+” to the left of it. If the “+” is highlighted, then that node can be expanded. If the “+” is not highlighted, that node does not contain any subnodes and cannot be expanded. Once a node is expanded, the “+” changes to a “–”.

You can expand a node by clicking directly on the highlighted “+”, or by selecting the node name or node icon, and then choosing the Expand button, or by selecting the node name or the node icon and choosing **Navigator—>Expand**.

Try expanding the Queries node. Notice that your pie query appears.

Watch the Object Navigator throughout the tutorial to see the different objects you create appear.

**Continue or Exit**

Continue with the next lesson or choose **File—>Exit** to end your current Graphics session.

---

**Lesson 2: Create a Column Chart**

In this lesson you will create a column chart that shows more detailed information about the inventory levels in a specific warehouse. Specifically, this chart will show the amount in stock and reorder points for each product in a certain warehouse. Although there are five warehouses, you will concentrate on only one in this lesson.

In Lesson 4 on page 2–13, you will create a drill-down relationship between the two charts that will allow users to control for which warehouse this chart shows information.
Use the rulers to help you measure the chart size.
3.

Choose Update from the pop-up menu (or Chart—>Update Chart)
to redraw the chart after resizing it.
Sometimes after resizing, a chart can become distorted. Forcing an
update of it can refresh its appearance.

Save the Display

Choose File—>Save As—>File System to save the display to a file.
Give the display a name other than the default DISP1.

Investigate the Object
Navigator

Now that you’ve created some objects in your display, try expanding
some of the nodes in the Object Navigator to examine the object
hierarchy. The Object Navigator is also useful for accessing an object’s
property sheet, or for editing an object’s name.
Each node in the Navigator has a “+” to the left of it. If the “+” is
highlighted, then that node can be expanded. If the “+” is not
highlighted, that node does not contain any subnodes and cannot be
expanded. Once a node is expanded, the “+” changes to a “–”.
You can expand a node by clicking directly on the highlighted “+”, or by
selecting the node name or node icon, and then choosing the Expand
button, or by selecting the node name or the node icon and choosing
Navigator—>Expand.
Try expanding the Queries node. Notice that your pie query appears.
Watch the Object Navigator throughout the tutorial to see the different
objects you create appear.

Continue or Exit

Continue with the next lesson or choose File—>Exit to end your current
Graphics session.

Lesson 2: Create a Column Chart
In this lesson you will create a column chart that shows more detailed
information about the inventory levels in a specific warehouse.
Specifically, this chart will show the amount in stock and reorder points
for each product in a certain warehouse. Although there are five
warehouses, you will concentrate on only one in this lesson.
In Lesson 4 on page 2 – 13, you will create a drill-down relationship
between the two charts that will allow users to control for which
warehouse this chart shows information.

2–8

Graphics Developer’s Guide


Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

Create the Chart

In the previous lesson you created a query and a chart in the same process. In this lesson, you will create the query independently of the chart. Once you create a query, you can assign it to a chart at any time.

Create the Query

When you create queries independently of creating a chart, you use the Query property sheet rather than the New Query dialog. The Query property sheet is very similar to the New Query dialog; however, it provides an Options tab, and a query drop-down list of other existing queries you can choose from.

1. Select the Queries node in the Object Navigator.
2. Choose the Create button, or Navigator—>Create, to show the Query property sheet.
3. Edit the name field to name the query column.
4. Click in the SQL Statement field, and type the following SQL SELECT statement:

   SELECT product_id, amount_in_stock, reorder_point
   FROM s_inventory
   WHERE warehouse_id = 101

5. Choose the Execute button to execute the query and display the data returned in the Data property page.
6. Choose the OK button to close the Query property sheet.

   The query is created.

Define the Chart

In this lesson, you will draw out a specific chart area in your display where the chart will be drawn. After the chart is drawn, you can still resize and move it by selecting the chart and using its control points.

1. Select the Chart tool from the Tool palette in the Layout editor.

   Notice that the cursor changes from a pointer to a crosshair.
If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose **File** —> **Close** to close the default display that is presented.
3. Choose **File** —> **Open** or the Open button to open the display you saved at the end of the previous lesson.
4. Choose **File** —> **Connect** to connect to the database.

In the previous lesson you created a query and a chart in the same process. In this lesson, you will create the query independently of the chart. Once you create a query, you can assign it to a chart at any time. When you create queries independently of creating a chart, you use the Query property sheet rather than the New Query dialog. The Query property sheet is very similar to the New Query dialog; however, it provides an Options tab, and a query drop-down list of other existing queries you can choose from.

1. Select the Queries node in the Object Navigator.
2. Choose the Create button, or **Navigator** —> **Create**, to show the Query property sheet.
3. Edit the name field to name the query column.
4. Click in the SQL Statement field, and type the following SQL SELECT statement:
   ```sql
   SELECT product_id, amount_in_stock, reorder_point
   FROM s_inventory
   WHERE warehouse_id = 101
   ```
5. Choose the Execute button to execute the query and display the data returned in the Data property page.
6. Choose the OK button to close the Query property sheet. The query is created.

In this lesson, you will draw out a specific chart area in your display where the chart will be drawn. After the chart is drawn, you can still resize and move it by selecting the chart and using its control points.

1. Select the Chart tool from the Tool palette in the Layout editor. Notice that the cursor changes from a pointer to a crosshair.
2. In the layout, use your mouse to drag out a region underneath the pie chart.

Place the cursor roughly 1/4 inch from the left edge of the display, four inches down from the top of the display. Use the rulers and the position indicators in the status bar to determine your position.

Drag out a region approximately eight inches across and three and half inches down. Use the rulers in the Layout editor to measure your chart area.

When you release the mouse button, the Chart Genie appears instead of the the New Query dialog.

The Chart Genie appears only when you have existing queries from which Graphics can create a chart.

3. Select the column query from the Existing Queries drop-down list.

Notice that all of your existing queries appear in this drop-down list. You could also have selected the New Query radio button to create a new query.

4. Choose the OK button to accept the query and close the Chart Genie.

The Chart property sheet appears.

5. Type warehouse_inv in the Name field.

6. Type Inventory Levels by Product in the Title field.

The default chart type, column, is selected for you.

7. Choose the OK button to create the column chart.

The chart appears in the display area. Notice that Graphics automatically gives the two types of data (amount in stock and reorder point) different colors and groups the related data along the X-axis.

This is a rather large chart. You can use the horizontal and vertical scrollbars in the Layout editor to view the chart. You may need to resize it.

**Save the Display**  
Choose File—>Save or the Save button to save your changes to the display.

**Continue or Exit**  
Continue with the next lesson or choose File—>Exit to end your current Graphics session.
2. In the layout, use your mouse to drag out a region underneath the pie chart. Place the cursor roughly 1/4 inch from the left edge of the display, four inches down from the top of the display. Use the rulers and the position indicators in the status bar to determine your position. Drag out a region approximately eight inches across and three and half inches down. Use the rulers in the Layout editor to measure your chart area. When you release the mouse button, the Chart Genie appears instead of the New Query dialog.

The Chart Genie appears only when you have existing queries from which Graphics can create a chart.

3. Select the column query from the Existing Queries drop-down list. Notice that all of your existing queries appear in this drop-down list. You could also have selected the New Query radio button to create a new query.

4. Choose the OK button to accept the query and close the Chart Genie.

The Chart property sheet appears.

5. Type `warehouse_inv` in the Name field.

6. Type `Inventory Levels by Product` in the Title field. The default chart type, column, is selected for you.

7. Choose the OK button to create the column chart. The chart appears in the display area. Notice that Graphics automatically gives the two types of data (amount in stock and reorder point) different colors and groups the related data along the X–axis.

This is a rather large chart. You can use the horizontal and vertical scrollbars in the Layout editor to view the chart. You may need to resize it.

Choose `File —> Save` or the Save button to save your changes to the display.

Continue with the next lesson or choose `File —> Exit` to end your current Graphics session.
Lesson 3: Modify the Charts

Now that you’ve created the two charts, let’s modify their appearance to make them look a little nicer. When Graphics creates a chart, it uses default settings for all display aspects of the chart. You can change these default settings directly in the Layout editor.

Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

Customize the Column Chart

You can edit the appearance of your charts directly in the layout. In this section, you’ll define custom axis labels for your chart, and change the number format of the chart’s tick labels.

Assign Custom Axis Labels

Use the Axis property sheet to specify settings for chart elements that belong to one of the chart axes. You can toggle between available axes using the Axis drop-down list in the upper left corner of the property sheet.

1. Select the column chart.
2. Choose Axes from the pop-up menu to show the Axis property sheet.
   Alternatively, you can choose Chart—>Axes to show the Axis property sheet.
Now that you've created the two charts, let's modify their appearance to make them look a little nicer. When Graphics creates a chart, it uses default settings for all display aspects of the chart. You can change these default settings directly in the Layout editor.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

You can edit the appearance of your charts directly in the layout. In this section, you'll define custom axis labels for your chart, and change the number format of the chart's tick labels.

Use the Axis property sheet to specify settings for chart elements that belong to one of the chart axes. You can toggle between available axes using the Axis drop-down list in the upper left corner of the property sheet.

1. Select the column chart.
2. Choose Axes from the pop-up menu to show the Axis property sheet.
  Alternatively, you can choose Chart —> Axes to show the Axis property sheet.
3. Select the X-axis in the drop-down list in the upper left corner, if it is not already selected.

4. Type Product ID in the Custom Label field.
   The default label for the chart axes are the names of the columns you select in your SQL query. These column names are assigned when the table is created and are not always formatted well.

5. Choose the Apply button to apply the change to the X axis.

6. Use the Axis drop-down list to select the Y1 axis.

7. Type Inventory in the Custom Label field.

8. Choose the OK button to apply your changes and close the Axis property sheet.
3. Select the X–axis in the drop-down list in the upper left corner, if it is not already selected.
4. Type Product ID in the Custom Label field.

The default label for the chart axes are the names of the columns you select in your SQL query. These column names are assigned when the table is created and are not always formatted well.

5. Choose the Apply button to apply the change to the X axis.
6. Use the Axis drop-down list to select the Y1 axis.
7. Type Inventory in the Custom Label field.
8. Choose the OK button to apply your changes and close the Axis property sheet.
Number formats other than the default, are stored in the Tools Options property sheet. You can select a preferred number format from a pre-defined list, or edit an individual number format directly on the Layout editor.

1. Select the column chart.
2. Select one of the Inventory Level tick labels along the Y axis.
   All of the Inventory Level tick labels should appear in select mode.
3. Choose Number Format from the pop-up menu to show the Number Format dialog.
4. Delete the current format from the Format field.
5. Type the format 999,999,990 in the Format field.
6. Choose the Add button to add this new number format to the list.
   This newly added number format appears in the Number Format list.
7. Choose the OK button to apply your selection and close the Number Format dialog.
   The new number format should be reflected in the chart.

Choose File—>Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File—>Exit to end your current Graphics session.

Lesson 4: Create a Drill-down Relationship

So far you’ve created a pie chart that shows some very high-level general information for all of Summit’s warehouses, and a column chart that shows some detailed information for one specific warehouse. The charts contain one common item of data: the warehouse identification number. Now you’ll dynamically link the two charts together using this common data item.

Remember that the column chart uses a query that refers to a specific warehouse identification number in the WHERE clause. However, rather than creating five separate charts, one for each warehouse, you’ll create a parameter, a global variable, to represent the value of WAREHOUSE_ID in the column query WHERE clause.
Number formats other than the default, are stored in the Tools Options property sheet. You can select a preferred number format from a pre-defined list, or edit an individual number format directly on the Layout editor.

1. Select the column chart.
2. Select one of the Inventory Level tick labels along the Y axis. All of the Inventory Level tick labels should appear in select mode.
3. Choose Number Format from the pop-up menu to show the Number Format dialog.
4. Delete the current format from the Format field.
5. Type the format 999,999,990 in the Format field.
6. Choose the Add button to add this new number format to the list. This newly added number format appears in the Number Format list.
7. Choose the OK button to apply your selection and close the Number Format dialog. The new number format should be reflected in the chart.

Choose File —> Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File —> Exit to end your current Graphics session.

Lesson 4: Create a Drill-down Relationship

So far you've created a pie chart that shows some very high–level general information for all of Summit's warehouses, and a column chart that shows some detailed information for one specific warehouse. The charts contain one common item of data: the warehouse identification number. Now you'll dynamically link the two charts together using this common data item.

Remember that the column chart uses a query that refers to a specific warehouse identification number in the WHERE clause. However, rather than creating five separate charts, one for each warehouse, you'll create a parameter, a global variable, to represent the value of WAREHOUSE_ID in the column query WHERE clause.
Further, you can specify an event that sets the value of the parameter and then executes the column query.

This chain of events between two charts is called a “drill-down” relationship. For any given drill-down relationship, there are four basic components and related actions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>master chart element</td>
<td>determines the parameter value</td>
</tr>
<tr>
<td>parameter</td>
<td>referenced in query</td>
</tr>
<tr>
<td>query</td>
<td>creates the detail chart</td>
</tr>
<tr>
<td>detail chart</td>
<td>shows the data from the query</td>
</tr>
</tbody>
</table>

A user can “drill-down” on the master chart at runtime by selecting, or clicking on the specified master chart element. When Graphics detects a mouse event on the master chart element, it performs the following actions:

- assigns a value to a parameter based on the chart element selected
- executes the query assigned to the detail chart
- updates the detail chart based on the new data

While this may seem very complicated, it can be set up in the Designer in one property page, and without any PL/SQL programming.

### Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose **File**—>**Close** to close the default display that is presented.
3. Choose **File**—>**Open** or the Open button to open the display you saved at the end of the previous lesson.
4. Choose **File**—>**Connect** to connect to the database.

### Select the Active Chart Element

The active chart element is the one that will drive the drill-down relationship. This is an element of the “master” chart. In this case, the user will select a pie slice to activate a change in the column (detail) chart.

Each pie slice represents a warehouse ID number, the common data item. So use the pie slice element to set your parameter value.

5. Select the pie chart.
Further, you can specify an event that sets the value of the parameter and then executes the column query. This chain of events between two charts is called a "drill-down" relationship. For any given drill-down relationship, there are four basic components and related actions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>master chart element</td>
<td>determines the parameter value</td>
</tr>
<tr>
<td>parameter referenced in query</td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>creates the detail chart</td>
</tr>
<tr>
<td>detail chart</td>
<td>shows the data from the query</td>
</tr>
</tbody>
</table>

A user can "drill-down" on the master chart at runtime by selecting, or clicking on the specified master chart element. When Graphics detects a mouse event on the master chart element, it performs the following actions:

- assigns a value to a parameter based on the chart element selected
- executes the query assigned to the detail chart
- updates the detail chart based on the new data

While this may seem very complicated, it can be set up in the Designer in one property page, and without any PL/SQL programming.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

The active chart element is the one that will drive the drill-down relationship. This is an element of the "master" chart. In this case, the user will select a pie slice to activate a change in the column (detail) chart.

Each pie slice represents a warehouse ID number, the common data item. So use the pie slice element to set your parameter value.

5. Select the pie chart.
6. Select one of the pie slices in the chart.
   All of the pie slices should now appear in select mode.

   **Note:** Do not double-click on the chart. The pie slices are considered individual elements of the chart itself. To select a chart element, you must first select the parent object (the chart), then separately select the child object (the element).

Create the Parameter

You create parameters using the Parameters dialog. You can access this dialog through the Object property sheet Drill-down page, or through the Object Navigator. In addition to using parameters for drill-down relationships, you can use parameters as global variables in PL/SQL program units.

1. Choose **Properties** from the pop-up menu to show the Object property sheet for the pie slices. Alternatively, you can choose **Tools**→**Properties**.

   You will use the Object property sheet to define the parameter and drill-down relationship.

2. Click on the Drill-down tab of the Object property sheet to show the Drill-down property page.

   The Drill-down property page is designed to allow you to create dynamic actions without having to do any PL/SQL programming.

3. Choose the New button to the right of the Set Parameter field to create a new parameter.
6. Select one of the pie slices in the chart. All of the pie slices should now appear in select mode.

Note: Do not double-click on the chart. The pie slices are considered individual elements of the chart itself. To select a chart element, you must first select the parent object (the chart), then separately select the child object (the element).

You create parameters using the Parameters dialog. You can access this dialog through the Object property sheet Drill-down page, or through the Object Navigator. In addition to using parameters for drill-down relationships, you can use parameters as global variables in PL/SQL program units.

1. Choose Properties from the pop-up menu to show the Object property sheet for the pie slices. Alternatively, you can choose Tools —> Properties.

You will use the Object property sheet to define the parameter and drill-down relationship.

2. Click on the Drill-down tab of the Object property sheet to show the Drill-down property page. The Drill-down property page is designed to allow you to create dynamic actions without having to do any PL/SQL programming.

3. Choose the New button to the right of the Set Parameter field to create a new parameter.
The Parameters dialog appears. The parameter is given a default name of \textit{PARAMn}.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PARAM0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>PARAM0</td>
</tr>
<tr>
<td>Type:</td>
<td>Char</td>
</tr>
<tr>
<td>Initial Value:</td>
<td></td>
</tr>
</tbody>
</table>

4. Edit the Name field to rename the parameter \textit{whse}.

5. Select the NUMBER datatype from the Type drop-down list.
   The value of this field must match the datatype declared for the common data item. In this case, the common data item corresponds to the column WAREHOUSE_ID, which is declared as the datatype NUMBER.

6. Type 201 in the Initial Value field.
   This initial value will be over-ridden when a user selects a pie slice at runtime. You'll need it temporarily to execute the query.

7. Choose the OK button to accept the parameter settings.
   Graphics returns you to the Object property sheet.

**Specify the Drill-Down Action**

Now that you've defined a parameter, you can use the other fields in the Drill-down property page to specify an action that Graphics will perform based on the drill-down event.

1. Select WAREHOUSE_ID from the To Value Of drop-down list.
   This ties the value of the parameter to a value of the WAREHOUSE_ID column as returned by the pie query. Consider each slice of the pie to be one row of data returned by your query. Each of these rows has two columns: \textit{SUM(AMOUNT\_IN\_STOCK)}, and WAREHOUSE_ID.
   When a user selects a pie slice at runtime, the parameter value will be set to the value of WAREHOUSE_ID for that row(slice).
   For instance, selecting the pie slice labeled 401 will set the parameter value of \textit{whse} to 401.

2. Select the column query from the Execute Query drop-down list.
The Parameters dialog appears. The parameter is given a default name of **PARAMn**.

4. Edit the Name field to rename the parameter **whse**.

5. Select the NUMBER datatype from the Type drop-down list. The value of this field must match the datatype declared for the common data item. In this case, the common data item corresponds to the column **WAREHOUSE_ID**, which is declared as the datatype NUMBER.

6. Type **201** in the Initial Value field. This initial value will be over-ridden when a user selects a pie slice at runtime. You'll need it temporarily to execute the query.

7. Choose the OK button to accept the parameter settings.

Graphics returns you to the Object property sheet.

Now that you've defined a parameter, you can use the other fields in the Drill-down property page to specify an action that Graphics will perform based on the drill-down event.

1. Select **WAREHOUSE_ID** from the To Value Of drop-down list. This ties the value of the parameter to a value of the **WAREHOUSE_ID** column as returned by the pie query. Consider each slice of the pie to be one row of data returned by your query. Each of these rows has two columns: SUM(AMOUNT_IN_STOCK), and **WAREHOUSE_ID**.

When a user selects a pie slice at runtime, the parameter value will be set to the value of **WAREHOUSE_ID** for that row(slice). For instance, selecting the pie slice labeled **401** will set the parameter value of **whse** to **401**.

2. Select the column query from the Execute Query drop-down list.
This is the query you used to create the column chart. For each parameter value set by a user action, this query will be executed and the column chart will be updated to reflect the new data returned by the query.

**Edit the Query**

Remember that when you originally created the column query, you used a specific warehouse ID in the WHERE clause. You want to replace this static value with the new dynamic parameter.

1. Choose the Edit button to the right of the Execute Query field to show the Query property sheet.

   The Query property sheet containing the column query appears.

2. Edit the last line of the query as follows:

   ```sql
   WHERE warehouse_id = :whse
   ```

   When you reference a parameter like the one created above, you must precede the parameter name with a colon(:).

   Rather than specifying a set value for your WHERE clause, the parameter allows you determine the value from outside the query itself.

   This modified query will be executed when a pie slice is selected and a corresponding parameter value is set.

3. Choose the OK button to accept the changes to the query.

   Graphics returns you to the Object property sheet.

4. Choose the OK button to confirm the drill-down relationship.

   Graphics closes the Object property sheet.

**Update the Chart**

Now that you’ve defined your drill-down relationship, you can test it quickly by updating the column chart to reflect the new initial parameter value in the WHERE clause.

1. Select the column query in the Object Navigator.

2. Choose **Execute** from the pop-up menu to execute the new query and update your chart with the `whse` parameter initial value of 201.

   Notice that the chart changes to reflect the new warehouse identification number in the WHERE clause of the query.

   You will test the interactive features of the drill-down relationship in Lesson 11 on page 2 – 39.

**Save the Display**

Choose **File**→**Save** or the Save button to save your changes to the display.
This is the query you used to create the column chart. For each parameter value set by a user action, this query will be executed and the column chart will be updated to reflect the new data returned by the query.

Remember that when you originally created the column query, you used a specific warehouse ID in the WHERE clause. You want to replace this static value with the new dynamic parameter.

1. Choose the Edit button to the right of the Execute Query field to show the Query property sheet.

The Query property sheet containing the column query appears.

2. Edit the last line of the query as follows:

```sql
WHERE warehouse_id = :whse
```

When you reference a parameter like the one created above, you must precede the parameter name with a colon (:).

Rather than specifying a set value for your WHERE clause, the parameter allows you determine the value from outside the query itself.

This modified query will be executed when a pie slice is selected and a corresponding parameter value is set.

3. Choose the OK button to accept the changes to the query.

Graphics returns you to the Object property sheet.

4. Choose the OK button to confirm the drill-down relationship.

Graphics closes the Object property sheet.

Now that you've defined your drill-down relationship, you can test it quickly by updating the column chart to reflect the new initial parameter value in the WHERE clause.

1. Select the column query in the Object Navigator.

2. Choose **Execute** from the pop-up menu to execute the new query and update your chart with the **whse** parameter initial value of **201**.

Notice that the chart changes to reflect the new warehouse identification number in the WHERE clause of the query.

You will test the interactive features of the drill-down relationship in Lesson 11 on page 2–39.

Choose **File** —> **Save** or the Save button to save your changes to the display.
Lesson 5: Create a Format Trigger

Format triggers are a way to assign a visual signal to a set of circumstances. The visual signal can appear in any element within the chart.

For example, the column chart compares the amount of a given product currently in stock to that product's reorder point. You can see if the amount in stock column falls below the reorder point column, but perhaps you’d like a more noticeable indicator that a product is below its reorder point.

Let’s create a format trigger that changes the product ID number along the X-axis to red under the condition that the amount in stock is less than the reorder point.

Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File→Close to close the default display that is presented.
3. Choose File→Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File→Connect to connect to the database.

Set Up the Format Trigger

In this section, you’ll set up the format trigger by selecting the chart element you want your data to format and use its property sheet to create a new format trigger program unit.

1. Select the column chart.
2. Select a product ID tick label along the X-axis.
   All product ID numbers should appear in select mode.
3. Choose Properties from the pop-up menu to show the Object property sheet for the tick labels.
   Alternatively, you can choose Tools→Properties to show the Object property sheet.
Format triggers are a way to assign a visual signal to a set of circumstances. The visual signal can appear in any element within the chart.

For example, the column chart compares the amount of a given product currently in stock to that product's reorder point. You can see if the amount in stock column falls below the reorder point column, but perhaps you'd like a more noticeable indicator that a product is below its reorder point.

Let's create a format trigger that changes the product ID number along the X–axis to red under the condition that the amount in stock is less than the reorder point.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

In this section, you'll set up the format trigger by selecting the chart element you want your data to format and use its property sheet to create a new format trigger program unit.

1. Select the column chart.
2. Select a product ID tick label along the X–axis. All product ID numbers should appear in select mode.
3. Choose Properties from the pop-up menu to show the Object property sheet for the tick labels. Alternatively, you can choose Tools —> Properties to show the Object property sheet.
4. Select the New button to the right of the Format Trigger field to show the Program Unit editor.

You may have to select the Object tab if you continued directly from the previous lesson.

**Enter the Format Trigger Body**

Notice that a template for this PL/SQL procedure appears in the Program Unit editor. You merely need to fill in the blanks. You may need to move the Object Property sheet. The new program unit is named `OGFORMATTRIG0` by default.
4. Select the New button to the right of the Format Trigger field to show the Program Unit editor. You may have to select the Object tab if you continued directly from the previous lesson. Notice that a template for this PL/SQL procedure appears in the Program Unit editor. You merely need to fill in the blanks. The new program unit is named **OGFORMATTRIG0** by default.
Define Variables

You can use variables to store and represent values in your program units. Variables are always defined or declared before the program unit body (indicated by the BEGIN line). Each variable must have a unique name within that program unit and a datatype that corresponds to the value it will represent.

If you want to use the same variables in more than one program unit, create a parameter. Parameters do not need to be defined at the beginning of the program unit.

Type the following two lines immediately between the opening line that defines the procedure (ending in IS) and the BEGIN line:

```plsql
stock NUMBER;
reord NUMBER;
```

You will use these two variables later to compare the amount in stock and reorder points for a given product ID number. These variables will store values from the AMOUNT_IN_STOCK and REORDER_POINT columns, both of which are datatype NUMBER. If you are not sure of a table column’s datatype, check with your database administrator.

Your format trigger procedure should now look like this:

```plsql
PROCEDURE OGFORMATTRIG0(elem IN og_object,
query IN og_query) IS
stock NUMBER;
reord NUMBER;
BEGIN
END;
```

Assign Values to the Variables

Type these two lines immediately after the BEGIN line to assign values to the variables:

```plsql
stock:=OG_GET_NUMCELL(query,OG_NEWDATA,
'amount_in_stock');
reord:=OG_GET_NUMCELL(query,OG_NEWDATA,'reorder_point');
```

The OG_GET_NUMCELL function is an Graphics built-in function that returns a single item of data to be evaluated. For the specified column, this function reads one row at a time, until all rows in the data table returned by the query have been evaluated.

The OG_NEWDATA argument specifies that the OG_GET_NUMCELL function use the most recent data retrieved by the query.

The lines above assign a variable to each incremental value that OG_GET_NUMCELL returns. This variable will be evaluated by a condition. Depending on the condition, an action may or may not take place.
You can use variables to store and represent values in your program units. Variables are always defined or declared before the program unit body (indicated by the BEGIN line). Each variable must have a unique name within that program unit and a datatype that corresponds to the value it will represent.

If you want to use the same variables in more than one program unit, create a parameter. Parameters do not need to be defined at the beginning of the program unit.

Type the following two lines immediately between the opening line that defines the procedure (ending in IS) and the BEGIN line:

```
stock NUMBER;
reord NUMBER;
```

You will use these two variables later to compare the amount in stock and reorder points for a given product ID number. These variables will store values from the AMOUNT_IN_STOCK and REORDER_POINT columns, both of which are datatype NUMBER. If you are not sure of a table column's datatype, check with your database administrator.

Your format trigger procedure should now look like this:

```
PROCEDURE OGFORMATTRIG0(elem IN og_object,
query IN og_query) IS
stock NUMBER;
reord NUMBER;
BEGIN
END;
```

Type these two lines immediately after the BEGIN line to assign values to the variables:

```
stock:=OG_GET_NUMCELL(query,OG_NEWDATA,'amount_in_stock');
reord:=OG_GET_NUMCELL(query,OG_NEWDATA,'reorder_point');
```

The OG_GET_NUMCELL function is an Graphics built-in function that returns a single item of data to be evaluated. For the specified column, this function reads one row at a time, until all rows in the data table returned by the query have been evaluated.

The OG_NEWDATA argument specifies that the OG_GET_NUMCELL function use the most recent data retrieved by the query.

The lines above assign a variable to each incremental value that OG_GET_NUMCELL returns. This variable will be evaluated by a condition. Depending on the condition, an action may or may not take place.
Set the Condition for the Trigger

Type these lines immediately before the END; line and immediately after the lines you typed in the previous step to set the condition and the action for the trigger:

```
IF stock <= reord THEN
  OG_SET_GCOLOR(elem, 'red');
END IF;
```

This PL/SQL statement says if the value assigned to the stock variable is less than or equal to the value assigned to the reord variable, set the text color of the Product ID tick label (elem) to red. If this condition is not met, no action is taken.

Your entire procedure should now look like this:

```
PROCEDURE OGFORMATTRIG0(elem IN og_object,
                          query IN og_query) IS
  stock  NUMBER;
  reord  NUMBER;
BEGIN
  stock:=OG_GET_NUMCELL(query, OG_NEWDATA,
                         'amount_in_stock');
  reord:=OG_GET_NUMCELL(query,OG_NEWDATA,
                         'reorder_point');
  IF stock <= reord THEN
    OG_SET_GCOLOR(elem, 'red');
  END IF;
END;
```

Complete the Format Trigger

Once you’ve entered the body of the format trigger program unit, perform the following steps to apply it to the display.

1. Choose the Compile button in the Program Unit editor to compile the procedure.

   Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines.

   If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.

   The program unit must be compiled without errors before you can proceed.

2. Choose the Close button to close the Program Unit editor.

3. Choose OK to close the Object property sheet.
Set the Condition for the Trigger

Complete the Format Trigger

Type these lines immediately before the END; line and immediately after the lines you typed in the previous step to set the condition and the action for the trigger:

```
IF stock <= reord THEN
  OG_SET_GCOLOR(elem, 'red');
END IF;
```

This PL/SQL statement says if the value assigned to the stock variable is less than or equal to the value assigned to the reord variable, set the text color of the Product ID tick label (elem) to red. If this condition is not met, no action is taken.

Your entire procedure should now look like this:

```
PROCEDURE OGFORMATTRIG0(elem IN og_object, query IN og_query) IS
  stock  NUMBER;
  reord  NUMBER;
BEGIN
  stock:=OG_GET_NUMCELL(query, OG_NEWDATA, 'amount_in_stock');
  reord:=OG_GET_NUMCELL(query,OG_NEWDATA,'reorder_point');
  IF stock <= reord THEN
    OG_SET_GCOLOR(elem, 'red');
  END IF;
END;
```

Once you've entered the body of the format trigger program unit, perform the following steps to apply it to the display.

1. Choose the Compile button in the Program Unit editor to compile the procedure.
   Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines.
   If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.
   The program unit must be compiled without errors before you can proceed.

2. Choose the Close button to close the Program Unit editor.

3. Choose OK to close the Object property sheet.
Now that you’ve defined your format trigger, you can update the column chart to see the results. Select the column chart and choose **Update** from the pop-up menu, or **Chart —> Update Chart**, to activate the format trigger and view the results. Notice that some of the Product ID tick labels now appear in red.

Your chart should look something like this:

![Inventory Levels by Product](chart.png)

**Update the Chart**

**Save the Display**

Choose **File—>Save** or the Save button to save your changes to the display.

**Continue or Exit**

Continue with the next lesson or choose **File—>Exit** to end your current Graphics session.

**Lesson 6: Add a Text Object**

Since the drill-down procedure you created requires some user input to make it work, let’s add some instructions so users know how to activate it at runtime.
Now that you've defined your format trigger, you can update the column chart to see the results. Select the column chart and choose Update from the pop-up menu, or Chart —> Update Chart, to activate the format trigger and view the results. Notice that some of the Product ID tick labels now appear in red. Your chart should look something like this:

Choose File —> Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File —> Exit to end your current Graphics session.

Lesson 6: Add a Text Object

Since the drill-down procedure you created requires some user input to make it work, let's add some instructions so users know how to activate it at runtime.
Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

Create the Text Object

You can create graphic objects anywhere on the layout using the tools in the Layout editor tool palette.

1. Select the Text tool in the Layout editor’s Tool palette.
   The cursor changes from a pointer to a crosshair.
2. Place the cursor underneath the pie chart and click to create a text object.
   A text entry field appears on the display and the cursor changes to a text entry indicator.
3. Type the following text:
   
   Select a pie slice to update the chart below
   Enter a hard return to insert the line break in the text.
   To get the text object out of text entry mode, click somewhere else on the layout. The text object is still selected. You can move the text object by selecting it and dragging it with the mouse.
4. Click on the Line Color button to show the Line Color palette.
5. Choose No Line from the Line Color palette menu.
   The palette closes. Once you specify the No Line option, it is applied to all other graphic objects you create with the tools in the Layout editor Tool palette in that display.
   To apply a border line to an object that does not have one, select the object, click on the Line Color button to show the Line Color palette, and choose any color for the line.
6. Select the text object and move it so it is centered beneath the pie chart.

Save the Display

Choose File—>Save or the Save button to save your changes to the display.
If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose **File** —> **Close** to close the default display that is presented.
3. Choose **File** —> **Open** or the Open button to open the display you saved at the end of the previous lesson.
4. Choose **File** —> **Connect** to connect to the database.

You can create graphic objects anywhere on the layout using the tools in the Layout editor tool palette.

1. Select the Text tool in the Layout editor's Tool palette. The cursor changes from a pointer to a crosshair.
2. Place the cursor underneath the pie chart and click to create a text object. A text entry field appears on the display and the cursor changes to a text entry indicator.
3. Type the following text:
   
   Select a pie slice to update the chart below
   Enter a hard return to insert the line break in the text.

   To get the text object out of text entry mode, click somewhere else on the layout. The text object is still selected. You can move the text object by selecting it and dragging it with the mouse.
4. Click on the Line Color button to show the Line Color palette.
5. Choose **No Line** from the Line Color palette menu. The palette closes. Once you specify the No Line option, it is applied to all other graphic objects you create with the tools in the Layout editor Tool palette in that display.

To apply a border line to an object that does not have one, select the object, click on the Line Color button to show the Line Color palette, and choose any color for the line.

6. Select the text object and move it so it is centered beneath the pie chart.
7. Choose **File** —> **Save** or the Save button to save your changes to the display.
Lesson 7: Create a Break Chart

The column chart you created shows product inventory information for a specific warehouse. Now you’ll create a chart that does the opposite: shows inventory information of all warehouses. In Lesson 8 on page 2 – 26 you’ll use a query filter to refine this chart to just a few products. For each product, you will be charting two categories of data: inventory level, and warehouse ID.

Break charts allow you to plot two categories of data along a chart axis. This enables you to combine two charts into one and see relative information side by side.

Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:
1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

Create the Chart

This chart will be rather large. Use the scroll bars in the Layout editor to scroll to a clear area of the display.
1. Select the Chart tool in the Layout editor tool palette.
2. Click the mouse in the display area underneath your column chart.
   It is not necessary to drag out a region for the chart. If you simply click the mouse in the display area where you wish the chart to be placed, Graphics will create a chart at that location using its own size defaults. You can then resize the chart as necessary.
3. Select the New Query radio button in the Chart Genie.
4. Name the new query break.
5. Choose the OK button to show the New Query dialog.
Lesson 7: Create a Break Chart

The column chart you created shows product inventory information for a specific warehouse. Now you'll create a chart that does the opposite: shows inventory information of all warehouses. In Lesson 8 on page 2–26 you'll use a query filter to refine this chart to just a few products. For each product, you will be charting two categories of data: inventory level, and warehouse ID.

Break charts allow you to plot two categories of data along a chart axis. This enables you to combine two charts into one and see relative information side by side.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

This chart will be rather large. Use the scroll bars in the Layout editor to scroll to a clear area of the display.

1. Select the Chart tool in the Layout editor tool palette.
2. Click the mouse in the display area underneath your column chart. It is not necessary to drag out a region for the chart. If you simply click the mouse in the display area where you wish the chart to be placed, Graphics will create a chart at that location using its own size defaults. You can then resize the chart as necessary.
3. Select the New Query radio button in the Chart Genie.
4. Name the new query break.
5. Choose the OK button to show the New Query dialog.
6. Type the following query in the SQL Statement field:

```sql
SELECT product_id "Product ID",
       SUM(amount_in_stock) "Inventory",
       warehouse_id
FROM s_inventory
GROUP BY product_id, warehouse_id
```

Notice that the above query uses column name aliases. These aliases will appear as the axis labels for your chart. You now know two ways to assign custom axis labels: using aliases in your query, and through the Axis property sheet.

7. Choose the OK button to execute the query and close the New Query dialog.

The Chart property sheet appears.

Specify a Subcategory

In this chart, you want to chart the amount in stock for each product in each warehouse. That is, for each product, you want to see how much stock is stored in each of Summit’s five warehouses.

Our main data category will be the product ID numbers. These will appear as tick labels along the X-axis. The subcategory data will be the warehouse ID. These will appear as sets of different colored or patterned columns.

1. Name the chart `break_inv`.

2. Choose the Categories tab to show the Categories property page.

   You can use this property page to specify which data columns are charted along your category or independent data axis. For break charts, you can specify a main data category as well as a subcategory.

3. Make sure Product ID (your alias for the PRODUCT_ID column) is the only item in the Chart Categories list.

   You can remove other items from the list by double-clicking them, or by selecting them and then choosing the Delete button.

4. Select WAREHOUSE_ID from the Subcategory drop-down list.

   This drop-down list is located in the lower right corner of the property page, underneath the Category list.

5. Choose the Values tab to show the Values property page.

   You can use this property page to specify which data columns from your query are charted along your value or dependent data axis.

6. Make sure Inventory is the only item in the Chart Values list.
6. Type the following query in the SQL Statement field:

```
SELECT product_id "Product ID",
       SUM(amount_in_stock) "Inventory",
       warehouse_id
FROM s_inventory
GROUP BY product_id, warehouse_id
```

Notice that the above query uses column name aliases. These aliases will appear as the axis labels for your chart. You now know two ways to assign custom axis labels: using aliases in your query, and through the Axis property sheet.

7. Choose the OK button to execute the query and close the New Query dialog. The Chart property sheet appears.

In this chart, you want to chart the amount in stock for each product in each warehouse. That is, for each product, you want to see how much stock is stored in each of Summit's five warehouses.

Our main data category will be the product ID numbers. These will appear as tick labels along the X–axis. The subcategory data will be the warehouse ID. These will appear as sets of different colored or patterned columns.

1. Name the chart `break_inv`.
2. Choose the Categories tab to show the Categories property page. You can use this property page to specify which data columns are charted along your category or independent data axis. For break charts, you can specify a main data category as well as a subcategory.
3. Make sure Product ID (your alias for the PRODUCT_ID column) is the only item in the Chart Categories list. You can remove other items from the list by double-clicking them, or by selecting them and then choosing the Delete button.
4. Select WAREHOUSE_ID from the Subcategory drop-down list. This drop-down list is located in the lower right corner of the property page, underneath the Category list.
5. Choose the Values tab to show the Values property page. You can use this property page to specify which data columns from your query are charted along your value or dependent data axis.
6. Make sure Inventory is the only item in the Chart Values list.
You can remove other items from the list by double-clicking them, or by selecting them and then choosing the Delete button.

7. Choose the OK button to create the chart.

This chart contains a lot of data. You may need to resize the chart so you can see all the data clearly.

Notice that for each product ID plotted along the X-axis, there are up to five columns: one for each warehouse. The legend to the right of the chart indicates the warehouse ID/color mapping.

**Save the Display**

Choose **File**—>**Save** or the Save button to save your changes to the display.

**Continue or Exit**

Continue with the next lesson or choose **File**—>**Exit** to end your current Graphics session.

**Lesson 8: Create a Query Filter**

The format trigger you created for the “Inventory Levels by Product” chart gives you a visual cue as to which products are below their reorder point. Format triggers use a condition or set of conditions, and ties a specified action to the chart elements that meet the condition(s).

Query filters are similar. However, with a query filter the action tied to your specified condition(s) is pre-determined. If a row of data in the query meets the condition, that row is plotted on the chart. If the row of data does not meet the condition set, the row is not plotted on the chart. This is called a boolean condition.

In this sense, query filters perform the same action as a WHERE clause. The benefit of using a query filter over specifying a WHERE clause in the query is performance.

Let’s say you were creating a number of charts and each was based on the same query, but used slightly different WHERE clause criteria. You could create a specific query for each chart. Each of those specific queries would have to be executed against the server each time the chart is updated.

Alternatively, you could define one generic query, and for each chart, define a specific query filter to do the fine tuning. This way, only one query needs to be executed against the database and the filtering of data is performed on the client side.
You can remove other items from the list by double-clicking them, or by selecting them and then choosing the Delete button.

7. Choose the OK button to create the chart. This chart contains a lot of data. You may need to resize the chart so you can see all the data clearly.

Notice that for each product ID plotted along the X-axis, there are up to five columns: one for each warehouse. The legend to the right of the chart indicates the warehouse ID/color mapping.

Choose File —> Save or the Save button to save your changes to the display. Continue with the next lesson or choose File —> Exit to end your current Graphics session.

Lesson 8: Create a Query Filter

The format trigger you created for the "Inventory Levels by Product" chart gives you a visual cue as to which products are below their reorder point. Format triggers use a condition or set of conditions, and ties a specified action to the chart elements that meet the condition(s).

Query filters are similar. However, with a query filter the action tied to your specified condition(s) is pre-determined. If a row of data in the query meets the condition, that row is plotted on the chart. If the row of data does not meet the condition set, the row is not plotted on the chart. This is called a boolean condition.

In this sense, query filters perform the same action as a WHERE clause. The benefit of using a query filter over specifying a WHERE clause in the query is performance.

Let's say you were creating a number of charts and each was based on the same query, but used slightly different WHERE clause criteria. You could create a specific query for each chart. Each of those specific queries would have to be executed against the server each time the chart is updated.

Alternatively, you could define one generic query, and for each chart, define a specific query filter to do the fine tuning. This way, only one query needs to be executed against the database and the filtering of data is performed on the client side.
The break chart you created in the previous lesson is a little too large to use for this display. In this lesson you’ll use a query filter to narrow down the range of data plotted to chart only Summit’s fastest selling products.

**Restart Graphics**

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

**Set Up the Query Filter**

In this section, you’ll set up the query filter for the chart using the Chart property sheet.

1. Select the break chart.
2. Open the chart’s property sheet.
3. Choose the Data tab to show the Data property page.
4. Select the New button to the left of the Filter Function field to show the Program Unit editor.

**Enter the Query Filter Body**

Notice that a template for the filter function is provided in the editor. As with the format trigger you created in Lesson 5, you simply need to fill in the blanks.

The filter function is named `OGQUERYFILTER0` by default and contains the following initial content:

```plaintext
FUNCTION OGQUERYFILTER0(chartobj IN og_object,
                          query IN og_query)
RETURN BOOLEAN IS
```

Type the following line immediately before the BEGIN line:

```plaintext
prodid NUMBER;
```

This variable will store a value based on the PRODUCT_ID column, which is defined as the datatype NUMBER. Remember that in the query we used an alias for the PRODUCT_ID column. You must use that alias when referring to this column.
The break chart you created in the previous lesson is a little too large to use for this display. In this lesson you'll use a query filter to narrow down the range of data plotted to chart only Summit's fastest selling products.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

In this section, you'll set up the query filter for the chart using the Chart property sheet.

1. Select the break chart.
2. Open the chart's property sheet.
3. Choose the Data tab to show the Data property page.
4. Select the New button to the left of the Filter Function field to show the Program Unit editor.

Notice that a template for the filter function is provided in the editor. As with the format trigger you created in Lesson 5, you simply need to fill in the blanks.

The filter function is named OGQUERYFILTER0 by default and contains the following initial content:

```
FUNCTION OGQUERYFILTER0(chartobj IN og_object,
query IN og_query)
RETURN BOOLEAN IS
```

Type the following line immediately before the BEGIN line:

```
prodid  NUMBER;
```

This variable will store a value based on the PRODUCT_ID column, which is defined as the datatype NUMBER. Remember that in the query we used an alias for the PRODUCT_ID column. You must use that alias when referring to this column.
Assign a Value to the Variable

Type this line immediately after the BEGIN line:

```plaintext
prodid := OG_GET_NUMCELL(query, OG_NEWDATA, 'Product ID');
```

Set the Condition for the Filter

Type the following IF statement immediately after the variable assignment line above:

```plaintext
IF prodid IN ('20512', '32861', '41050') THEN
    RETURN TRUE;
ELSE
    RETURN FALSE;
END IF;
```

This condition states that if the value assigned to the prodid variable by the OG_GET_NUMCELL function is found in the list of three values, plot the data, otherwise, do not plot the data. So your chart will have only three sets of data charted in it.

Your program unit should now look like this:

```plaintext
FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query)
RETURN BOOLEAN IS

  prodid NUMBER;
BEGIN
  prodid := OG_GET_NUMCELL(query, OG_NEWDATA, 'Product ID');
  IF prodid IN ('20512', '32861', '41050') THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END;
```

This function will “filter” and plot three rows of data on the chart (one for each product ID you specified in the list). Rows that do not meet the set condition are not plotted on the chart.

Complete the Query Filter

Perform the following two steps to complete and apply the query filter to the break chart:

1. Choose the Compile button in the Program Unit editor to compile the procedure.

   Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines.

   If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.
Type this line immediately after the BEGIN line:
`prodid := OG_GET_NUMCELL(query, OG_NEWDATA, 'Product ID');`

Type the following IF statement immediately after the variable assignment line above:
`IF prodid IN ('20512', '32861', '41050') THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;`

This condition states that if the value assigned to the prodid variable by the OG_GET_NUMCELL function is found in the list of three values, plot the data, otherwise, do not plot the data. So your chart will have only three sets of data charted in it.

Your program unit should now look like this:
```java
FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query) RETURN BOOLEAN IS
  prodid NUMBER;
BEGIN
  prodid := OG_GET_NUMCELL(query, OG_NEWDATA, 'Product ID');
  IF prodid IN ('20512', '32861', '41050') THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END;
```

This function will "filter" and plot three rows of data on the chart (one for each product ID you specified in the list). Rows that do not meet the set condition are not plotted on the chart.

Perform the following two steps to complete and apply the query filter to the break chart:

1. Choose the Compile button in the Program Unit editor to compile the procedure.

Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines.

If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.
2. Choose the Close button to close the Program Unit editor.

**Update the Chart**

Now that you’ve completed the query filter, you can update the break chart to activate the query filter and view the results. First though, let’s assign an appropriate title to the new chart.

1. In the Chart property sheet, choose the Chart tab to show the Chart property page.

2. Type **Popular Product Inventory Distribution** in the chart title field.

3. Choose the OK button to update the chart with the query filter.

   Your break chart now contains only three sets of data, one for each product identification number you entered in the filter.

**Customize the Chart**

Assign the **999,999,990** number format to the tick labels on the Inventory (Y) axis.

**Move and Resize the Chart**

Since the break chart now contains much less data, move and resize it so that it fits next to your pie chart at the top right corner of the display.

1. Select the break chart.

2. Select the lower right corner control point and drag it so your chart is approximately three inches across by two and a half inches high.

   If necessary, select **Update** from the pop-up menu to force a redraw of the chart. Sometimes drastically resizing an object can distort it.

3. Select the outer frame of the chart’s legend.

4. Click inside the legend frame and drag it so it is centered vertically with the chart.

   Select **Update** from the pop-up menu to force a redraw of the legend.

5. Select the chart and move it to the upper right corner of your display so it fits next to the pie chart.

   Your display should look something like this:
Now that you’ve completed the query filter, you can update the break chart to activate the query filter and view the results. First though, let’s assign an appropriate title to the new chart.

1. In the Chart property sheet, choose the Chart tab to show the Chart property page.

2. Type **Popular Product Inventory Distribution** in the chart title field.

3. Choose the OK button to update the chart with the query filter.

Your break chart now contains only three sets of data, one for each product identification number you entered in the filter.

Assign the number format **999,999,990** to the tick labels on the Inventory (Y) axis.

Since the break chart now contains much less data, move and resize it so that it fits next to your pie chart at the top right corner of the display.

1. Select the break chart.

2. Select the lower right corner control point and drag it so your chart is approximately three inches across by two and a half inches high. If necessary, select **Update** from the pop-up menu to force a redraw of the chart. Sometimes drastically resizing an object can distort it.

3. Select the outer frame of the chart’s legend.

4. Click inside the legend frame and drag it so it is centered vertically with the chart. Select **Update** from the pop-up menu to force a redraw of the legend.

5. Select the chart and move it to the upper right corner of your display so it fits next to the pie chart. Your display should look something like this:
Save the Display

Choose File—>Save or the Save button to save your changes to the display.

Continue or Exit

Continue with the next lesson or choose File—>Exit to end your current Graphics session.
Save the Display

Continue or Exit

Choose File —> Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File —> Exit to end your current Graphics session.
Lesson 9: Create a Second Layer

Your display now contains some key pieces of information about inventory levels of the products sold by Summit Sporting Goods. However, what you’d really like is to be able to use that original break chart you created in Lesson 7. It provided some good overview information but just didn’t fit on the display.

Rather than create a separate display, you can create multiple layers or surfaces within one display to hide and show different display elements. In this case, what you want is one layer for your three summary charts, and a second layer for your large break chart.

Users can only interact with one layer at a time, so you need to provide them with a mechanism for navigating back and forth from one layer to another. In the next lesson, you’ll create a button procedure with PL/SQL to do this.

In this lesson, you’ll create the second layer and create the break chart there. Remember that you still have the query to create this chart.

Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File→Close to close the default display that is presented.
3. Choose File→Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File→Connect to connect to the database.

Create a New Layer

In this section, you’ll learn how to create, rename, activate, hide, and show layers. Notice in the Object Navigator, under the display node, that a default layer named layer0 is created for us by default.

1. Choose Tools→Layers to show the Layer Settings dialog.
Your display now contains some key pieces of information about inventory levels of the products sold by Summit Sporting Goods. However, what you'd really like is to be able to use that original break chart you created in Lesson 7. It provided some good overview information but just didn't fit on the display.

Rather than create a separate display, you can create multiple layers or surfaces within one display to hide and show different display elements. In this case, what you want is one layer for your three summary charts, and a second layer for your large break chart.

Users can only interact with one layer at a time, so you need to provide them with a mechanism for navigating back and forth from one layer to another. In the next lesson, you'll create a button procedure with PL/SQL to do this.

In this lesson, you'll create the second layer and create the break chart there. Remember that you still have the query to create this chart. If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

In this section, you'll learn how to create, rename, activate, hide, and show layers. Notice in the Object Navigator, under the display node, that a default layer named layer0 is created for us by default.

1. Choose Tools —> Layers to show the Layer Settings dialog.
You will use this dialog to perform all layer activities in the Designer.

Notice that layer0 appears in the Existing Layers list box. In addition, layer0 appears in the Active Layer field below the list box. Only one layer can be active at a time. You can only interact with objects on the active layer.

The active layer must be shown. The “+” sign to the left of the layer name indicates that the layer is showing. You can hide or show different layers, but you cannot hide the active layer. If a layer is hidden, a “−” sign appears to the left of its name in the Existing Layers list box.

If you want to hide the active layer, you must activate another layer first. When you activate another layer, that layer is automatically shown.

2. Choose the New button to create a new layer.

The new layer is named layer1 by default. Notice that the new layer you’ve created becomes the active layer. In the layout though, nothing seems to have happened. You can still see your charts.

Try to select one of them. You can’t. You can only interact with objects on the active layer, and the active layer is now the empty layer1.

3. Select layer0 in the Existing Layers list box and choose the Hide/Show button to hide it.
You will use this dialog to perform all layer activities in the Designer.

Notice that layer0 appears in the Existing Layers list box. In addition, layer0 appears in the Active Layer field below the list box.

Only one layer can be active at a time. You can only interact with objects on the active layer.

The active layer must be shown. The "+" sign to the left of the layer name indicates that the layer is showing. You can hide or show different layers, but you cannot hide the active layer. If a layer is hidden, a "-" sign appears to the left of its name in the Existing Layers list box.

If you want to hide the active layer, you must activate another layer first. When you activate another layer, that layer is automatically shown.

2. Choose the New button to create a new layer. The new layer is named layer1 by default. Notice that the new layer you've created becomes the active layer. In the layout though, nothing seems to have happened. You can still see your charts. Try to select one of them. You can't. You can only interact with objects on the active layer, and the active layer is now the empty layer1.

3. Select layer0 in the Existing Layers list box and choose the Hide/Show button to hide it.
Layer0 “disappears” and you are left with what appears to be a blank display. However, what you are really seeing is layer1.

Layer0 is still there, but it is now hidden.

4. Choose the OK button to close the Layer Settings dialog.

Create the Break Chart

Because you created a query filter function in Lesson 8, the query you used to create the original query still exists in its original form so you can use it again to recreate the original break chart.

Again, this is one of the benefits of using query filters. The same break query can be used with a query filter to create the “Popular Products” chart as well as the chart you will create in this lesson.

1. Select the Chart tool in the Layout editor Tool palette.
2. Click the mouse anywhere in the layout.
3. Select the break query from the Existing Query drop-down list in the Chart Genie.
   This is the query you used to create the original break chart.
4. Choose the OK button to close the Chart Genie.
   The Chart property sheet appears.
5. Type the name break_full in the Name field.
6. Type the title Inventory Distribution in the Title field.
7. Choose the Categories tab to show the Categories property page.
8. Make sure Product ID is the only item in the Chart Categories list.
9. Select WAREHOUSE_ID from the Subcategory drop-down list.
10. Choose the Values tab to show the Values property page.
11. Make sure Inventory is the only item in the Chart Values list.
12. Choose the OK button to create the chart.
13. Resize and move the chart as necessary.

Save the Display
Choose File—>Save or the Save button to save your changes to the display.

Continue or Exit
Continue with the next lesson or choose File—>Exit to end your current Graphics session.
Layer0 "disappears" and you are left with what appears to be a blank display. However, what you are really seeing is layer1. Layer0 is still there, but it is now hidden.

4. Choose the OK button to close the Layer Settings dialog.

Because you created a query filter function in Lesson 8, the query you used to create the original query still exists in its original form so you can use it again to recreate the original break chart. Again, this is one of the benefits of using query filters. The same break query can be used with a query filter to create the "Popular Products" chart as well as the chart you will create in this lesson.

1. Select the Chart tool in the Layout editor Tool palette.
2. Click the mouse anywhere in the layout.
3. Select the break query from the Existing Query drop-down list in the Chart Genie. This is the query you used to create the original break chart.
4. Choose the OK button to close the Chart Genie. The Chart property sheet appears.
5. Type the name break_full in the Name field.
6. Type the title Inventory Distribution in the Title field.
7. Choose the Categories tab to show the Categories property page.
8. Make sure Product ID is the only item in the Chart Categories list.
9. Select WAREHOUSE_ID from the Subcategory drop-down list.
10. Choose the Values tab to show the Values property page.
11. Make sure Inventory is the only item in the Chart Values list.
12. Choose the OK button to create the chart.
13. Resize and move the chart as necessary.

Choose File —> Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File —> Exit to end your current Graphics session.
Lesson 10: Create a Button Procedure

Now you have a display with two layers. What you need is a method for users to navigate from one layer to the other to view the different charts at runtime. In this lesson, you’ll create a button procedure using a graphic object and some PL/SQL code. Button procedures are similar to the drill-down relationship you created in Lesson 4 on page 2 – 13.

However, instead of the user selecting a chart element such as a pie slice, the user will select a “button” that you will create in the layout. When Graphics detects a mouse event on the button object, it will activate the PL/SQL program unit you will create in this lesson.

Restart Graphics

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File—>Connect to connect to the database.

Return to the First Layer

You will create a button on the first layer.

1. Choose Tools—>Layers to show the Layer Setting dialog.
2. Select Layer0 from the Existing Layers list.
3. Choose the Activate button to make Layer0 the active layer.
4. Select Layer1 in the Existing Layers list.
5. Choose the Hide/Show button to hide Layer1.
6. Choose the OK button to apply your changes and close the Layer Settings dialog.

Create a Button Object

You will create a button that will activate a procedure to switch layers at the user’s request. First you’ll create another text object, then you’ll draw the button around it.

1. Select the Text tool in the Layout editor Tool palette.
2. Click on the layout, between the top two charts, to create the text object.
3. Type Next Page.
4. Click outside the text object to get out of text entry mode.
Now you have a display with two layers. What you need is a method for users to navigate from one layer to the other to view the different charts at runtime. In this lesson, you'll create a button procedure using a graphic object and some PL/SQL code. Button procedures are similar to the drill-down relationship you created in Lesson 4 on page 2–13. However, instead of the user selecting a chart element such as a pie slice, the user will select a "button" that you will create in the layout. When Graphics detects a mouse event on the button object, it will activate the PL/SQL program unit you will create in this lesson.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.
4. Choose File —> Connect to connect to the database.

You will create a button on the first layer.

1. Choose Tools —> Layers to show the Layer Setting dialog.
2. Select Layer0 from the Existing Layers list.
3. Choose the Activate button to make Layer0 the active layer.
4. Select Layer1 in the Existing Layers list.
5. Choose the Hide/Show button to hide Layer1.
6. Choose the OK button to apply your changes and close the Layer Settings dialog.

You will create a button that will activate a procedure to switch layers at the user's request. First you'll create another text object, then you'll draw the button around it.

1. Select the Text tool in the Layout editor Tool palette.
2. Click on the layout, between the top two charts, to create the text object.
3. Type Next Page.
4. Click outside the text object to get out of text entry mode.
The text object has no border line because you turned off border lines for all objects in Lesson 6.

5. Select the Rectangle tool.

The cursor changes from a pointer to a crosshair.

6. Click and drag in the layout to create a button shaped object around the text object.

Release the mouse when the object is the desired size.

7. Click on the Line Color button to show the Line Color palette.

8. Select the box in the upper left corner of the palette to set the line color to black.

Notice that as you move the pointer around the palette, the Text/Line/Fill Display area in the Tool palette changes to reflect the different colors in the color palette.

9. With the rectangle still selected, choose **Arrange—>Send to Back**.

By creating the rectangle after the text, you’ve placed the rectangle above the text in the layout. Your rectangle is currently transparent, so this has little effect on the appearance of the button. But if you were to select a color for the rectangle with it still on top, it would obscure the text. For this reason, you move the rectangle behind the text.

**Arrange the Button Object Elements**

You’ve created two graphic objects that make up your button: the rounded rectangle, and the text. In this section you’ll group and align the two objects.

1. Select the rounded rectangle object.

2. Use Shift-Click to select the text object.

Both objects should be selected.

3. Choose **Arrange—>Align Objects** to show the Alignment Settings dialog.

4. Select the following alignment options:
   - align to each other
   - align centers horizontally
   - align centers vertically

5. Choose the OK button to perform the alignment.

6. With both objects still selected, choose **Arrange—>Group** to group the two objects.
5. Select the Rectangle tool.

6. Click and drag in the layout to create a button shaped object around the text object. Release the mouse when the object is the desired size.

7. Click on the Line Color button to show the Line Color palette.

8. Select the box in the upper left corner of the palette to set the line color to black.

Notice that as you move the pointer around the palette, the Text/Line/Fill Display area in the Tool palette changes to reflect the different colors in the color palette.

9. With the rectangle still selected, choose **Arrange** —> **Send to Back**.

By creating the rectangle after the text, you've placed the rectangle above the text in the layout. Your rectangle is currently transparent, so this has little effect on the appearance of the button. But if you were to select a color for the rectangle with it still on top, it would obscure the text. For this reason, you move the rectangle behind the text.

You've created two graphic objects that make up your button: the rounded rectangle, and the text. In this section you'll group and align the two objects.

1. Select the rounded rectangle object.

2. Use Shift-Click to select the text object. Both objects should be selected.

3. Choose **Arrange** —> **Align Objects** to show the Alignment Settings.

4. Select the following alignment options:
   - align to each other
   - align centers horizontally
   - align centers vertically

5. Choose the OK button to perform the alignment.

6. With both objects still selected, choose **Arrange** —> **Group** to group the two objects.
Grouping the objects allows you to treat the text and the rectangle as one object. Notice that if you select the button now and move it, the text moves as well.

**Apply Color to the Button Object**

To make the button stand out more, let’s fill it in with some color. If your system does not support color, you can skip this section.

1. Select the button object group.
2. Choose the Fill Color button in the Layout editor to show the Fill Color palette.
   - You can tear off the palette and move it to another location on the layout by choosing the Tear Off Palette option in the palette and then dragging the palette by the title bar.
3. Choose a color from the palette to apply to the button object.
   - Be sure to select a color that is light enough so the button text is still legible. If necessary, repeat the above steps and select a different color.

**Set Edge Properties**

You’ve applied some color to the button. You can also define properties for the line around the button.

1. With the button object still selected, choose **Format**—>**Line**.
2. Choose the 4pt. option to give the button object a heavy weight.
3. Choose **Format**—>**Bevel**.
4. Choose one of the bevelling options to give the button a three dimensional appearance on the layout.

**Set Up the Button Procedure**

Now you can assign a button procedure using the grouped object’s Object property sheet.

1. Select the button object.
2. Select Properties from the pop-up menu, or choose **Tools**—>**Properties** to show the Object property sheet for the button.
3. Type the name `next_page` in the Name field.
4. Choose the New button to the right of the Procedure field to create a new button procedure.
   - The Program Unit editor appears.
Apply Color to the Button Object

Set Edge Properties

Set Up the Button Procedure

Grouping the objects allows you to treat the text and the rectangle as one object. Notice that if you select the button now and move it, the text moves as well.

To make the button stand out more, let's fill it in with some color. If your system does not support color, you can skip this section.

1. Select the button object group.
2. Choose the Fill Color button in the Layout editor to show the Fill Color palette.
   You can tear off the palette and move it to another location on the layout by choosing the Tear Off Palette option in the palette and then dragging the palette by the title bar.
3. Choose a color from the palette to apply to the button object.
   Be sure to select a color that is light enough so the button text is still legible. If necessary, repeat the above steps and select a different color.

You've applied some color to the button. You can also define properties for the line around the button.

1. With the button object still selected, choose Format —> Line.
2. Choose the 4pt. option to give the button object a heavy weight.
3. Choose Format —> Bevel.
4. Choose one of the bevelling options to give the button a three-dimensional appearance on the layout.

Now you can assign a button procedure using the grouped object's Object property sheet.

1. Select the button object.
2. Select Properties from the pop-up menu, or choose Tools —> Properties to show the Object property sheet for the button.
3. Type the name next_page in the Name field.
4. Choose the New button to the right of the Procedure field to create a new button procedure.

The Program Unit editor appears.
The Program Unit editor that appears contains a button procedure template named OGBUTTONPROCx by default. In this section you will create the body of the procedure to determine what action is taken when the button object is clicked on by a user at runtime.

The Program Unit editor contains the following initial content:

```plsql
PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object,
                           hitobj IN og_object,
                           win IN og_window,
                           eventinfo IN og_event) IS
  BEGIN
  END;
```

You will create two variables to store values for the two layers in the display. Type the following two lines immediately before the BEGIN line in the program unit:

```plsql
home_layr  OG_LAYER;
next_layr   OG_LAYER;
```

These variables will represent the different layers you will manipulate in this program unit, so they are defined using the Graphics built-in datatype OG_LAYER.

Graphics provides many built-in datatypes, functions, and procedures for use in your PL/SQL program units. Each of these are described in the Graphics Reference Manual.

Type the following lines immediately after the BEGIN line in the program unit to assign values to the above defined variables:

```plsql
home_layr:=OG_GET_LAYER('layer0');
next_layr:=OG_GET_LAYER('layer1');
```

The Graphics PL/SQL built-in function OG_GET_LAYER retrieves the internal name or handle for the layer objects in the display. The internal handle for each layer is then assigned to the variables you defined above.

To switch from one layer to another, you’re really doing two things: activating a new layer, and hiding the old one. When you activate a new layer, that layer is automatically shown. However, the old layer is not automatically hidden.

To perform these actions, you’ll use the Graphics PL/SQL built-in procedures OG_ACTIVATE_LAYER, and OG_HIDE_LAYER.
The Program Unit editor that appears contains a button procedure template named \texttt{OGBUTTONPROCx} by default. In this section you will create the body of the procedure to determine what action is taken when the button object is clicked on by a user at runtime.

The Program Unit editor contains the following initial content:

\begin{verbatim}
PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
BEGIN
END;
\end{verbatim}

You will create two variables to store values for the two layers in the display. Type the following two lines immediately before the \texttt{BEGIN} line in the program unit:

\begin{verbatim}
home_layr  OG_LAYER;
next_layr   OG_LAYER;
\end{verbatim}

These variables will represent the different layers you will manipulate in this program unit, so they are defined using the Graphics built-in datatype \texttt{OG\_LAYER}.

Graphics provides many built-in datatypes, functions, and procedures for use in your PL/SQL program units. Each of these are described in the Graphics Reference Manual.

Type the following lines immediately after the \texttt{BEGIN} line in the program unit to assign values to the above defined variables:

\begin{verbatim}
home_layr:=OG\_GET\_LAYER('layer0');
next_layr:=OG\_GET\_LAYER('layer1');
\end{verbatim}

The Graphics PL/SQL built-in function \texttt{OG\_GET\_LAYER} retrieves the internal name or \texttt{handle} for the layer objects in the display. The internal handle for each layer is then assigned to the variables you defined above.

To switch from one layer to another, you're really doing two things: activating a new layer, and hiding the old one. When you activate a new layer, that layer is automatically shown. However, the old layer is not automatically hidden.

To perform these actions, you'll use the Graphics PL/SQL built-in procedures \texttt{OG\_ACTIVATE\_LAYER}, and \texttt{OG\_HIDE\_LAYER}. 

Type the following lines after the variable assignment lines and before the END line:

```plaintext
OG_ACTIVATE_LAYER(next_layr);
OG_HIDE_LAYER(home_layr);
```

These procedures take the value assigned to the variables, and perform the specified action on the specified display objects. Notice that these procedures must be entered in a certain order. In this case, you cannot hide the active layer, so you must activate a new layer first.

Your program unit should now look like this:

```plaintext
PROCEDURE OGBUTTONPROC0(buttonobj IN og_object,
                          hitobj IN og_object,
                          win IN og_window,
                          eventinfo IN og_event) IS
  home_layr OG_LAYER;
  next_layr   OG_LAYER;
BEGIN
  home_layr:=OG_GET_LAYER('layer0');
  next_layr:=OG_GET_LAYER('layer1');
  OG_ACTIVATE_LAYER(next_layr);
  OG_HIDE_LAYER(home_layr);
END;
```

Once you’ve entered the body of the program unit, perform the following steps to apply it.

1. Choose the Compile button in the Program Unit editor to compile the procedure.

   Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines.

   If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.

   The program unit must be compiled without errors before you can proceed.

2. Choose the Close button to close the Program Unit editor.

3. Choose the OK button to close the Object property sheet.

Now that you’ve created the first button procedure, the second one will be easy!

1. Choose **Tools**—>**Layers** to show the Layers dialog.
2. Activate `layer1` and hide `layer0`.

---

**Complete the Button Procedure**

**Repeat for the Second Layer**

Type the following lines after the variable assignment lines and before the END line:

OG_ACTIVATE_LAYER(next_layr);
OG_HIDE_LAYER(home_layr);

These procedures take the value assigned to the variables, and perform the specified action on the specified display objects. Notice that these procedures must be entered in a certain order. In this case, you cannot hide the active layer, so you must activate a new layer first.

Your program unit should now look like this:

PROCEDURE OGBUTTONPROC0(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
  home_layr  OG_LAYER;
  next_layr   OG_LAYER;
BEGIN
  home_layr:=OG_GET_LAYER('layer0');
  next_layr:=OG_GET_LAYER('layer1');
  OG_ACTIVATE_LAYER(next_layr);
  OG_HIDE_LAYER(home_layr);
END;

Once you've entered the body of the program unit, perform the following steps to apply it.

1. Choose the Compile button in the Program Unit editor to compile the procedure. Any errors encountered appear below the procedure in the Compilation Messages pane. Check for common errors like a misspelling, or missing semicolons at the end of lines. If you did not encounter any errors, proceed to the next step. If you did encounter errors, correct them and re-compile.

The program unit must be compiled without errors before you can proceed.

2. Choose the Close button to close the Program Unit editor.

3. Choose the OK button to close the Object property sheet.

Now that you've created the first button procedure, the second one will be easy!

1. Choose Tools —> Layers to show the Layers dialog.
2. Activate layer1 and hide layer0.
3. Choose the OK button to close the Layers dialog.

4. Create another grouped button object, but use different text on the button, such as “Previous Page” or “Go Back.”

Because you have already specified alignment settings, you can use Arrange—>Repeat Alignment to align the button object and the text object before you group them.

Notice that the fill color you set for the previous button is still active. You can change the fill color if you wish.

5. Create a button procedure for the button object that activates layer0 and hides layer1.

The contents of your button procedure will be virtually the same, but the order of the layers will be switched: you will be activating Layer0 and hiding Layer1.

Note: Because variables are local to the program unit where they are declared, you can use the same variable names (home_layr and next_layr) in both program units. You must still declare the variables in the new program unit.

Reset the Layers Before you proceed to the final lesson, reactivate Layer0 and hide Layer1.

Save the Display Choose File—>Save or the Save button to save your changes to the display.

Continue or Exit Continue with the next lesson or choose File—>Exit to end your current Graphics session.

Lesson 11: View The Results

Let’s take a look at how your display will work in Graphics Runtime. In this lesson, you’ll get to test out some of the interactive procedures you’ve set up in this tutorial.

Restart Graphics If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File—>Close to close the default display that is presented.
3. Choose the OK button to close the Layers dialog.

4. Create another grouped button object, but use different text on the button, such as "Previous Page" or "Go Back.

Because you have already specified alignment settings, you can use Arrange —> Repeat Alignment to align the button object and the text object before you group them.

Notice that the fill color you set for the previous button is still active. You can change the fill color if you wish.

5. Create a button procedure for the button object that activates layer0 and hides layer1.

The contents of your button procedure will be virtually the same, but the order of the layers will be switched: you will be activating Layer0 and hiding Layer1.

Note: Because variables are local to the program unit where they are declared, you can use the same variable names (home_layr and next_layr) in both program units. You must still declare the variables in the new program unit.

Before you proceed to the final lesson, reactivate Layer0 and hide Layer1.

Choose File —> Save or the Save button to save your changes to the display.

Continue with the next lesson or choose File —> Exit to end your current Graphics session.

Lesson 11: View The Results

Let's take a look at how your display will work in Graphics Runtime. In this lesson, you'll get to test out some of the interactive procedures you've set up in this tutorial.

If you did not exit Graphics at the end of the previous lesson, you can skip this section of the lesson. If you did exit Graphics at the end of the previous lesson, perform the following steps:

1. Restart Graphics.
2. Choose File —> Close to close the default display that is presented.
3. Choose File—>Open or the Open button to open the display you saved at the end of the previous lesson.

4. Choose File—>Connect to connect to the database.

**Run the Display**

Graphics Designer provides a runtime simulation so you can test your displays as you build them. It is a good idea to save your displays right before you run them to make sure all changes are applied.

1. Choose File—>Run, or choose the Run button in the Layout editor to run the display.

   A Graphics Debugger Main Layout appears containing your charts in a “Runtime Simulation” mode. Notice that the runtime simulation window has no Toolbar, Tool palette, rulers or gridlines. The Layout editor window is minimized.

2. Click on a pie slice in the pie chart and watch the result in the column chart below to test the drill-down relationship you created in Lesson 4.

   Each time you select a pie slice, the value of the warehouse parameter changes, and the query for the column chart is re-executed using the new value in the WHERE clause.

   Notice that for each warehouse you select from the pie chart, different product ID numbers appear red in the column chart.

3. Click on the “Next Page” button to test the button procedure you created in Lesson 10.

   Did the layers switch? You should now see your large break chart and your original three charts should be hidden from view.

4. If you created it, click on the “Go Back” button to switch layers back again.

5. Choose File—>Close to close the Debugger display.

   Graphics closes the Debugger layout and returns you to the Designer Main Layout window.

3. Choose File —> Open or the Open button to open the display you saved at the end of the previous lesson.

4. Choose File —> Connect to connect to the database.

Graphics Designer provides a runtime simulation so you can test your displays as you build them. It is a good idea to save your displays right before you run them to make sure all changes are applied.

1. Choose File —> Run, or choose the Run button in the Layout editor to run the display.

A Graphics Debugger Main Layout appears containing your charts in a "Runtime Simulation" mode. Notice that the runtime simulation window has no Toolbar, Tool palette, rulers or gridlines. The Layout editor window is minimized.

2. Click on a pie slice in the pie chart and watch the result in the column chart below to test the drill-down relationship you created in Lesson 4.

Each time you select a pie slice, the value of the warehouse parameter changes, and the query for the column chart is re-executed using the new value in the WHERE clause.

Notice that for each warehouse you select from the pie chart, different product ID numbers appear red in the column chart.

3. Click on the "Next Page" button to test the button procedure you created in Lesson 10.

Did the layers switch? You should now see your large break chart and your original three charts should be hidden from view.

4. If you created it, click on the "Go Back" button to switch layers back again.

5. Choose File —> Close to close the Debugger display.

Graphics closes the Debugger layout and returns you to the Designer Main Layout window.

Congratulations! You’ve completed the Graphics tutorial. You’ve learned how to:

- create and modify charts
- create and modify graphic objects
- link charts together with a drill-down procedure
- create visual cues on a chart with a format trigger
- modify a “master” query using a query filter
- use layers to hide and show different aspects of your display
- create a button procedure on an object

The rest of this Guide provides information on all aspects of using Graphics to develop displays for use in Graphics Runtime, or to be embedded in other applications.
Congratulations! You've completed the Graphics tutorial. You've learned how to:

• create and modify charts
• create and modify graphic objects
• link charts together with a drill-down procedure
• create visual cues on a chart with a format trigger
• modify a “master” query using a query filter
• use layers to hide and show different aspects of your display
• create a button procedure on an object

The rest of this Guide provides information on all aspects of using Graphics to develop displays for use in Graphics Runtime, or to be embedded in other applications.
his chapter discusses the main components of Graphics and is
designed to introduce you to the basic concepts involved in working
with Graphics.

The following topics are covered in this chapter:

- components of Graphics – 3 – 2
- basic tasks – 3 – 3
- displays – 3 – 7
Understanding Graphics

This chapter discusses the main components of Graphics and is designed to introduce you to the basic concepts involved in working with Graphics.

The following topics are covered in this chapter:

• components of Graphics
• basic tasks
• displays
Components of Graphics

There are two major components of Graphics: the modules you create with Graphics, and the editors you use to create them.

Types of Modules

Modules are the top-level objects that you create with Developer/2000. You can develop modules to stand alone, or you can reference one or more modules from within another module. For example, you can build a display with Graphics, and from within it you can reference a report created with Reports. Integrating modules in this way allows you to create larger applications.

With Graphics, you can create two types of modules:

- displays
- PL/SQL libraries

Displays

Displays are modules unique to Graphics. They are binary files made up of components created with or imported into Graphics for the purpose of graphically and dynamically portraying information. Each display has a central theme or focus that unifies its various components and sets it apart as a distinct information resource.

You create displays using the Designer and test them using Procedure Builder functionality, also called the debugger. Separate Runtime and Batch executables are also available for running displays.

PL/SQL Libraries

PL/SQL libraries are collections of PL/SQL program units that you can create using Graphics. Unlike individual program units that you can store with displays, PL/SQL libraries are stored in the database or file system.

You can reference PL/SQL libraries within a single display or share them with other modules in an application. To reference PL/SQL libraries in Graphics, you need to attach them to your displays.

For more information about creating and attaching PL/SQL libraries, see NO TAG “Using PL/SQL Libraries” in chapter “Working with PL/SQL Constructs” of the Procedure Builder Developer’s Guide.
Components of Graphics

There are two major components of Graphics: the modules you create with Graphics, and the editors you use to create them. Modules are the top-level objects that you create with Developer/2000. You can develop modules to stand alone, or you can reference one or more modules from within another module. For example, you can build a display with Graphics, and from within it you can reference a report created with Reports. Integrating modules in this way allows you to create larger applications.

With Graphics, you can create two types of modules:

- **displays**
- **PL/SQL libraries**

Displays are modules unique to Graphics. They are binary files made up of components created with or imported into Graphics for the purpose of graphically and dynamically portraying information. Each display has a central theme or focus that unifies its various components and sets it apart as a distinct information resource.

You create displays using the Designer and test them using Procedure Builder functionality, also called the debugger. Separate Runtime and Batch executables are also available for running displays.

PL/SQL libraries are collections of PL/SQL program units that you can create using Graphics. Unlike individual program units that you can store with displays, PL/SQL libraries are stored in the database or file system.

You can reference PL/SQL libraries within a single display or share them with other modules in an application. To reference PL/SQL libraries in Graphics, you need to attach them to your displays. For more information about creating and attaching PL/SQL libraries, see "Using PL/SQL Libraries" in chapter "Working with PL/SQL Constructs" of the Procedure Builder Developer's Guide.
You can store modules in the database or file system, depending on the space you have available and the level of security you want to assign.

If you store a module in a file, the file’s security is assigned by your operating system and you can change it as you would for any other file.

If you store a module in the database, you are the only one who can access it unless you specifically grant access to other users. You can do this in the Designer using the File—>Administration—>Access menu item. Although people on the access list can use a module, only the creator of the module can change or grant access to it.

For more information about database security and storage requirements, see chapter “Storage” in the Graphics Reference Manual.

You create and manipulate components of displays using a variety of editors. Editors are work areas in which you can create, modify, and delete components of displays. Graphics editors include the following:

- Layout editor
- PL/SQL Program Unit editor
- Chart Template editor

The Layout editor is the work area in which you build displays. It has its own set of menus from which you can access all other editors and functions in the Designer. A display has only one Layout editor.

The PL/SQL Program Unit editor is the work area in which you can create, import, and debug PL/SQL program units. You can open multiple PL/SQL editors for a display.

The Chart Template editor is the work area in which you select and edit chart and field templates. A display has only one Chart Template editor.

The module that combines objects, sounds, queries, chart templates, and PL/SQL program units in Graphics is called a display. This section describes the basic tasks involved in building displays:

- creating a new display
- designing the visual framework
- specifying the data
You can store modules in the database or file system, depending on the space you have available and the level of security you want to assign. If you store a module in a file, the file's security is assigned by your operating system and you can change it as you would for any other file. If you store a module in the database, you are the only one who can access it unless you specifically grant access to other users. You can do this in the Designer using the File —> Administration —> Access menu item. Although people on the access list can use a module, only the creator of the module can change or grant access to it.

For more information about database security and storage requirements, see chapter "Storage" in the Graphics Reference Manual.

You create and manipulate components of displays using a variety of editors. Editors are work areas in which you can create, modify, and delete components of displays. Graphics editors include the following:

- Layout editor
- PL/SQL Program Unit editor
- Chart Template editor

The Layout editor is the work area in which you build displays. It has its own set of menus from which you can access all other editors and functions in the Designer. A display has only one Layout editor.

The PL/SQL Program Unit editor is the work area in which you can create, import, and debug PL/SQL program units. You can open multiple PL/SQL editors for a display.

The Chart Template editor is the work area in which you select and edit chart and field templates. A display has only one Chart Template editor.

Basic Tasks

The module that combines objects, sounds, queries, chart templates, and PL/SQL program units in Graphics is called a display. This section describes the basic tasks involved in building displays:

- creating a new display
- designing the visual framework
- specifying the data
Creating a New Display

The Graphics executable you use to build displays is called the Designer. This is where you’ll do most of your work in Graphics. When you invoke the Designer, you’ll see the following:

- the Graphics Object Navigator
- the Layout editor

The Graphics Object Navigator

Use the Graphics Object Navigator to create and manipulate all of the objects associated with your Graphics displays.

The Object Navigator provides a hierarchical overview of all currently open displays, and each display’s respective components. Components are represented as nodes in the Navigator which you can expand or collapse to view sub-components, if any.

The Object Navigator provides a Find facility you can use to search for and locate a specific object.

The Layout Editor

The Layout editor is the work area in which you build displays. Each display appears in its own Layout editor. When you invoke the Designer, a display with the default name Disp1 is opened for you to use.

The menus at the top provide access to all other editors and functions in the Designer. The Tool palette on the left lets you create and position charts, text, and artwork. The status line at the bottom provides information about the cursor position, mouse drag distance, rotation angle and magnification level, depending on the task being performed.

For more information about using and customizing the Layout Editor, see the “Working with the Layout Editor” chapter.

Designing the Visual Framework

The visual framework is the part of your display that end-users see and interact with. It should be balanced, logically organized, and focused on a central theme. If your display calls for user input, your visual framework needs to clearly communicate actions end-users should take.
Creating a New Display

The Graphics Object

Navigator

The Layout Editor

Designing the Visual Framework

The Graphics executable you use to build displays is called the Designer. This is where you'll do most of your work in Graphics. When you invoke the Designer, you'll see the following:

• the Graphics Object Navigator
• the Layout editor

Use the Graphics Object Navigator to create and manipulate all of the objects associated with your Graphics displays. The Object Navigator provides a hierarchical overview of all currently open displays, and each display's respective components. Components are represented as nodes in the Navigator which you can expand or collapse to view sub-components, if any.

The Object Navigator provides a Find facility you can use to search for and locate a specific object.

The Layout editor is the work area in which you build displays. Each display appears in its own Layout editor. When you invoke the Designer, a display with the default name Disp1 is opened for you to use.

The menus at the top provide access to all other editors and functions in the Designer. The Tool palette on the left lets you create and position charts, text, and artwork. The status line at the bottom provides information about the cursor position, mouse drag distance, rotation angle and magnification level, depending on the task being performed.

For more information about using and customizing the Layout Editor, see the "Working with the Layout Editor" chapter.

The visual framework is the part of your display that end-users see and interact with. It should be balanced, logically organized, and focused on a central theme. If your display calls for user input, your visual framework needs to clearly communicate actions end-users should take.
In designing the visual framework, you’ll use the following:

- objects
- Tool palette
- layout

**Objects**
Objects are the visual components of a display, and include charts, text, drawings, and images. You can create objects to stand alone as artwork, or you can associate them with data, as in a chart. You can also associate objects with a PL/SQL program unit, as in a button that end-users can click on at runtime to perform an operation.

**Tool Palette**
The Tool palette contains tools, represented by icons, you can use to draw and modify line-art objects in Graphics. To add bit-mapped images or clip-art to your layout, you need to import them from other sources.

**Layout**
The layout is the central area of the Layout editor on which you place objects. The layout can be composed of multiple surfaces, called layers, you can create to hide and show objects at different times.

**Specifying the Data**
You’ll probably need to retrieve data from a data source for many of your displays, whether they involve charts or other illustrations. The data you retrieve is placed in the Query property sheet.

Specifying the data may involve the following tasks:

- connecting to the database
- creating queries

**Connecting to the Database**
Whenever you perform operations involving a database, you need to connect to it. Connecting involves logging on to the database with a specific username and password. If you try to perform database operations when not connected, you will be prompted to do so first.

**Creating Queries**
Queries are SQL SELECT statements or files used to specify the data you want Graphics to retrieve. When you specify a query, Graphics retrieves the data and places it in the Data property page of the Query property sheet.

If your data source is a database, you’ll use a SQL SELECT statement as your query. If your data source is a file, such as a spreadsheet, you’ll specify the file.

For more information about queries, see the “Using Queries” chapter.
In designing the visual framework, you'll use the following:

- **Objects**: The visual components of a display, including charts, text, drawings, and images. You can create objects to stand alone as artwork, or associate them with data, such as in a chart. You can also associate objects with a PL/SQL program unit, such as a button that end-users can click on at runtime to perform an operation.

- **Tool palette**: Contains tools, represented by icons, that you can use to draw and modify line-art objects in Graphics. To add bit-mapped images or clip-art to your layout, you need to import them from other sources.

- **Layout**: The central area of the Layout editor on which you place objects. The layout can be composed of multiple layers, which you can create to hide and show objects at different times.

You'll probably need to retrieve data from a data source for many of your displays, whether they involve charts or other illustrations. The data you retrieve is placed in the Query property sheet.

Specifying the data may involve the following tasks:

- **Connecting to the database**: Whenever you perform operations involving a database, you need to connect to it. Connecting involves logging on to the database with a specific username and password. If you try to perform database operations when not connected, you will be prompted to do so first.

- **Creating queries**: Queries are SQL SELECT statements or files used to specify the data you want Graphics to retrieve. When you specify a query, Graphics retrieves the data and places it in the Data property page of the Query property sheet.

If your data source is a database, you'll use a SQL SELECT statement as your query. If your data source is a file, such as a spreadsheet, you'll specify the file.

For more information about queries, see the "Using Queries" chapter.
Creating a Chart

Charts are the most powerful objects in Graphics because of the wide variety of options they provide for representing data. The following tasks are involved in creating charts:

- drawing the chart
- updating the chart
- modifying the chart

Drawing the Chart

To draw a chart, use the Chart tool in the Tool palette to define a region on the layout. Then Graphics uses the query you’ve defined previously to draw the chart in the specified region.

Updating the Chart

To make a chart reflect new data in the Query property sheet or changes in the chart template, you need to update the chart. When you update a chart, Graphics looks for changes in the data and chart template, then re-draws the chart based on the new information.

Modifying the Chart

You can modify a chart’s elements directly in the layout, or through a chart template. If you plan to have multiple charts of the same type (bar, column, pie) in a display, you can define the settings for the appearance of your charts only once, by modifying the template. Graphics has a set of pre-defined templates you can use and customize to suit your needs.

For more information about charts, see the “Using Charts” chapter.

Testing the Display

Graphics Designer provides a built-in runtime simulation mode that you can use to test your Graphics displays as you build them. This allows you to create interactive displays, test them as Runtime displays at any time, and return to the Designer for any necessary trouble-shooting within one Graphics session.

Defining Parameters

Parameters are global variables you can use to assign and store values in Graphics. You define parameters in the Parameters dialog box and reference them in SQL SELECT statements or PL/SQL program units.

For more information about parameters, see the “Using Parameters” chapter.

Using PL/SQL

You can include PL/SQL in your displays to programatically create and modify objects or control certain operations. You can write PL/SQL program units within Graphics, or use your own editor and import or export program units.

For more information about PL/SQL, see the PL/SQL User’s Guide and Reference, and the “Working with PL/SQL Program Units” chapter in this Guide.
Charts are the most powerful objects in Graphics because of the wide variety of options they provide for representing data. The following tasks are involved in creating charts:

- **Drawing the Chart:** Use the Chart tool in the Tool palette to define a region on the layout. Then Graphics uses the query you've defined previously to draw the chart in the specified region.

- **Updating the Chart:** To make a chart reflect new data in the Query property sheet or changes in the chart template, you need to update the chart. When you update a chart, Graphics looks for changes in the data and chart template, then re-draws the chart based on the new information.

- **Modifying the Chart:** You can modify a chart's elements directly in the layout, or through a chart template. If you plan to have multiple charts of the same type (bar, column, pie) in a display, you can define the settings for the appearance of your charts only once, by modifying the template. Graphics has a set of pre-defined templates you can use and customize to suit your needs.

For more information about charts, see the "Using Charts" chapter.

Graphics Designer provides a built–in runtime simulation mode that you can use to test your Graphics displays as you build them. This allows you to create interactive displays, test them as Runtime displays at any time, and return to the Designer for any necessary trouble-shooting within one Graphics session.

Parameters are global variables you can use to assign and store values in Graphics. You define parameters in the Parameters dialog box and reference them in SQL SELECT statements or PL/SQL program units.

For more information about parameters, see the "Using Parameters" chapter.

You can include PL/SQL in your displays to programmatically create and modify objects or control certain operations. You can write PL/SQL program units within Graphics, or use your own editor and import or export program units.

For more information about PL/SQL, see the PL/SQL User's Guide and Reference, and the "Working with PL/SQL Program Units" chapter in this Guide.
**Setting Preferences**

You can specify your preferences for a number of different environment settings, such as:

- size of the layout
- rulers and grids
- magnification level
- hiding or showing layout editor elements

You can either save your settings for all Graphics sessions, or apply them only to your current session. Preferences are saved in a preferences file on your file system.

For more information about preferences, see “Customizing Your Work Environment” on page 4–32.

**Displays**

The components you use to build displays provide all of the information needed to run displays as stand-alone modules. Specifically, they:

- provide a visual framework for the presentation of data; retrieve data;
- dynamically link objects to data; and guide user interaction. These components include the following:
  - layout objects
  - chart templates
  - queries
  - parameters
  - sounds
  - PL/SQL program units

In most cases, you can import or export components outside of Graphics. All components except queries and parameters can be exported to the database or file system. The storage and security considerations that apply to modules also apply to components of displays.

Of the various components, objects and sounds are the only ones that will be apparent to end-users when a display is run. Queries, parameters, and PL/SQL program units function behind the scenes at runtime and are only accessible in the Designer.
You can specify your preferences for a number of different environment settings, such as:

- size of the layout
- rulers and grids
- magnification level
- hiding or showing layout editor elements

You can either save your settings for all Graphics sessions, or apply them only to your current session. Preferences are saved in a preferences file on your file system. For more information about preferences, see "Customizing Your Work Environment" on page 4–32.

The components you use to build displays provide all of the information needed to run displays as stand-alone modules. Specifically, they:

- provide a visual framework for the presentation of data
- retrieve data
- dynamically link objects to data
- guide user interaction

These components include the following:

- layout objects
- chart templates
- queries
- parameters
- sounds
- PL/SQL program units

In most cases, you can import or export components outside of Graphics. All components except queries and parameters can be exported to the database or file system. The storage and security considerations that apply to modules also apply to components of displays.

Of the various components, objects and sounds are the only ones that will be apparent to end-users when a display is run. Queries, parameters, and PL/SQL program units function behind the scenes at runtime and are only accessible in the Designer.
Objects are the visual components of a display, and include the following:

- **charts** Graphic diagrams you can create to represent data in a display. A chart is made up of a chart template and data retrieved with a query. You can use a chart to dynamically represent changing database values or you can convert it to a static artwork object.

- **drawings** Line-art graphics, such as polygons and other shapes you can create to provide a visual framework for your display.

- **images** Bit-mapped graphics created outside of Graphics (such as scanned photographs), which you can import to enhance the visual framework for your display.

- **text** Blocks of alphanumerical characters that you can type in your display to provide titles, label artwork, or communicate instructions to end-users.

You can create layout objects on one or more layers or surfaces within the display. You can let users dynamically hide and show different layers, or overlap them.

Perhaps the most powerful feature of Graphics is its ability to create dynamic, database-driven graphics. This means that you can make charts and other objects dependent upon the data source for their appearance and behavior in a display. At runtime, or whenever new data is retrieved, Graphics determines the data or conditions present and creates or manipulates the objects accordingly.

**Charts** The most common example of dynamic representation is a chart. Graphics needs data with which to initially draw a chart. After the chart is drawn, it remains linked to the data source. If the data changes, you can update the chart and Graphics will redraw it using the newly retrieved data. Alternatively, you can break a chart’s connection to the data source by converting it to a static artwork object.

When you create a chart manually, you specify the query and template to use and define its field properties and region on the layout. However, you can also do these things programmatically if you want the data at runtime to determine how a chart should appear.

**Other Objects** You can create objects manually by drawing them on the layout, and you can manipulate them using the interface elements in the Designer. This is practical if you want to present the same visual elements no matter what the data is, such as your company logo.
Objects are the visual components of a display, and include the following:

- Graphic diagrams you can create to represent data in a display. A chart is made up of a chart template and data retrieved with a query. You can use a chart to dynamically represent changing database values or you can convert it to a static artwork object.
- Line-art graphics, such as polygons and other shapes you can create to provide a visual framework for your display.
- Bit-mapped graphics created outside of Graphics (such as scanned photographs), which you can import to enhance the visual framework for your display.
- Blocks of alphanumeric characters that you can type in your display to provide titles, label artwork, or communicate instructions to end-users.

You can create layout objects on one or more layers or surfaces within the display. You can let users dynamically hide and show different layers, or overlap them.

Perhaps the most powerful feature of Graphics is its ability to create dynamic, database-driven graphics. This means that you can make charts and other objects dependent upon the data source for their appearance and behavior in a display. At runtime, or whenever new data is retrieved, Graphics determines the data or conditions present and creates or manipulates the objects accordingly.

**Charts**

The most common example of dynamic representation is a chart. Graphics needs data with which to initially draw a chart. After the chart is drawn, it remains linked to the data source. If the data changes, you can update the chart and Graphics will redraw it using the newly retrieved data. Alternatively, you can break a chart's connection to the data source by converting it to a static artwork object.

When you create a chart manually, you specify the query and template to use and define its field properties and region on the layout. However, you can also do these things programmatically if you want the data at runtime to determine how a chart should appear.

**Other Objects**

You can create objects manually by drawing them on the layout, and you can manipulate them using the interface elements in the Designer. This is practical if you want to present the same visual elements no matter what the data is, such as your company logo.
However, if you want objects to change based on data or user input at runtime, you can create and modify them dynamically. Using PL/SQL, you can define an object’s properties and position on the layout. Then you can set up certain conditions under which objects or their properties should appear.

Object Properties

Object properties provide information such as appearance or physical characteristics, or an object’s relationship to other objects.

The properties that apply and how you define them vary depending on the type of object. In general, you define an object’s properties by selecting the object, choosing a Tools or Format menu command, and specifying the options you want to define for that object. You can also define an object’s properties programmatically, using PL/SQL.

The following are examples of properties you can define for objects:

- name, such as orgchart or pie_temp
- associated component, such as a PL/SQL procedure or a query
- associated event type, such as mouse up or mouse down
- text properties, such as font or size
- line properties, such as weight or style
- color properties, such as foreground fill color
- date and number formats, such as mm–dd–yy or 9,999

Chart Templates

Chart templates are definition files that determine the format of a chart and its various properties. Graphics has pre-defined chart templates you can use and customize for your displays.

You can also modify the format of a chart directly in the Layout Editor. Chart templates are recommended if you plan to have multiple charts of a given type in one display.

Queries

Queries are SQL SELECT statements or files used to retrieve data for a display. In most cases, you’ll use queries to retrieve data for charts. However, you can also execute a query programmatically and use the data to create and modify other types of objects.

You can create any number of queries for a display. Once you’ve defined a query, you can reference it multiple times or use it for multiple charts. The type of query you use depends on your data source. If your data source is a file, such as a spreadsheet, you specify the file. If your data source is a database, you use a SQL SELECT statement to retrieve a specific set of values.
However, if you want objects to change based on data or user input at runtime, you can create and modify them dynamically. Using PL/SQL, you can define an object’s properties and position on the layout. Then you can set up certain conditions under which objects or their properties should appear.

Object properties provide information such as appearance or physical characteristics, or an object’s relationship to other objects. The properties that apply and how you define them vary depending on the type of object. In general, you define an object’s properties by selecting the object, choosing a Tools or Format menu command, and specifying the options you want to define for that object. You can also define an object’s properties programmatically, using PL/SQL.

The following are examples of properties you can define for objects:

• name, such as orgchart or pie_temp
• associated component, such as a PL/SQL procedure or a query
• associated event type, such as mouse up or mouse down
• text properties, such as font or size
• line properties, such as weight or style
• color properties, such as foreground fill color
• date and number formats, such as mm–dd–yy or 9,999

Chart templates are definition files that determine the format of a chart and its various properties. Graphics has pre-defined chart templates you can use and customize for your displays. You can also modify the format of a chart directly in the Layout Editor. Chart templates are recommended if you plan to have multiple charts of a given type in one display.

Queries are SQL SELECT statements or files used to retrieve data for a display. In most cases, you’ll use queries to retrieve data for charts. However, you can also execute a query programmatically and use the data to create and modify other types of objects.

You can create any number of queries for a display. Once you’ve defined a query, you can reference it multiple times or use it for multiple charts. The type of query you use depends on your data source. If your data source is a file, such as a spreadsheet, you specify the file. If your data source is a database, you use a SQL SELECT statement to retrieve a specific set of values.
For more information about creating SQL SELECT statements, see the Oracle7 Server SQL Language Reference Manual.

Data Classification

Each column in a query contains one of two types of data:

**Categories**
A category is a piece of data that does not depend on other data for its value. For example, an employee’s name may have the value *Jones*, which is independent of the values of other employee’s names or associated data.

In a chart, categories are usually plotted along the discrete axis (also called the category axis) and are not mathematically related.

**Values**
A value is a piece of data that depends on other data for its value. For example, an employee’s salary may have the value 1600, which is dependent upon the value of the employee’s name.

In a chart, values are usually plotted along the continuous axis (also called the value axis) and are mathematically related.

Parameters

Parameters are global variables to which you can assign values to be used by your display at runtime. You can reference parameters in SQL SELECT statements or PL/SQL program units. Parameters are useful if you want to reference the same variable from more than one location.

A parameter has a name, a datatype (NUMBER, CHAR, or DATE), and an initial value that may be superseded at runtime depending on how the parameter is used.

Referencing

You can use either bind or lexical references to reference a parameter. The type of reference you use depends on the data type of the parameter, and the parameter’s value:

<table>
<thead>
<tr>
<th>Reference Type</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>a single word, number, or date</td>
</tr>
<tr>
<td>lexical</td>
<td>a compound character string</td>
</tr>
</tbody>
</table>

Bind references are used in both PL/SQL program units, and SQL SELECT statements. You make a bind reference by preceding the parameter name with a colon (:).

Lexical references are generally used in SQL SELECT statements — for example, to represent a WHERE clause. You make a lexical reference by preceding the parameter name with an ampersand (&).
For more information about creating SQL SELECT statements, see the Oracle7 Server SQL Language Reference Manual.

Each column in a query contains one of two types of data:

A category is a piece of data that does not depend on other data for its value. For example, an employee's name may have the value Jones, which is independent of the values of other employee's names or associated data.

In a chart, categories are usually plotted along the discrete axis (also called the category axis) and are not mathematically related.

A value is a piece of data that depends on other data for its value. For example, an employee's salary may have the value 1600, which is dependent upon the value of the employee's name.

In a chart, values are usually plotted along the continuous axis (also called the value axis) and are mathematically related.

Parameters are global variables to which you can assign values to be used by your display at runtime. You can reference parameters in SQL SELECT statements or PL/SQL program units. Parameters are useful if you want to reference the same variable from more than one location.

A parameter has a name, a datatype (NUMBER, CHAR, or DATE), and an initial value that may be superseded at runtime depending on how the parameter is used.

You can use either bind or lexical references to reference a parameter. The type of reference you use depends on the data type of the parameter, and the parameter's value:

<table>
<thead>
<tr>
<th>Reference Type</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>a single word, number, or date</td>
</tr>
<tr>
<td>lexical</td>
<td>a compound character string</td>
</tr>
</tbody>
</table>

Bind references are used in both PL/SQL program units, and SQL SELECT statements. You make a bind reference by preceding the parameter name with a colon (:).

Lexical references are generally used in SQL SELECT statements — for example, to represent a WHERE clause. You make a lexical reference by preceding the parameter name with an ampersand (&).
Assigning Values  You can assign values to parameters in three ways:
  • initially, in the Parameters dialog box
  • within PL/SQL program units
  • as arguments to the Runtime or Batch executables
At runtime, the values of the parameters are substituted for their references in the SQL SELECT statements or PL/SQL program units.

Sounds  Sounds are audio recordings you can create in Graphics if your system supports sound. You can use sounds in your displays to alert end-users to specific actions, or cue them for input.

PL/SQL Program Units  PL/SQL program units are code structures using the PL/SQL programming language, which you can use in your displays to create and modify objects, control the flow of operations, interpret end-user actions, or cause certain operations to occur conditionally.
You can assign values to parameters in three ways:

- Initially, in the Parameters dialog box
- Within PL/SQL program units
- As arguments to the Runtime or Batch executables

At runtime, the values of the parameters are substituted for their references in the SQL SELECT statements or PL/SQL program units.

Sounds are audio recordings you can create in Graphics if your system supports sound. You can use sounds in your displays to alert end-users to specific actions, or cue them for input.

PL/SQL program units are code structures using the PL/SQL programming language, which you can use in your displays to create and modify objects, control the flow of operations, interpret end-user actions, or cause certain operations to occur conditionally.
Working with Graphics
PART II

II
his chapter provides step-by-step instructions on how to perform Graphics tasks with the Layout editor.

The following topics are covered in this chapter:

- components of the Layout Editor – 4 – 2
- working with objects – 4 – 6
- manipulating objects – 4 – 15
- customizing your work environment – 4 – 32
- using windows – 4 – 37
CHAPTER 4

Working with the Layout Editor

This chapter provides step-by-step instructions on how to perform Graphics tasks with the Layout editor. The following topics are covered in this chapter:

• components of the Layout Editor – 4 – 2
• working with objects – 4 – 6
• manipulating objects – 4 – 15
• customizing your work environment – 4 – 32
• using windows – 4 – 37
Components of the Layout Editor

The Layout editor is the main work area in the Graphics Designer. In it you design the layout of your display, by creating and modifying graphical objects (lines, polygons, rectangles, text, etc.).

Every display contains exactly one layout, and every layout belongs to exactly one display.

When you first run a display, Graphics shows the contents of the layout. The user can view the information presented, or interact with the display with a mouse or keyboard.

The Layout editor has the following components:

- tool bar
- tool palette
- status line
- layout

The figure below shows the locations of these components:
The Layout editor is the main work area in the Graphics Designer. In it you design the layout of your display, by creating and modifying graphical objects (lines, polygons, rectangles, text, etc.). Every display contains exactly one layout, and every layout belongs to exactly one display.

When you first run a display, Graphics shows the contents of the layout. The user can view the information presented, or interact with the display with a mouse or keyboard.

The Layout editor has the following components:

- **tool bar**
- **tool palette**
- **status line**
- **layout**

The figure below shows the locations of these components:

![Diagram of Layout Editor Components]

- Tool Bar
- Tool Palette
- Status Line
- Layout
| **Layout** | The layout is where you can create, modify, or delete objects. The appearance of objects on the layout represents exactly how those objects will appear when the display is run. In the Layout editor, you can change the size, shape, color, and other graphical properties of objects, and see the results immediately. For more information about setting options for the Layout editor, see the “Customizing Your Work Environment” section on page 4 – 32. |
| **Objects** | The graphical objects that you can create and modify on the layout include arcs, charts, images, lines, polygons, rectangles, rounded rectangles, symbols, and text objects. |
| **Rulers** | Rulers are located around the left and top edges of the layout, and can be used as a reference when sizing or positioning objects. To show or hide the rulers, choose **View**—>**Rulers**. Rulers are shown by default. |
| **Ruler Markers** | Ruler markers are lines that appear in both rulers, and that move to follow the horizontal and vertical position of the cursor. This makes it easier to position the cursor more precisely, by using the ruler markers as a reference. |
| **Ruler Guides** | A ruler guide is a dashed line on the layout that is parallel to a ruler, and can be used as a point of reference when sizing or positioning objects. Ruler guides appear on the layout in the Designer only, and are not visible when the display is run or printed. To show or hide ruler guides, choose **View**—>**Ruler Guides**. Ruler guides are shown by default. |
| **Grid** | The layout grid is a series of equidistant horizontal and vertical dashed lines that you can use as a reference when sizing or positioning objects. You can set the interval at which grid lines appear, based on the rulers’ unit of measurement. Grid lines appear on the layout in the Designer only, and are not visible when the display is run or printed. To show or hide the grid, choose **View**—>**Grid**. The grid is shown by default. |
| **Grid Snap** | Grid snap is a feature that forces the cursor to align itself with the grid whenever you move it close to a grid line. This provides greater accuracy and control when sizing or positioning objects. You can set the number of “snap points” between each pair of grid lines that the cursor will snap to. To activate or deactivate grid snap, choose **View**—>**Grid Snap**. Grid snap is deactivated by default. |
The layout is where you can create, modify, or delete objects. The appearance of objects on the layout represents exactly how those objects will appear when the display is run. In the Layout editor, you can change the size, shape, color, and other graphical properties of objects, and see the results immediately.

For more information about setting options for the Layout editor, see the “Customizing Your Work Environment” section on page 4–32.

The graphical objects that you can create and modify on the layout include arcs, charts, images, lines, polygons, rectangles, rounded rectangles, symbols, and text objects.

Rulers are located around the left and top edges of the layout, and can be used as a reference when sizing or positioning objects. To show or hide the rulers, choose View —> Rulers. Rulers are shown by default.

Ruler markers are lines that appear in both rulers, and that move to follow the horizontal and vertical position of the cursor. This makes it easier to position the cursor more precisely, by using the ruler markers as a reference.

A ruler guide is a dashed line on the layout that is parallel to a ruler, and can be used as a point of reference when sizing or positioning objects. Ruler guides appear on the layout in the Designer only, and are not visible when the display is run or printed.

To show or hide ruler guides, choose View —> Ruler Guides. Ruler guides are shown by default.

The layout grid is a series of equidistant horizontal and vertical dashed lines that you can use as a reference when sizing or positioning objects. You can set the interval at which grid lines appear, based on the rulers' unit of measurement. Grid lines appear on the layout in the Designer only, and are not visible when the display is run or printed.

To show or hide the grid, choose View —> Grid. The grid is shown by default.

Grid snap is a feature that forces the cursor to align itself with the grid whenever you move it close to a grid line. This provides greater accuracy and control when sizing or positioning objects. You can set the number of “snap points” between each pair of grid lines that the cursor will snap to.

To activate or deactivate grid snap, choose View —> Grid Snap. Grid snap is deactivated by default.
Views

A view is a part of the layout that is focused on one small section of your display. Each layout is initially shown with one large view. However, you can create up to four small views, each focusing on a different part of the layout.

Tool Bar

The Tool bar runs horizontally across the top of the Layout editor and provides buttons that perform common menu commands such as Save, Run, Cut, Copy, and Paste.

Tool Palette

The tools you use for drawing and modifying objects on the layout appear as separate icons in the Tool palette. To use a tool, you need to activate it. Only one tool can be active at a time. The Select tool is the default tool.

The following tools are provided:
A view is a part of the layout that is focused on one small section of your display. Each layout is initially shown with one large view. However, you can create up to four small views, each focusing on a different part of the layout.

The Tool bar runs horizontally across the top of the Layout editor and provides buttons that perform common menu commands such as Save, Run, Cut, Copy, and Paste.

The tools you use for drawing and modifying objects on the layout appear as separate icons in the Tool palette. To use a tool, you need to activate it. Only one tool can be active at a time. The Select tool is the default tool.

The following tools are provided:

- Select
- Magnify
- Rectangle
- Ellipse
- Polygon
- Rounded Rectangle
- Chart
- Symbol
- Line/Fill/Text
- Display Area

- Rotate
- Reshape
- Line
- Arc
- Polyline
- Freehand
- Text
- Text Field

- Fill Color Palette
- Line Color Palette
- Text Color Palette
- Symbol Palette
For more information on using these tools to create and manipulate objects, see the following sections in this chapter:

- “Creating Shapes” – 4 – 7
- “Creating Text and Text Fields” – 4 – 9
- “Using Symbols” – 4 – 14
- “Selecting and Deselecting Objects” – 4 – 16
- “Moving Objects” – 4 – 18
- “Resizing Objects” – 4 – 20
- “Reshaping Objects” – 4 – 21
- “Rotating Objects” – 4 – 22

Since charts are more complex than the other types of objects, we’ve included them in a separate chapter. For more information, see “Using Charts” on page 8 – 1.

Activating a Tool

To activate a tool for single or multiple operations, do one of the following:

- To activate a tool for a single operation, click on its icon. After you perform the operation, the default tool becomes active again.
- To activate a tool for multiple operations, double-click on its icon. The entire icon is highlighted and the tool remains active until you activate another tool.

Status Line

The status line shows the following information:

- absolute mouse position (in the currently selected ruler unit)
- mouse drag distance (in the currently selected ruler unit)
- rotation angle
- zoom level

The information that appears on the status line varies based on the tool being used and the action being performed. The following table shows what is shown on the status line under different circumstances:
To activate a tool for single or multiple operations, do one of the following:

- To activate a tool for a single operation, click on its icon. After you perform the operation, the default tool becomes active again.
- To activate a tool for multiple operations, double-click on its icon. The entire icon is highlighted and the tool remains active until you activate another tool.

The status line shows the following information:

- Absolute mouse position (in the currently selected ruler unit)
- Mouse drag distance (in the currently selected ruler unit)
- Rotation angle
- Zoom level

The information that appears on the status line varies based on the tool being used and the action being performed. The following table shows what is shown on the status line under different circumstances:
### Tool Actions Status Information

<table>
<thead>
<tr>
<th>Tool</th>
<th>Action</th>
<th>Status Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Tool</td>
<td>moving mouse</td>
<td>x, y coordinates of cursor; zoom level</td>
</tr>
<tr>
<td>Arc, Ellipse, Rectangle, Rounded Rectangle</td>
<td>creating object</td>
<td>x, y coordinates of cursor starting position; object width and height; zoom level</td>
</tr>
<tr>
<td>Freehand</td>
<td>creating object</td>
<td>x, y coordinates of cursor; zoom level</td>
</tr>
<tr>
<td>Line</td>
<td>creating object</td>
<td>x, y coordinates of cursor; line length; rotation angle (from horizontal); zoom level</td>
</tr>
<tr>
<td>Magnify</td>
<td>clicking mouse</td>
<td>zoom level</td>
</tr>
<tr>
<td>Polygon, Polyline</td>
<td>creating object</td>
<td>x, y coordinates of cursor; rotation angle (from horizontal); zoom level</td>
</tr>
<tr>
<td>Rotate</td>
<td>moving object</td>
<td>horizontal and vertical distance from original position; zoom level</td>
</tr>
<tr>
<td></td>
<td>rotating object</td>
<td>rotation angle from original position; zoom level</td>
</tr>
<tr>
<td>Select</td>
<td>moving object</td>
<td>horizontal and vertical distance from original position; zoom level</td>
</tr>
</tbody>
</table>

### Working with Objects

Objects are the visual elements on your layout. This section describes how to perform the following tasks with objects:

- creating shapes
- creating text and text fields
- using symbols

Since charts are more complex than the other types of objects, we’ve included them in a separate chapter. For more information, see “Using Charts” on page 8 – 1.

In addition to creating artwork within Graphics, you can import drawing and image files from third party applications. For more information, see the section “Importing and Exporting Artwork” on page 6 – 4.
Tool | Action | Status | Information
---|---|---|---
Every Tool | moving mouse | | x, y coordinates of cursor; zoom level
Arc, Ellipse, Rectangle, Rounded Rectangle | creating object | | x, y coordinates of cursor; object width and height; zoom level
Freehand | creating object | | x, y coordinates of cursor; zoom level
Line | creating object | | x, y coordinates of cursor; line length; rotation angle (from horizontal); zoom level
Magnify | clicking mouse | | zoom level
Polygon, Polyline | creating object | | x, y coordinates of cursor; rotation angle (from horizontal); zoom level
Rotate | moving object | | horizontal and vertical distance from original position; zoom level
Select | moving object | | horizontal and vertical distance from original position; zoom level

Working with Objects

Objects are the visual elements on your layout. This section describes how to perform the following tasks with objects:

- creating shapes
- creating text and text fields
- using symbols

Since charts are more complex than the other types of objects, we've included them in a separate chapter. For more information, see "Using Charts" on page 8 – 1.

In addition to creating artwork within Graphics, you can import drawing and image files from third party applications. For more information, see the section "Importing and Exporting Artwork" on page 6 – 4.
Creating Shapes

Using shapes, you can create artwork elements for your displays. Shapes you can create include the following:

- arcs
- ellipses
- freehand shapes
- lines
- polygons
- polylines
- rectangles
- rounded rectangles

For more information about specifying default drawing options, see “Specifying Drawing Options” on page 4 – 37.

Creating Arcs

To draw an arc on the layout:

1. Select the Arc tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired arc, then release.

  **Suggestion:** To draw the arc of a circle, hold down the Shift key while dragging.

Creating Ellipses

To draw an ellipse on the layout:

1. Select the Ellipse tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired ellipse, then release.

  **Suggestion:** To draw a circle, hold down the Shift key while dragging.

Creating Freehand Shapes

To draw a freehand shape on the layout:

1. Select the Freehand tool from the Tool palette.
2. Click-and-drag on the layout to define the desired freehand shape, then release.

  The Freehand tool cannot be constrained.

Creating Lines

To draw a line on the layout:

1. Select the Line tool from the Tool palette.
Using shapes, you can create artwork elements for your displays. Shapes you can create include the following:

- arcs
- ellipses
- freehand shapes
- lines
- polygons
- polylines
- rectangles
- rounded rectangles

For more information about specifying default drawing options, see "Specifying Drawing Options" on page 4–37.

To draw an arc on the layout:

1. Select the Arc tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired arc, then release.

Suggestion: To draw the arc of a circle, hold down the Shift key while dragging.

To draw an ellipse on the layout:

1. Select the Ellipse tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired ellipse, then release.

Suggestion: To draw a circle, hold down the Shift key while dragging.

To draw a freehand shape on the layout:

1. Select the Freehand tool from the Tool palette.
2. Click-and-drag on the layout to define the desired freehand shape, then release.

The Freehand tool cannot be constrained.

To draw a line on the layout:

1. Select the Line tool from the Tool palette.
2. Click on the layout at the desired start point, drag to the desired end point, then release.

Suggestion: To draw a horizontal, vertical, or 45 degree line, hold down the Shift key while dragging in the desired direction.

For more information about setting line and edge properties (e.g., edge width, arrows, etc.), see “Setting Line and Edge Properties” on page 4 – 29.

Creating Polygons

To draw a polygon on the layout:

1. Select the Polygon tool from the Tool palette.
2. Click on the layout at the desired location of the first vertex, then click at the next vertex, etc. To finish drawing the polygon, double-click at the final vertex.

A line is drawn connecting the first and last vertices.

Suggestion: To draw a horizontal, vertical, or 45 degree segment, hold down the Shift key while defining a vertex. Constraining one vertex does not affect subsequent vertices.

Creating Polylines

To draw a polyline on the layout:

1. Select the Polyline tool from the Tool palette.
2. Click on the layout at the desired location of the first vertex, then click at the next vertex, etc. To finish drawing the polyline, double-click at the final vertex.

Unless you manually connect the first and last vertex, no line is created connecting them as with the polygon.

Suggestion: To draw a horizontal, vertical, or 45 degree segment, hold down the Shift key while defining a vertex.

Creating Rectangles

To draw a rectangle on the layout:

1. Select the Rectangle tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired rectangle, then release.

Suggestion: To draw a square, hold down the Shift key while dragging.

Creating Rounded Rectangles

To draw a rounded rectangle on the layout:

1. Select the Rounded Rectangle tool from the Tool palette.
2. Click on the layout at the desired start point, drag to the desired end point, then release.

Suggestion:
To draw a horizontal, vertical, or 45 degree line, hold down the Shift key while dragging in the desired direction.

For more information about setting line and edge properties (e.g., edge width, arrows, etc.), see “Setting Line and Edge Properties” on page 4–29.

To draw a polygon on the layout:
1. Select the Polygon tool from the Tool palette.
2. Click on the layout at the desired location of the first vertex, then click at the next vertex, etc. To finish drawing the polygon, double-click at the final vertex. A line is drawn connecting the first and last vertices.

Suggestion:
To draw a horizontal, vertical, or 45 degree segment, hold down the Shift key while defining a vertex. Constraining one vertex does not affect subsequent vertices.

To draw a polyline on the layout:
1. Select the Polyline tool from the Tool palette.
2. Click on the layout at the desired location of the first vertex, then click at the next vertex, etc. To finish drawing the polyline, double-click at the final vertex. Unless you manually connect the first and last vertex, no line is created connecting them as with the polygon.

Suggestion:
To draw a horizontal, vertical, or 45 degree segment, hold down the Shift key while defining a vertex.

To draw a rectangle on the layout:
1. Select the Rectangle tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired rectangle, then release.

Suggestion:
To draw a square, hold down the Shift key while dragging.

To draw a rounded rectangle on the layout:
1. Select the Rounded Rectangle tool from the Tool palette.
2. Click-and-drag diagonally on the layout to define the desired rounded rectangle, then release.

   **Suggestion:** To draw a rounded square, hold down the Shift key while dragging.

Creating Shapes with PL/SQL

You can also create a shape using PL/SQL. The following example creates a large, colored rectangle to be used as a background for other objects and charts:

```plsql
PROCEDURE create_background IS
    the_back   OG_OBJECT;
    rect_pos   OG_POINT;
    rect_ht    NUMBER;
    rect_wi    NUMBER;
BEGIN
    /*Set the start position of the rectangle*/
    rect_pos.x:=.5*OG_INCH;
    rect_pos.y:=.5*OG_INCH;
    /*Set the rectangle’s height and width*/
    rect_ht:=6*OG_INCH;
    rect_wi:=4*OG_INCH;
    /*Make the rounded rectangle object*/
    the_back:=OG_MAKE_RRECT(rect_pos, rect_ht, rect_wi);
    /*Set the object’s foreground fill color and pattern*/
    OG_SET_FFCOLOR(the_back, 'color 25');
    OG_SET_FILLPAT(the_back, 'gray50');
END;
```

The above example uses PL/SQL property built-ins for the rounded rectangle and graphic objects.

For more information about setting an object’s size and position using OG_INCH, see “Specifying Coordinate and Size Information” on page 13 – 5.


Creating Text and Text Fields

You can create two types of text object: text and text fields. You can use text to provide titles, label artwork elements, or communicate instructions to end-users. You use text fields to provide input areas in which end-users can type text at runtime.
Creating Shapes with PL/SQL

Creating Text and Text Fields

2. Click-and-drag diagonally on the layout to define the desired rounded rectangle, then release.

Suggestion: To draw a rounded square, hold down the Shift key while dragging.

You can also create a shape using PL/SQL. The following example creates a large, colored rectangle to be used as a background for other objects and charts:

```plsql
PROCEDURE create_background IS
  the_back OG_OBJECT;
  rect_pos OG_POINT;
  rect_ht NUMBER;
  rect_wi NUMBER;
BEGIN
  /*Set the start position of the rectangle*/
  rect_pos.x:=.5*OG_INCH;
  rect_pos.y:=.5*OG_INCH;
  /*Set the rectangle's height and width*/
  rect_ht:=6*OG_INCH;
  rect_wi:=4*OG_INCH;
  /*Make the rounded rectangle object*/
  the_back:=OG_MAKE_RRECT(rect_pos, rect_ht, rect_wi);
  /*Set the object's foreground fill color and pattern*/
  OG_SET_FFCOLOR(the_back, 'color 25');
  OG_SET_FILLPAT(the_back, 'gray50');
END;
```

The above example uses PL/SQL property built-ins for the rounded rectangle and graphic objects.

For more information about setting an object's size and position using OG_INCH, see "Specifying Coordinate and Size Information" on page 13 – 5.


You can create two types of text object: text and text fields. You can use text to provide titles, label artwork elements, or communicate instructions to end-users. You use text fields to provide input areas in which end-users can type text at runtime.
Creating Text Objects

You can create, edit, and set properties for all or part of a text object. To create a text object:

1. Select the Text tool from the Tool palette.
2. Do one of the following to create a bounding box containing the text cursor in the layout:
   - Click on the layout where you want the text object to start. The text will word wrap to the next line only when you type Return.
   - Drag out a specific region for the text object to occupy. The text will word wrap to fit within the horizontal boundaries of the box. If necessary, the box will grow vertically to accommodate the text you type.
3. Type the text.
   Use the Return key to force a line break.
4. When you’re finished, click on the layout outside the text object to exit text mode.

Note that if you create a text object without typing any text, the object is automatically deleted. To create an empty text object, first create it with some text, then edit it and delete the text. You can create an empty text object only if it has a fixed bounding box.

Editing Text Objects

To edit a text object:

1. Select the Text tool from the Tool palette.
2. Click on the text object where you want to begin typing or editing. The text cursor appears where you clicked.
3. Type or edit the text.
   You can select a block of text by dragging over it.
4. When you’re finished, click on the layout outside the text object to exit text mode.

Creating Text Field Objects

You can create an input field in which end-users can type text at runtime. You can also enter default text which the end-user can change.

For more information about specifying default drawing options, see “Specifying Drawing Options” on page 4 – 37.

For more information about specifying color and line options for text objects, see “Setting Patterns and Colors for Objects” on page 4 – 25.
For more information about specifying default drawing options, see "Specifying Drawing Options" on page 4–37.

For more information about specifying color and line options for text objects, see "Setting Patterns and Colors for Objects" on page 4–25.

You can create, edit, and set properties for all or part of a text object. To create a text object:

1. Select the Text tool from the Tool palette.
2. Do one of the following to create a bounding box containing the text cursor in the layout:
   - Click on the layout where you want the text object to start. The text will word wrap to the next line only when you type Return.
   - Drag out a specific region for the text object to occupy. The text will word wrap to fit within the horizontal boundaries of the box. If necessary, the box will grow vertically to accommodate the text you type.
3. Type the text. Use the Return key to force a line break.
4. When you're finished, click on the layout outside the text object to exit text mode.

Note that if you create a text object without typing any text, the object is automatically deleted. To create an empty text object, first create it with some text, then edit it and delete the text. You can create an empty text object only if it has a fixed bounding box.

To edit a text object:

1. Select the Text tool from the Tool palette.
2. Click on the text object where you want to begin typing or editing. The text cursor appears where you clicked.
3. Type or edit the text. You can select a block of text by dragging over it.
4. When you're finished, click on the layout outside the text object to exit text mode.

You can create an input field in which end-users can type text at runtime. You can also enter default text which the end-user can change.
To create a text field object:
1. Select the Text Field tool from the Tool palette.
2. Click-and-drag on the layout to define a rectangular bounding box, then release.

    **Suggestion:** To create a square text field, hold down the Shift key while dragging.

To enter default text in the text field:
1. Select the Text tool from the Tool palette.
2. Click on the text field object. A text cursor appears where you clicked.
3. Type the default text.
4. When you’re finished, click on the layout outside the text field object to exit text mode.

    At runtime, the default text will appear in the text field. Users can change this text.

---

### Setting Text Properties

You can set text properties such as font, spacing, and justification for a whole text object or text field, or part of a text object.

For instructions on how to set color for a text object or text in a text field, see “Setting Colors for Text” on page 4–29.

**Whole Text Object or Text Field** To set text properties for a whole text object or text field:
1. Select the text object or text field.
2. Choose **Format—>Font, Spacing, or Justification** to show a submenu or dialog (Font) of options.
3. Specify the desired options.
   The properties you select are applied to the text object or text field.
4. When you’re finished, click on the layout outside the text object to deselect the text object or text field.

**Partial Text Object or Default Text Field Text** To set text properties for part of a text object or part of the default text in a text field:
1. Select the Text tool from the Tool palette.
2. Click on the text object or text field where you want to change properties.
   The text cursor appears where you clicked.
To create a text field object:
1. Select the Text Field tool from the Tool palette.
2. Click-and-drag on the layout to define a rectangular bounding box, then release.

Suggestion: To create a square text field, hold down the Shift key while dragging.

To enter default text in the text field:
1. Select the Text tool from the Tool palette.
2. Click on the text field object.
A text cursor appears where you clicked.
3. Type the default text.
4. When you're finished, click on the layout outside the text field object to exit text mode.

At runtime, the default text will appear in the text field. Users can change this text.

You can set text properties such as font, spacing, and justification for a whole text object or text field, or part of a text object.

For instructions on how to set color for a text object or text in a text field, see "Setting Colors for Text" on page 4 – 29.

**Whole Text Object or Text Field**

To set text properties for a whole text object or text field:
1. Select the text object or text field.
2. Choose Format —> Font, Spacing, or Justification to show a submenu or dialog (Font) of options.
3. Specify the desired options.
The properties you select are applied to the text object or text field.
4. When you're finished, click on the layout outside the text object to deselect the text object or text field.

**Partial Text Object or Default Text Field Text**

To set text properties for part of a text object or part of the default text in a text field:
1. Select the Text tool from the Tool palette.
2. Click on the text object or text field where you want to change properties.
The text cursor appears where you clicked.
3. Drag over the text you want to change.

4. Choose **Format**—>**Font, Spacing**, or **Justification** to show a submenu of options.

5. Specify the desired options.
   
The properties you select are applied to the text.

6. When you’re finished, click on the layout outside the text object or text field to exit text mode and deselect the text.

You can also create a text object using PL/SQL. Creating a text object using PL/SQL requires the following steps:

1. Create the text object.

2. Create and insert a compound text element into the main text object.
   
   A compound text element represents a line of text within the text object. The compound text element can contain one or more strings of text (simple text elements).
   
The first compound text element is inserted at the index point zero (0) of the text object unless otherwise specified.

3. Create one a simple text elements to be inserted in the previously created compound text object.
   
   A simple text element is an actual string of text that will be inserted in the line created by the compound text element.

4. Insert the simple text element into the compound text element.
   
   You insert simple text elements one at a time, each at a specified index point within the compound text element.

5. Repeat steps 3 and 4 until you have populated the text object with the desired text.
   
   You can set size, font, and spacing properties individually for each simple text element, or for the entire text object as a whole.

The following example creates a text object at point (1", 1") on the layout, and contains two compound text objects (lines of text). The first compound text object contains one simple text element(string), the second compound text object contains two simple text elements. These two lines of text will be in a 12–point Times font as follows:

This is line 1.

And now line 2.
Creating Text Objects Programmatically

PL/SQL

1. Create the text object.
2. Create and insert a compound text element into the main text object. A compound text element represents a line of text within the text object. The compound text element can contain one or more strings of text (simple text elements).
3. Create one a simple text elements to be inserted in the previously created compound text object. A simple text element is an actual string of text that will be inserted in the line created by the compound text element.
4. Insert the simple text element into the compound text element. You insert simple text elements one at a time, each at a specified index point within the compound text element.
5. Repeat steps 3 and 4 until you have populated the text object with the desired text.

You can set size, font, and spacing properties individually for each simple text element, or for the entire text object as a whole.

The following example creates a text object at point (1", 1") on the layout, and contains two compound text objects (lines of text). The first compound text object contains one simple text element (string), the second compound text object contains two simple text elements. These two lines of text will be in a 12–point Times font as follows:

This is line 1.

And now line 2.
PROCEDURE make_text IS
    text_obj   OG_OBJECT;
    text_pos   OG_INCH;
    smp_rec    OG_SMPTXT_ATTR;
BEGIN
    /* Set text object’s origin properties */
    text_pos.x:=OG_INCH;
    text_pos.y:=OG_INCH;

    /* Make the text object */
    text_obj:=OG_MAKE_TEXT(text_pos);

    /* Insert a new compound text element into the text object at index 0 */
    OG_INSERT_CMPTEXT(text_obj, 0);

    /* Set simple text record for desired text string*/
    smp_rec.str:='This is line 1.';
    smp_rec.mask:=OG_STR_SMPTXTA;

    /* Insert a new simple text element at index 0 in text object’s compound text element at index 0, using defined simple text record */
    OG_INSERT_SMPTXT(text_obj, smp_rec, 0, 0);

    /* Insert a new compound text element into the text object at index 1 */
    OG_INSERT_CMPTEXT(text_obj, 1);

    /* Change the simple text record’s text string */
    smp_rec.str:='And now';

    /* Insert a new simple text element at index 0 in text object’s compound text element at index 1, using defined simple text record */
    OG_INSERT_SMPTXT(text_obj, smp_rec, 1, 0);

    /* Change the simple text record’s text string */
    smp_rec.str:=' line 2.';

    /* Insert a new simple text element at index 1 in text object’s compound text element at index 1, using defined simple text record */
    OG_INSERT_SMPTXT(text_obj, smp_rec, 1, 1);
PROCEDURE make_text IS
  text_obj   OG_OBJECT;
  text_pos   OG_INCH;
  smp_rec    OG_SMPTEXT_ATTR;
BEGIN
  /* Set text object's origin properties */
  text_pos.x:=OG_INCH;
  text_pos.y:=OG_INCH;
  /* Make the text object */
  text_obj:=OG_MAKE_TEXT(text_pos);
  /* Insert a new compound text element into the text object at index 0 */
  OG_INSERT_CMPTEXT(text_obj, 0);
  /* Set simple text record for desired text string */
  smp_rec.str:='This is line 1.';
  smp_rec.mask:=OG_STR_SMPTEXTA;
  /* Insert a new simple text element at index 0 in text object's compound text element at index 0, using defined simple text record */
  OG_INSERT_SMPTEXT(text_obj, smp_rec, 0, 0);
  /* Insert a new compound text element into the text object at index 1 */
  OG_INSERT_CMPTEXT(text_obj, 1);
  /* Change the simple text record's text string */
  smp_rec.str:='And now';
  /* Insert a new simple text element at index 0 in text object's compound text element at index 1, using defined simple text record */
  OG_INSERT_SMPTEXT(text_obj, smp_rec, 1, 0);
  /* Change the simple text record's text string */
  smp_rec.str:=' line 2.';
  /* Insert a new simple text element at index 1 in text object's compound text element at index 1, using defined simple text record */
  OG_INSERT_SMPTEXT(text_obj, smp_rec, 1, 1);
Using Symbols

You can use symbols as decorative artwork elements on the layout, or you can specify symbols to be used as the plot type in a chart field template. You can specify size settings for symbols, but cannot specify colors or patterns for symbols.

Selecting a Symbol from the Symbol Palette

To select a symbol to use from the Symbol palette:

**Note:** You cannot change the available symbols.

1. Open the Symbol palette by clicking on its icon in the Tool palette. The palette opens to show the symbols available.
2. Select the symbol you want to use. The palette closes and the symbol appears in the Symbol palette icon. The symbol is applied to any currently selected symbol object and new symbol objects you place on the layout using the Symbol tool. The selection remains in effect for the current display until you change it again.

Placing Symbols in the Layout

To place a symbol object on the layout:

1. Select the Symbol tool from the Tool palette.
2. Click on the layout where you want the symbol to appear. The currently selected symbol in the Symbol palette appears on the layout.

Setting Symbol Size

To set the size of symbol objects:

1. Select the symbol object.
2. Choose **Format**—>**Symbol** to show a submenu of options.
You can use symbols as decorative artwork elements on the layout, or you can specify symbols to be used as the plot type in a chart field template. You can specify size settings for symbols, but cannot specify colors or patterns for symbols.

To select a symbol to use from the Symbol palette:

1. Open the Symbol palette by clicking on its icon in the Tool palette. The palette opens to show the symbols available.
2. Select the symbol you want to use. The palette closes and the symbol appears in the Symbol palette icon. The symbol is applied to any currently selected symbol object and new symbol objects you place on the layout using the Symbol tool. The selection remains in effect for the current display until you change it again.

To place a symbol object on the layout:

1. Select the Symbol tool from the Tool palette.
2. Click on the layout where you want the symbol to appear. The currently selected symbol in the Symbol palette appears on the layout.

To set the size of symbol objects:

1. Select the symbol object.
2. Choose Format —> Symbol to show a submenu of options.
3. Choose **Large, Medium, or Small**.

The size you select is applied to the currently–selected symbol. The setting remains in effect for the current display until you change it.

Creating Symbols Programmatically

You can create a symbol using the OG_MAKE_SYMBOL symbol property built-in. For example:

```plsql
PROCEDURE make_sym IS
    place  OG_POINT;
BEGIN
    place.x:=1*OG_INCH;
    place.y:=2*OG_INCH;
    OG_MAKE_SYMBOL(place, 5, OG_MEDIUM_SYMSIZE);
END;
```

For more information about setting an object’s size and position using OG_INCH, see “Specifying Coordinate and Size Information” on page 13–5.

Manipulating Objects

You can manipulate objects on the layout in the following ways:

- selecting objects
- moving objects
- copying and duplicating objects
- resizing objects
- reshaping objects
- rotating objects
- aligning objects
- grouping objects
- setting patterns and colors for objects
- setting line and edge properties
- undoing actions on objects
- renaming objects
- deleting objects
3. Choose Large, Medium, or Small. The size you select is applied to the currently–selected symbol. The setting remains in effect for the current display until you change it.

You can create a symbol using the OG_MAKE_SYMBOL symbol property built-in. For example:

```plsql
PROCEDURE make_sym IS
  place OG_POINT;
BEGIN
  place.x:=1*OG_INCH;
  place.y:=2*OG_INCH;
  OG_MAKE_SYMBOL(place, 5, OG_MEDIUM_SYMSIZE);
END;
```

For more information about setting an object's size and position using OG_INCH, see "Specifying Coordinate and Size Information" on page 13 – 5.

Manipulating Objects
You can manipulate objects on the layout in the following ways:
- selecting objects
- moving objects
- copying and duplicating objects
- resizing objects
- reshaping objects
- rotating objects
- aligning objects
- grouping objects
- setting patterns and colors for objects
- setting line and edge properties
- undoing actions on objects
- renaming objects
- deleting objects
Selecting Objects

To work with an object, you first need to select it. An object is selected when its control points, located at the four corners of its rectangular bounding box, are visible. When you deselect an object, its control points are no longer visible.

Depending on what action you wish to take on the selected objects, you can also select single or multiple objects by their node name in the Object Navigator. Selecting an object name node in the Object Navigator will also select the object in the Layout Editor.

Selecting objects in the Object Navigator can be useful if your objects are very small or close together and difficult to select in the Layout editor.

If you select an object that is a member of a group, the group object at the top of its group tree is selected. You can select multiple objects by shift-clicking on each one. When the Select tool is active, you can move objects by clicking on them and dragging them. You can resize objects by dragging their control points.

Scoping Rules

Graphics enforces certain scoping rules for selecting objects. These rules are enforced while any object is selected. The rules are:

1. The Select tool allows you to select only objects that belong to the same group as the currently selected object(s). To select an object in another group or one not in any group, do one of the following:
   - First deselect all objects by clicking on an unoccupied part of the layout, then select the other object as you normally would.
   - Choose one of the items in the Arrange—>Group Operations submenu to add an object to an existing group.
2. You can select multiple objects only if they belong to the same group (i.e., they are siblings).

Selecting Single or Group Objects

You can select objects by themselves or while other objects are selected.

To select a single or group object:

1. Select the Select tool from the Tool palette.
2. Click on the object to select it.
   - You can also drag a region around the object to select it.

To select a single or group object while other objects are selected:

1. Select the Select tool from the Tool palette.
2. Shift-click on the object to select it.
To work with an object, you first need to select it. An object is selected when its control points, located at the four corners of its rectangular bounding box, are visible. When you deselect an object, its control points are no longer visible.

Depending on what action you wish to take on the selected objects, you can also select single or multiple objects by their node name in the Object Navigator. Selecting an object name node in the Object Navigator will also select the object in the Layout Editor.

Selecting objects in the Object Navigator can be useful if your objects are very small or close together and difficult to select in the Layout editor. If you select an object that is a member of a group, the group object at the top of its group tree is selected. You can select multiple objects by shift-clicking on each one. When the Select tool is active, you can move objects by clicking on them and dragging them. You can resize objects by dragging their control points.

Scoping Rules

Graphics enforces certain scoping rules for selecting objects. These rules are enforced while any object is selected. The rules are:

1. The Select tool allows you to select only objects that belong to the same group as the currently selected object(s). To select an object in another group—or one not in any group—do one of the following:
   • First deselect all objects by clicking on an unoccupied part of the layout, then select the other object as you normally would.
   • Choose one of the items in the Arrange —> Group Operations submenu to add an object to an existing group.

2. You can select multiple objects only if they belong to the same group (i.e., they are siblings).

You can select objects by themselves or while other objects are selected. To select a single or group object:

1. Select the Select tool from the Tool palette.
2. Click on the object to select it.

You can also drag a region around the object to select it.

To select a single or group object while other objects are selected:

1. Select the Select tool from the Tool palette.
2. Shift-click on the object to select it.
Selecting Multiple Objects

To select more than one object at a time:
1. Select the Select tool from the Tool palette.
2. Drag a region around the objects to select them or select one object at a time using shift-click after you’ve selected the first object.

Selecting Single Objects within Group Objects

When you use the Select tool to click on a single object that is part of a group, you’ll select the highest level group to which that object belongs.

To select only the single object within a group object:
1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   
   You can also drag a region around the group object to select it.
3. Click on the child object to select it.

To select a single object within a group while other objects within the same group are selected:
1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   
   You can also hold down the Shift key while you drag a region around the group object to select it.
3. Click on a child object to select it.
4. Shift-click on a second child object in the group to select it.
5. Repeat Step 4 for each child object you want to select.

   Note: You may select multiple objects simultaneously only if they share the same parent.

Selecting All Objects

To select all objects on the layout:

Choose Edit—>Select All.

All objects on the active layer are selected.

Deselecting Objects

If one or more objects are selected, you can deselect them in the following ways:

- To deselect all selected objects, click on a blank area of the layout.
You can also hold down the Shift key while you drag a region around the object to select it.

3. Repeat this process for each object you want to select.

To select more than one object at a time:

1. Select the Select tool from the Tool palette.
2. Drag a region around the objects to select them or select one object at a time using shift-click after you've selected the first object.

When you use the Select tool to click on a single object that is part of a group, you'll select the highest level group to which that object belongs.

To select only the single object within a group object:

1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   You can also drag a region around the group object to select it.
3. Click on the child object to select it.

To select a single object within a group while other objects within the same group are selected:

1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   You can also hold down the Shift key while you drag a region around the group object to select it.
3. Click on a child object to select it.
4. Shift-click on a second child object in the group to select it.
5. Repeat Step 4 for each child object you want to select.

Note: You may select multiple objects simultaneously only if they share the same parent.

To select all objects on the layout:

Choose Edit —> Select All.
All objects on the active layer are selected.

If one or more objects are selected, you can deselect them in the following ways:

• To deselect all selected objects, click on a blank area of the layout.
Moving Objects

You can move objects and change their stacking order on the layout.

Moving a Single or Group Object

To move a single or group object on the layout:

1. Select the object you want to move.
2. Click inside the bounding box of the object and drag it to a new location.

Suggestion: To maintain an object’s position with respect to the horizontal or vertical axis, hold down the Shift key while dragging it horizontally or vertically.

You can also move an object one pixel at a time in any direction using the arrow keys.

Moving Objects Programmatically

You can also move objects using PL/SQL. The following example moves the specified object to another position on the layout:

```plsql
PROCEDURE move_inventory(invent_obj IN og_object) IS
    distance   OG_POINT;
BEGIN
    distance.x:=(3*OG_INCH);
    distance.y:=(3*OG_INCH);
    OG_MOVE(invent_obj, distance);
END;
```


For more information about setting an object’s size and position using OG_INCH, see “Specifying Coordinate and Size Information” on page 13 – 5.

Moving Multiple Objects

To move more than one object at a time:

1. Select the objects you want to move.
2. Click inside the bounding box of a single selected object and drag it to a new location. All other selected objects will move with it.

• To deselect one of multiple selected objects, shift-click on the object(s).
You can move objects and change their stacking order on the layout.

To move a single or group object on the layout:

1. Select the object you want to move.
2. Click inside the bounding box of the object and drag it to a new location.

Suggestion: To maintain an object's position with respect to the horizontal or vertical axis, hold down the Shift key while dragging it horizontally or vertically.

You can also move an object one pixel at a time in any direction using the arrow keys.

You can also move objects using PL/SQL. The following example moves the specified object to another position on the layout:

```
PROCEDURE move_inventory(invent_obj IN og_object) IS
    distance   OG_POINT;
BEGIN
    distance.x:=(3*OG_INCH);
    distance.y:=(3*OG_INCH);
    OG_MOVE(invent_obj, distance);
END;
```


For more information about setting an object's size and position using OG_INCH, see "Specifying Coordinate and Size Information" on page 13 – 5.

To move more than one object at a time:

1. Select the objects you want to move.
2. Click inside the bounding box of a single selected object and drag it to a new location. All other selected objects will move with it.
Changing Stack Order of Objects

To change the order in which objects are stacked on a layer:
1. Select the objects(s) you want to move.
2. Choose the Arrange menu and then do one of the following:
   - To move the object(s) in front of the object directly above it, choose Move Forward.
   - To move the object(s) in back of the object directly behind it, choose Move Backward.
   - To move the object(s) in front of all other objects, choose Bring to Front.
   - To move the object(s) behind all other objects, choose Send to Back.

Copying and Duplicating Objects

You can copy or duplicate objects or text blocks in a display. To copy objects to the Clipboard so that you can paste them elsewhere (such as on another layer), use the Copy option. To copy objects directly on the layout, use the Duplicate option.

Copying Objects

You can copy to and paste from the Clipboard.

To copy objects to the Clipboard:
1. Select the object(s) you want to copy.
2. Choose the Copy button or Edit—>Copy, or choose the Copy command from the pop-up menu.

Graphics stores the object(s) on the Clipboard.

To paste objects from the Clipboard:
1. Choose the Paste button or Edit—>Paste.

Graphics pastes the object(s) from the Clipboard to the layout in the same position as the original(s). The object is selected and you can move it.

Note: If you paste directly over the original object, you won’t see the original until you move the copy.

Duplicating Objects

To duplicate objects on the layout:
1. Select the object(s) you want to duplicate.
2. Choose Edit—>Duplicate.

Graphics duplicates the object(s) on the layout, slightly offset from the position of the original.
Changing Stack Order of Objects

Copying and Duplicating Objects

Copying Objects

Duplicating Objects

4 – 19 Working with the Layout Editor

To change the order in which objects are stacked on a layer:

1. Select the objects(s) you want to move.
2. Choose the Arrange menu and then do one of the following:
   - To move the object(s) in front of the object directly above it, choose Move Forward.
   - To move the object(s) in back of the object directly behind it, choose Move Backward.
   - To move the object(s) in front of all other objects, choose Bring to Front.
   - To move the object(s) behind all other objects, choose Send to Back.

You can copy or duplicate objects or text blocks in a display. To copy objects to the Clipboard so that you can paste them elsewhere (such as on another layer), use the Copy option. To copy objects directly on the layout, use the Duplicate option.

You can copy to and paste from the Clipboard.

To copy objects to the Clipboard:

1. Select the object(s) you want to copy.
2. Choose the Copy button or Edit —> Copy, or choose the Copy command from the pop-up menu. Graphics stores the object(s) on the Clipboard.

To paste objects from the Clipboard:

Choose the Paste button or Edit —> Paste. Graphics pastes the object(s) from the Clipboard to the layout in the same position as the original(s). The object is selected and you can move it.

Note: If you paste directly over the original object, you won’t see the original until you move the copy.

To duplicate objects on the layout:

1. Select the object(s) you want to duplicate.
2. Choose Edit —> Duplicate. Graphics duplicates the object(s) on the layout, slightly offset from the position of the original.
You can perform multiple duplications of the originally selected object by continuing to choose Edit—>Duplicate.

Duplicating Objects Programmatically

You can also duplicate an object using PL/SQL. The following example creates a new object that is identical to the specified object:

```plsql
PROCEDURE dup_object(my_old_object IN og_object) IS
  my_new_object   OG_OBJECT;
BEGIN
  my_new_object:=OG_CLONE(my_old_object);
END;
```


Resizing Objects

You can resize objects by dragging one of their four control points to a new position. The way in which you drag the control point determines how the object will change size.

**Suggestion:** To maintain an object’s aspect ratio, hold down the Shift key while dragging the control point to a new position.

Resizing Single or Group Objects

To resize a single or group object:
1. Select the object you want to resize.
2. Click on a control point and drag it to a new position.

Resizing Objects Programmatically

You can also resize objects using PL/SQL. The following example of a button procedure doubles the size of the specified object:

```plsql
PROCEDURE double(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
  anchor OG_POINT;
  newpt  OG_POINT;
  oldpt  OG_POINT;
BEGIN
  anchor.x:=OG_INCH;
  anchor.y:=OG_INCH;
  oldpt.x:=OG_INCH+1;
  oldpt.y:=OG_INCH+1;
  newpt.x:=OG_INCH+2;
  newpt.y:=OG_INCH+2;
```
You can perform multiple duplications of the originally selected object by continuing to choose Edit—>Duplicate.

You can also duplicate an object using PL/SQL. The following example creates a new object that is identical to the specified object:

```
PROCEDURE dup_object(my_old_object IN og_object) IS
  my_new_object   OG_OBJECT;
BEGIN
  my_new_object:=OG_CLONE(my_old_object);
END;
```


You can resize objects by dragging one of their four control points to a new position. The way in which you drag the control point determines how the object will change size.

Suggestion: To maintain an object's aspect ratio, hold down the Shift key while dragging the control point to a new position.

To resize a single or group object:

1. Select the object you want to resize.
2. Click on a control point and drag it to a new position.

You can also resize objects using PL/SQL. The following example of a button procedure doubles the size of the specified object:

```
PROCEDURE double(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
  anchor OG_POINT;
  newpt  OG_POINT;
  oldpt  OG_POINT;
BEGIN
  anchor.x:=OG_INCH;
  anchor.y:=OG_INCH;
  oldpt.x:=OG_INCH+1;
  oldpt.y:=OG_INCH+1;
  newpt.x:=OG_INCH+2;
  newpt.y:=OG_INCH+2;
```


For more information about setting an object’s size and position using OG_INCH, see “Specifying Coordinate and Size Information” on page 13 – 5.

To resize multiple objects:
1. Select the objects you want to resize.
2. Click on the control point of a single selected object and drag it to a new position.
   
   All other selected objects will change size relative to that object.

To make multiple objects the same size:
1. Select the objects you want to resize.
2. Choose Arrange—>Size Objects to show the Size Objects dialog.
3. Specify the desired options such as whether to size according to the larger or smaller object, or use an average, for width, height, or both, and what unit of measure to use.
4. Choose the OK button to resize the objects according to your settings and close the Size Objects dialog.

You can change the shape of objects by dragging their vertices to a new position. Note that the control points you see when reshaping an object differ from those you see when selecting an object. The Reshape tool produces different results for different objects:

<table>
<thead>
<tr>
<th>Object</th>
<th>Reshape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arc</td>
<td>change the sweep angle to something other than 90 degrees (the default)</td>
</tr>
<tr>
<td>polygons and polylines</td>
<td>move any vertex to a new position</td>
</tr>
<tr>
<td>lines</td>
<td>move the start point or end point to a new position</td>
</tr>
<tr>
<td>ellipses , rectangles,</td>
<td>move any vertex to a new position</td>
</tr>
<tr>
<td>and rounded rectangles</td>
<td></td>
</tr>
<tr>
<td>freehand objects</td>
<td>move any point to a new position</td>
</tr>
<tr>
<td>images</td>
<td>clip (or crop) the image</td>
</tr>
</tbody>
</table>
To resize multiple objects:
1. Select the objects you want to resize.
2. Click on the control point of a single selected object and drag it to a new position.

To make multiple objects the same size:
1. Select the objects you want to resize.
2. Choose **Arrange** —> **Size Objects** to show the Size Objects dialog.
3. Specify the desired options such as whether to size according to the larger or smaller object, or use an average, for width, height, or both, and what unit of measure to use.
4. Choose the OK button to resize the objects according to your settings and close the Size Objects dialog.

You can change the shape of objects by dragging their vertices to a new position. Note that the control points you see when reshaping an object differ from those you see when selecting an object. The Reshape tool produces different results for different objects:

- **arc** change the sweep angle to something other than 90 degrees (the default)
- **polygons and polylines** move any vertex to a new position
- **lines** move the start point or end point to a new position
- **ellipses, rectangles, and rounded rectangles** move any vertex to a new position
- **freehand objects** move any point to a new position
- **images** clip (or crop) the image
Note: Reshaping ellipses, rectangles, and rounded rectangles differs from using the Select tool only if the object has been rotated such that the select control points no longer coincide with the reshape control points.

To reshape an object:
1. Select the object you want to reshape.
2. Select the Reshape tool from the Tool palette.
   Depending on the type of object, the control points move to the vertices of the object.
3. Click on one of the vertices and drag it to a new position.
   The Reshape tool produces different results for different types of objects.

Rotating Objects
You can rotate single or multiple objects in any direction using the Rotate tool. Objects are rotated on their centerpoint.

Some systems do not support rotating text. For more information, see the Developer/2000 Installation Guide.

Rotating Single or Multiple Objects
To rotate single or multiple objects:
1. Select the Rotate tool from the Tool palette.
2. Click on a control point of the object(s) and drag it in the direction you want the object(s) to rotate.

Rotating Objects within Groups
To rotate one or more child objects within a group:
1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   You can also drag a region around the group object to select it.
3. Click on the child object to select it.
   Use Shift-click to select multiple child objects in the group.
4. Select the Rotate tool from the Tool palette.
5. Click on a control point of the object(s) and drag it in the direction you want the object(s) to rotate.
Note: Reshaping ellipses, rectangles, and rounded rectangles differs from using the Select tool only if the object has been rotated such that the select control points no longer coincide with the reshape control points.

To reshape an object:
1. Select the object you want to reshape.
2. Select the Reshape tool from the Tool palette.
   Depending on the type of object, the control points move to the vertices of the object.
3. Click on one of the vertices and drag it to a new position.
   The Reshape tool produces different results for different types of objects.

You can rotate single or multiple objects in any direction using the Rotate tool. Objects are rotated on their centerpoint.

Some systems do not support rotating text. For more information, see the Developer/2000 Installation Guide.

To rotate single or multiple objects:
1. Select the Rotate tool from the Tool palette.
2. Click on a control point of the object(s) and drag it in the direction you want the object(s) to rotate.

To rotate one or more child objects within a group:
1. Select the Select tool from the Tool palette.
2. Click on the group object to select it.
   You can also drag a region around the group object to select it.
3. Click on the child object to select it.
   Use Shift-click to select multiple child objects in the group.
4. Select the Rotate tool from the Tool palette.
5. Click on a control point of the object(s) and drag it in the direction you want the object(s) to rotate.
You can also rotate objects using PL/SQL. The following example rotates the specified object by the specified number of degrees counterclockwise:

```plsql
PROCEDURE rotate_needle (degrees IN NUMBER, center_pt IN og_point) IS
the_needle OG_OBJECT;
BEGIN
  the_needle := og_get_object('Needle 1');
  OG_ROTATE(the_needle, center_pt, degrees);
END;
```


Some systems do not support rotating text. For more information, see the Developer/2000 Installation Guide.

You can move objects so that their top, side, or bottom edges, or centers align with each other.

Once you specify alignment settings, those settings remain active until you change them. You can define your alignment settings once, and repeat them as needed.

To align objects and set alignment settings:

1. Select the objects you wish to align.
2. Choose Arrange—>Align Objects to show the Alignment Settings dialog.
3. Specify the desired alignment settings such as whether to align the objects along their top or bottom edges, left or right sides, or centers. You can also specify that objects be aligned to each other, or to the grid.
4. Choose the OK button to confirm the alignment settings and align the objects.

To align objects using the current alignment settings:

1. Select the objects you want to align.
2. Choose Arrange—>Repeat Alignment.
   The objects are aligned according to the current settings in the Alignment Settings dialog box.
You can also rotate objects using PL/SQL. The following example rotates the specified object by the specified number of degrees counterclockwise:

```plsql
PROCEDURE rotate_needle (degrees IN NUMBER, center_pt IN og_point) IS
    the_needle     OG_OBJECT;
BEGIN   the_needle:=og_get_object('Needle 1');
OG_ROTATE(the_needle, center_pt, degrees);
END;
```


Some systems do not support rotating text. For more information, see the Developer/2000 Installation Guide.

You can move objects so that their top, side, or bottom edges, or centers align with each other.

Once you specify alignment settings, those settings remain active until you change them. You can define your alignment settings once, and repeat them as needed.

To align objects and set alignment settings:

1. Select the objects you wish to align.
2. Choose Arrange —> Align Objects to show the Alignment Settings dialog.
3. Specify the desired alignment settings such as whether to align the objects along their top or bottom edges, left or right sides, or centers. You can also specify that objects be aligned to each other, or to the grid.
4. Choose the OK button to confirm the alignment settings and align the objects.

To align objects using the current alignment settings:

1. Select the objects you want to align.
2. Choose Arrange —> Repeat Alignment.
   The objects are aligned according to the current settings in the Alignment Settings dialog box.
Grouping Objects
You can group multiple objects together so that you can manipulate them as one object. The objects that belong to a group are the child objects, and the group is the parent. You can also group more than one group object together. The resulting group will be made up of subgroups.

To group multiple objects together as one object:
1. Select the objects you want to group together.
2. Choose Arrange—>Group.
   The objects are grouped together.

Ungrouping Objects
To ungroup objects within a group:
1. Select the group object.
2. Choose Arrange—>Ungroup.
   The grouped objects are separated.

Selecting the Parent Group
To select the parent group of the currently selected child object:
   ■ Choose Arrange—>Group Operations—>Select Parent.
   The parent group is selected.

Selecting the Child Object
To select the child objects of the currently selected group object:
   ■ Choose Arrange—>Group Operations—>Select Children.
   The child objects of the group object are selected.

Adding and Removing Child Objects
To add or remove child objects from their parent objects:
1. Select both the group object and the object you want to add, or select the child object you want to remove.
2. Choose Arrange—>Group Operations, and then do one of the following:
   - To add an object to the group object, select Add to Group.
   - To remove a child object from the group object, select Remove from Group.
   You can also add and remove child objects using PL/SQL. The following example removes a child object from one group object and inserts it into another:
You can group multiple objects together so that you can manipulate them as one object. The objects that belong to a group are the child objects, and the group is the parent. You can also group more than one group object together. The resulting group will be made up of subgroups.

To group multiple objects together as one object:
1. Select the objects you want to group together.
2. Choose **Arrange** —> **Group**.
   The objects are grouped together.

To ungroup objects within a group:
1. Select the group object.
2. Choose **Arrange** —> **Ungroup**.
   The grouped objects are separated.

To select the parent group of the currently selected child object:
- Choose **Arrange** —> **Group Operations** —> **Select Parent**.
  The parent group is selected.

To select the child objects of the currently selected group object:
- Choose **Arrange** —> **Group Operations** —> **Select Children**.
  The child objects of the group object are selected.

To add or remove child objects from their parent objects:
1. Select both the group object and the object you want to add, or select the child object you want to remove.
2. Choose **Arrange** —> **Group Operations**, and then do one of the following:
   - To add an object to the group object, select Add to Group.
   - To remove a child object from the group object, select Remove from Group.

You can also add and remove child objects using PL/SQL. The following example removes a child object from one group object and inserts it into another:
PROCEDURE move_prod(warehouse1 IN og_object, warehouse2 IN og_object, prod_index IN NUMBER) IS
oldpt.x:=OG_INCH+1;
oldpt.y:=OG_INCH+1;
newpt.x:=OG_INCH+2;
newpt.y:=OG_INCH+2;
OG_SCALE(hitobj, anchor, oldpt, newpt);
the_prod  og_object;
BEGIN
   the_prod:=OG_GET_CHILD(warehouse1, prod_index);
   OG_DELETE_CHILD(warehouse1, prod_index, 1);
   OG_INSERT_CHILD(warehouse2, the_prod, OG_LAST);
END;


For more information about setting an object’s size and position using OG_INCH, see “Specifying Coordinate and Size Information” on page 13 – 5.

### Setting Patterns and Colors for Objects

You can set patterns and, if your system supports color, you can set colors for objects on the layout.

### Line/Fill/Text Display

The Line/Fill/Text display shows the edge, fill, and text pattern and colors that are currently selected. Each object you create will use those settings unless you specify different ones.

### Setting Patterns

Different patterns can be applied to the fill of objects. For text objects, you can apply patterns to the fill of the bounding box, but not the characters themselves. For charts, you can apply patterns in the chart template.

To set the fill pattern:

1. Select any existing object(s) to which you want the pattern applied.
2. Select the Fill Color palette.
3. Choose the Patterns command from the Fill Color palette menu to show the Pattern palette.
4. Click on the pattern you wish to apply to the object.
You can set patterns and, if your system supports color, you can set colors for objects on the layout.

The Line/Fill/Text display shows the edge, fill, and text pattern and colors that are currently selected. Each object you create will use those settings unless you specify different ones.

Different patterns can be applied to the fill of objects. For text objects, you can apply patterns to the fill of the bounding box, but not the characters themselves. For charts, you can apply patterns in the chart template.

To set the fill pattern:

1. Select any existing object(s) to which you want the pattern applied.
2. Select the Fill Color palette.
3. Choose the Patterns command from the Fill Color palette menu to show the Pattern palette.
4. Click on the pattern you wish to apply to the object.

For more information about Graphics PL/SQL Built-ins, see the "Built-in Subprograms" and "Property Built-ins" chapters in the Graphics Reference Manual. For more information about setting an object's size and position using OG_INCH, see "Specifying Coordinate and Size Information" on page 13-5.
The pattern is applied to any selected objects and new objects you draw on the layout. The pattern also appears in the fill part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

5. Close the Pattern palette.

Setting Pattern Colors

The dark part of a pattern is called the foreground element, and the light part of a pattern is called the background element. If your system supports color, you can apply different colors to the foreground and background elements of the current fill pattern.

To set the foreground or background color for a pattern:

1. Select any existing object(s) or chart elements to which you want the foreground and background pattern colors applied.

   If you do not select an object, the settings will not be applied to any existing objects, but will remain in affect for any new objects you create until you change them.

2. Select the Fill Color palette.

3. Choose the Patterns command from the Fill Color palette menu.

   The Pattern palette appears showing the current settings for foreground and background colors. You can move this palette around the layout to improve visibility.

4. Select the drop-down color palette for either the Foreground color (left side) or the Background color (right side).

   The appropriate palette opens to show the colors available on your system.

5. Select the color you want to apply to the foreground or background of the pattern.

   The pattern foreground/background color palette closes and the changes are reflected in the Pattern palette. In addition, the color is applied to any selected objects, and in the fill part of the Line/Fill/Text display box.

6. Close the Pattern palette or select another color.

   The selection remains in effect for the current display until you change it again.

Setting Colors for Objects

If your system supports color, you can apply colors to both the inside (fill) and border (line) of an object.
The pattern is applied to any selected objects and new objects you draw on the layout. The pattern also appears in the fill part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

5. Close the Pattern palette.

The dark part of a pattern is called the foreground element, and the light part of a pattern is called the background element. If your system supports color, you can apply different colors to the foreground and background elements of the current fill pattern.

To set the foreground or background color for a pattern:

1. Select any existing object(s) or chart elements to which you want the foreground and background pattern colors applied. If you do not select an object, the settings will not be applied to any existing objects, but will remain in effect for any new objects you create until you change them.

2. Select the Fill Color palette.

3. Choose the Patterns command from the Fill Color palette menu. The Pattern palette appears showing the current settings for foreground and background colors. You can move this palette around the layout to improve visibility.

4. Select the drop-down color palette for either the Foreground color (left side) or the Background color (right side). The appropriate palette opens to show the colors available on your system.

5. Select the color you want to apply to the foreground or background of the pattern. The pattern foreground/background color palette closes and the changes are reflected in the Pattern palette. In addition, the color is applied to any selected objects, and in the fill part of the Line/Fill/Text display box.

6. Close the Pattern palette or select another color. The selection remains in effect for the current display until you change it again.

If your system supports color, you can apply colors to both the inside (fill) and border (line) of an object.
1. Select any existing object(s) or chart elements to which you want to apply color.

2. Select the Fill Color palette to select a background color for the object, or select the Line Color palette to select a foreground color for the border of the object.

   The appropriate color palette appears.

   You can “tear off” the color palette and move it to another location within the Layout Editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select a color from the palette.

   The palette closes and the color, if any, is applied to any selected objects and any new objects you draw on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

Removing Fill Patterns and Colors from Objects

The pattern and colors you select for an object remain active until you reset them. To remove a pattern or solid background color (fill) from an object:

1. Select any existing object(s) or chart elements for which you want to reset or remove color or patterns.

2. Select the Fill Color palette.

   You can “tear off” the color palette and move it to another location within the Layout editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select the No Fill option to remove the current color or pattern selection.

   The palette closes and the fill color is reset for any selected objects and any new objects you draw on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

Removing Object Borders

By default, a black border line is created for all objects you draw in the Layout editor. You can hide any object’s border (foreground color) using the Line Color palette.

1. Select any existing object(s) or chart elements you want to remove the border of.

2. Select the Line Color palette.
Removing Fill Patterns and Colors from Objects

Removing Object Borders

1. Select any existing object(s) or chart elements to which you want to apply color.
2. Select the Fill Color palette to select a background color for the object, or select the Line Color palette to select a foreground color for the border of the object. The appropriate color palette appears.
3. Select a color from the palette. The palette closes and the color, if any, is applied to any selected objects and any new objects you draw on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

The pattern and colors you select for an object remain active until you reset them. To remove a pattern or solid background color (fill) from an object:
1. Select any existing object(s) or chart elements for which you want to reset or remove color or patterns.
2. Select the Fill Color palette.
3. Select the No Fill option to remove the current color or pattern selection. The palette closes and the fill color is reset for any selected objects and any new objects you draw on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

By default, a black border line is created for all objects you draw in the Layout editor. You can hide any object's border (foreground color) using the Line Color palette.
1. Select any existing object(s) or chart elements you want to remove the border of.
2. Select the Line Color palette.
You can “tear off” the color palette and move it to another location within the Layout editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select the No Line option to hide the border.

The palette closes and the line color is reset for any selected objects and new objects you draw on the layout. The color, or lack thereof also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

Setting Colors Programatically

You can also change the color of an object using PL/SQL. The following example uses the OG_SET_FFCOLOR graphic property built-in to set the foreground fill property of the specified object:

```
PROCEDURE color_country(country_name) IS
  my_object    OG_OBJECT;
BEGIN
  my_object:=OG_GET_OBJECT(country_name);
  OG_SET_FFCOLOR(country_name, 'color 72');
END;
```


Editing Color Palettes

If your system supports color, you can edit the current color palette for your display. You can also import and export color palettes.

**Editing Color Palettes**

To edit the current color palette:

1. Choose Format—>Color Palette to show the Color Palette dialog.

   **Suggestion:** If this menu item is not available, make sure Color Mode is set to Editable in Tools Options–Defaults property sheet. If not, you’ll have to close the display and open it again for the new Color Mode setting to take effect.

2. Select a color to edit.

3. Choose the Edit button.

   Your system’s color editing dialog box appears.

4. Edit the color by following the procedure for your system.

5. Accept the dialog box to return to the Color Palette dialog box.
You can "tear off" the color palette and move it to another location within the Layout editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select the No Line option to hide the border. The palette closes and the line color is reset for any selected objects and new objects you draw on the layout. The color, or lack thereof, also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

You can also change the color of an object using PL/SQL. The following example uses the OG_SET_FFCOLOR graphic property built-in to set the foreground fill property of the specified object:

```plsql
PROCEDURE color_country(country_name) IS
my_object    OG_OBJECT;
BEGIN
my_object:=OG_GET_OBJECT(country_name);
OG_SET_FFCOLOR(country_name, 'color 72');
END;
```


If your system supports color, you can edit the current color palette for your display. You can also import and export color palettes.

**Editing Color Palettes**

To edit the current color palette:

1. Choose Format —> Color Palette to show the Color Palette dialog.

Suggestion: If this menu item is not available, make sure Color Mode is set to Editable in Tools Options–Defaults property sheet. If not, you'll have to close the display and open it again for the new Color Mode setting to take effect.

2. Select a color to edit.

3. Choose the Edit button. Your system's color editing dialog box appears.

4. Edit the color by following the procedure for your system.

5. Accept the dialog box to return to the Color Palette dialog box.
6. To rename the color, type a different name in the Current Color field and choose the Rename button.

7. Choose the OK button to close the Color Palette dialog.

Managing Color Palettes The procedures for renaming, granting access to, and deleting color palettes are the same as those for displays. Likewise, the procedures for importing and exporting color palettes are the same as those for artwork.

For more information about administering displays, see the “Managing Displays” chapter on page 6 – 1.

For more information about importing and exporting artwork, see “Importing and Exporting Artwork” on page 6 – 4.

Setting Colors for Text

In addition to setting colors and patterns for a text object’s bounding box, you can select colors for the text itself. For text within charts, you can apply colors in the chart template, or by selecting the chart element directly in the Layout Editor.

1. Select the text object or chart element to which you wish to apply a color.

2. Select the Text Color palette icon in the Layout Editor to show the Text Color palette.

   You can “tear off” the color palette and move it to another location within the Layout Editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select a color for the text.

   The palette closes and the color, if any, is applied to any selected text objects and new text objects you create on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

If you want to link a chart element color to your data source, you can apply a color to chart elements using a format trigger. For more information about format triggers, see “Using Format Triggers” in chapter “Using Triggers” on page 11 – 8.

Setting Line and Edge Properties

You can set properties for lines and for the edges of objects, including weight and bevelling.
To rename the color, type a different name in the Current Color field and choose the Rename button.

Choose the OK button to close the Color Palette dialog.

Managing Color Palettes

The procedures for renaming, granting access to, and deleting color palettes are the same as those for displays. Likewise, the procedures for importing and exporting color palettes are the same as those for artwork.

For more information about administering displays, see the "Managing Displays" chapter on page 6–1.

For more information about importing and exporting artwork, see "Importing and Exporting Artwork" on page 6–4.

In addition to setting colors and patterns for a text object's bounding box, you can select colors for the text itself. For text within charts, you can apply colors in the chart template, or by selecting the chart element directly in the Layout Editor.

1. Select the text object or chart element to which you wish to apply a color.

2. Select the Text Color palette icon in the Layout Editor to show the Text Color palette.

You can "tear off" the color palette and move it to another location within the Layout Editor by selecting the Tear Off Palette option and then dragging the title bar of the palette to move it.

3. Select a color for the text.

The palette closes and the color, if any, is applied to any selected text objects and new text objects you create on the layout. The color also appears in the appropriate part of the Line/Fill/Text display box and remains in effect for the current display until you change it again.

If you want to link a chart element color to your data source, you can apply a color to chart elements using a format trigger. For more information about format triggers, see "Using Format Triggers" in chapter "Using Triggers" on page 11–8.

You can set properties for lines and for the edges of objects, including weight and bevelling.
Setting Line Properties

To set line properties:

1. Select the line(s).
2. Choose one of the following commands from the **Format** menu:
   - **Line**
   - **Dash**
   - **Arrow**
   - Shows a submenu of line width sizes.
   - Shows a submenu of line dash styles.
   - Show a submenu of arrow styles.
3. Specify the desired options.
   - The properties you select are applied to the line object. Settings remain in effect for the current display until you change them again.

Setting Edge Properties

To set edge properties for a graphic object such as a rectangle:

1. Select the object(s).
2. Choose **Format** —> **Line** to set edge width, or **Format** —> **Bevel** to show a list of bevelling options.
   - **Note**: If you wish to set a bevelling option for the object’s edge, you should set a line width of at least 2 pt.
3. Specify the desired options.
   - The properties you select are applied to the edges of the selected object(s). Settings remain in effect for the current display until you change them again.

Undoing Actions on Objects

To reverse the most recent operation you performed:

- From the Layout editor, choose **Edit** —> **Undo**.
  - The operation is reversed. For example, if you deleted an object, the object is retrieved.

Renaming Objects

Once you have created an object, you can name it using the Object property sheet. Alternatively, you can name an object directly in the Object Navigator by selecting the object name and editing it accordingly.

If you plan on using your objects in any PL/SQL program units, you should assign each object a unique name so you can reference it easily.

- **Note**: If you change the names of any objects referenced in PL/SQL program units, you must change those references to reflect the new name.

To name a single object or group of objects:
Setting Line Properties

To set line properties:
1. Select the line(s).
2. Choose one of the following commands from the Format menu:
   - Shows a submenu of line width sizes.
   - Shows a submenu of line dash styles.
   - Show a submenu of arrow styles.
3. Specify the desired options.

The properties you select are applied to the line object. Settings remain in effect for the current display until you change them again.

Setting Edge Properties

To set edge properties for a graphic object such as a rectangle:
1. Select the object(s).
2. Choose Format —> Line to set edge width, or Format —> Bevel to show a list of bevelling options.
   - Note: If you wish to set a bevelling option for the object's edge, you should set a line width of at least 2 pt.
3. Specify the desired options.

The properties you select are applied to the edges of the selected object(s). Settings remain in effect for the current display until you change them again.

Undoing Actions on Objects

To reverse the most recent operation you performed:
- From the Layout editor, choose Edit —> Undo. The operation is reversed. For example, if you deleted an object, the object is retrieved.

Renaming Objects

Once you have created an object, you can name it using the Object property sheet. Alternatively, you can name an object directly in the Object Navigator by selecting the object name and editing it accordingly.

If you plan on using your objects in any PL/SQL program units, you should assign each object a unique name so you can reference it easily.

Note: If you change the names of any objects referenced in PL/SQL program units, you must change those references to reflect the new name.

To name a single object or group of objects:
- Line
- Dash
- Arrow
1. Double-click on the object or object group to show the Object property sheet.

   Alternatively, you can select the object or object group and choose the Properties command from the pop-up menu, or select the object and choose Tools—>Properties.

2. Type a name in the Name field.

3. Choose the OK button to accept the new name and close the Object property sheet.

Deleting Objects

You can delete objects by cutting or clearing them from the layout. Cutting deletes objects and stores them on the Clipboard so you can paste them somewhere else. Clearing deletes the objects without affecting the contents of the Clipboard.

In addition, you can delete objects from the Object Navigator.

Cutting Objects

To cut objects to the clipboard:

1. Select the object(s) you want to delete.

2. Choose the Cut command from the pop-up menu, or choose the Cut button or Edit—>Cut.

   Graphics deletes the object(s) from the layout and stores the object(s) on the Clipboard.

Clearing Objects

Cleared objects cannot be retrieved. To clear objects:

1. Select the object(s) you want to delete.

2. Choose the Clear command from the pop-up menu, or choose Edit—>Clear.

   Graphics deletes the object(s) from the layout without affecting the contents of the Clipboard.

Deleting Objects Programmaticaly

You can also delete an object using PL/SQL. The following example deletes ("destroys") the specified window from the runtime display:

```pl-sql
PROCEDURE destroy_USA IS
    the_window   OG_WINDOW;
BEGIN
    the_window:=OG_GET_WINDOW('USA_window');
    OG_DESTROY(the_window);
END;
```

Work with the Layout Editor 4 – 31
Deleting Objects

Cutting Objects

Clearing Objects

Deleting Objects

Programmmatically

PL/SQL

4 – 31

Working with the Layout Editor

1. Double-click on the object or object group to show the Object property sheet.
2. Alternatively, you can select the object or object group and choose the Properties command from the pop-up menu, or select the object and choose Tools —> Properties.
3. Type a name in the Name field.
4. Choose the OK button to accept the new name and close the Object property sheet.

You can delete objects by cutting or clearing them from the layout.

Cutting deletes objects and stores them on the Clipboard so you can paste them somewhere else. Clearing deletes the objects without affecting the contents of the Clipboard.

In addition, you can delete objects from the Object Navigator.

To cut objects to the clipboard:

1. Select the object(s) you want to delete.
2. Choose the Cut command from the pop-up menu, or choose the Cut button or Edit —> Cut.

Graphics deletes the object(s) from the layout and stores the object(s) on the Clipboard.

Cleared objects cannot be retrieved.

To clear objects:

1. Select the object(s) you want to delete.
2. Choose the Clear command from the pop-up menu, or choose Edit —> Clear.

Graphics deletes the object(s) from the layout without affecting the contents of the Clipboard.

You can also delete an object using PL/SQL. The following example deletes (“destroys”) the specified window from the runtime display:

```
PROCEDURE destroy_USA IS
  the_window   OG_WINDOW;
BEGIN
  the_window:=OG_GET_WINDOW('USA_window');
  OG_DESTROY(the_window);
END;
```

Customizing Your Work Environment

While you’re working in the Designer, you can perform the following actions to customize features of your Layout Editor work environment:

- setting preferences
- specifying layout settings
- increasing and decreasing magnification
- hiding and showing elements on the layout
- using rulers and grids
- using views
- specifying drawing options

Setting Preferences

To set various default options for either the current or subsequent Designer sessions:

1. Choose Tools—>Tools Options to show the Tools Options property sheet.

2. Specify the desired preferences such as your sound output device, image compression algorithm, page size, etc.

3. You can apply the options for either the current session, or the current and all subsequent sessions:
   - To set preferences for the current session only, choose the OK button. Settings remain in effect until you end the current session.
     
     Note: Some settings take effect immediately, and others the next time you create or open a display.
   - To set preferences for the current and all subsequent sessions, choose the Save button. Settings are saved in your Graphics preferences file, and remain in effect for subsequent sessions.

Customizing Your Work Environment

While you're working in the Designer, you can perform the following actions to customize features of your Layout Editor work environment:

- setting preferences
- specifying layout settings
- increasing and decreasing magnification
- hiding and showing elements on the layout
- using rulers and grids
- using views
- specifying drawing options

To set various default options for either the current or subsequent Designer sessions:

1. Choose **Tools** —> **Tools Options** to show the Tools Options property sheet.
2. Specify the desired preferences such as your sound output device, image compression algorithm, page size, etc.
3. You can apply the options for either the current session, or the current and all subsequent sessions:
   - To set preferences for the current session only, choose the OK button. Settings remain in effect until you end the current session.
   - To set preferences for the current and all subsequent sessions, choose the Save button. Settings are saved in your Graphics preferences file, and remain in effect for subsequent sessions.

Note: Some settings take effect immediately, and others the next time you create or open a display.
Specifying Layout Settings

To specify the dimensions of the layout or the direction in which to divide pages when printed:

1. Choose View—>View Options—>Layout to show the Layout Settings dialog.
2. Specify the desired layout size and page direction settings.
3. Choose the OK button to confirm the settings and close the Layout Settings dialog.

Settings remain in effect for the current Designer session only.

To show the entire layout:
- Choose View—>Fit to Window. Magnification is decreased so that the entire layout is visible.

Increasing and Decreasing Magnification

You can increase or decrease magnification of a selected point in the layout to make objects appear larger or smaller.

**Note:** While the objects may appear to be larger or smaller, their actual size does not change.

To increase magnification at a specific point:

1. Select the Magnify tool from the Tool palette.
2. Click on the layout where you want to increase magnification.

   Magnification is increased by a factor of two (objects appear twice as large). To double the magnification again, repeat the above steps. The maximum possible magnification is 16 times.

To decrease magnification from a specific point:

1. Select the Magnify tool from the Tool palette.
2. Hold down the Shift key and click on the layout where you want to decrease magnification.

   Magnification is decreased by a factor of one half (objects appear half as large). To decrease the magnification by one half again, repeat the above steps. You’ve reached the maximum possible reduction when the entire layout is visible.

To zoom in to the center of the layout by a factor of two:

- Choose View—>Zoom In
You can customize the layout to make it easier to create and manipulate objects. The preferences you set are saved in a local preferences file on the file system. For the name and location of this file, see the Developer/2000 Installation Guide.

To specify the dimensions of the layout or the direction in which to divide pages when printed:

1. Choose View —> View Options —> Layout to show the Layout Settings dialog.
2. Specify the desired layout size and page direction settings.
3. Choose the OK button to confirm the settings and close the Layout Settings dialog.

Settings remain in effect for the current Designer session only.

To show the entire layout:

- Choose View —> Fit to Window. Magnification is decreased so that the entire layout is visible.
- You can increase or decrease magnification of a selected point in the layout to make objects appear larger or smaller.

Note: While the objects may appear to be larger or smaller, their actual size does not change.

To increase magnification at a specific point:

1. Select the Magnify tool from the Tool palette.
2. Click on the layout where you want to increase magnification.
   
   Magnification is increased by a factor of two (objects appear twice as large). To double the magnification again, repeat the above steps. The maximum possible magnification is 16 times.

To decrease magnification from a specific point:

1. Select the Magnify tool from the Tool palette.
2. Hold down the Shift key and click on the layout where you want to decrease magnification.
   
   Magnification is decreased by a factor of one half (objects appear half as large). To decrease the magnification by one half again, repeat the above steps. You've reached the maximum possible reduction when the entire layout is visible.

To zoom in to the center of the layout by a factor of two:

- Choose View —> Zoom In.
To zoom out from the center of the layout by a factor of two:

- Choose View—>Zoom Out

To return to normal magnification:

- Choose View—>Normal Size. Magnification is returned to normal and objects appear as their actual sizes.

**Hiding and Showing Elements on the Layout**

You can hide and show the following elements on the layout:

- Tool palette
- Symbol palette
- status line
- rulers
- ruler guides
- grid
- page breaks

To hide or show elements:

- Choose the View menu and then select the element you want to hide or show. For example, to hide rulers when they’re showing, select Rulers.

The selected element is hidden. To show the element again, repeat the above steps to toggle to the alternate setting. Settings remain in effect for the current Designer session only.

**Using Rulers and Grids**

You can use rulers and grids as guides for positioning objects on the layout.

**Specifying Ruler and Grid Settings**

To specify ruler and grid settings:

1. Choose View—>View Options—>Ruler to show the Ruler Settings dialog.
2. Specify the desired ruler and grid settings such as which unit of measurement to use, grid spacing, etc.
3. Choose the OK button to confirm the settings and close the Ruler Settings dialog.

Settings remain in effect for the current Designer session only. Rulers and grids are not visible at runtime.
To zoom out from the center of the layout by a factor of two:

- Choose **View** —> **Zoom Out**

To return to normal magnification:

- Choose **View** —> **Normal Size**. Magnification is returned to normal and objects appear as their actual sizes.

You can hide and show the following elements on the layout:

- Tool palette
- Symbol palette
- Status line
- Rulers
- Ruler guides
- Grid
- Page breaks

To hide or show elements:

- Choose the **View** menu and then select the element you want to hide or show. For example, to hide rulers when they're showing, select **Rulers**.

The selected element is hidden. To show the element again, repeat the above steps to toggle to the alternate setting. Settings remain in effect for the current Designer session only.

You can use rulers and grids as guides for positioning objects on the layout.

To specify ruler and grid settings:

1. Choose **View** —> **View Options** —> **Ruler** to show the Ruler Settings dialog.
2. Specify the desired ruler and grid settings such as which unit of measurement to use, grid spacing, etc.
3. Choose the OK button to confirm the settings and close the Ruler Settings dialog.

Settings remain in effect for the current Designer session only.

Rulers and grids are not visible at runtime.
When you change the rulers’ unit of measurement, they are updated to reflect the change. In addition, when you magnify or scroll to a new area of the layout, the rulers are updated accordingly.

**Using Ruler Guides**

To place a ruler guide on the layout:

- Click on the horizontal or vertical ruler and drag the ruler guide to the desired position. Release the mouse button when the guide is positioned where you want it.

Once positioned, you can move the ruler guide by dragging it to a new location. You may place as many ruler guides on the layout as you wish.

- To remove the ruler guide, click-and-drag it back to its original ruler.

**Using Grid Snap**

When you turn grid snap on, new objects you create, move, or resize will align with the grid snap points defined in the Ruler Settings dialog box. To turn grid snap on or off:

- Choose **View**—>**Grid Snap**.

  Grid snap toggles on or off, depending on the prior setting. Settings remain in effect for the current Designer session only.

**Using Views**

You can divide the layout into multiple views to focus on different parts of a display. The layout is initially shown in one large view, but you can create up to four views, each of which can focus on a different object, part of an object, or layer. In addition, each view can have its own layer, ruler, grid, and magnification settings.

Each view is independent and has its own settings, such as rulers units, grid visibility, magnification, etc. Each view can also scroll the layout independently.

The view that you are currently working in is called the “active” view. You can recognize the active view by its scroll bars being enabled; the scroll bars of the inactive view(s) will be disabled.

Any changes you make to ruler, grid, and other settings will be applied only to the active view. Any changes you make to objects that appear on the layout will be reflected in all views focusing on that part of the layout.

To create views, you use the split bars that are initially located to the right of the horizontal scroll bar and below the vertical scroll bar, as shown in the figure below:
Working with the Layout Editor

When you change the rulers' unit of measurement, they are updated to reflect the change. In addition, when you magnify or scroll to a new area of the layout, the rulers are updated accordingly.

To place a ruler guide on the layout:
- Click on the horizontal or vertical ruler and drag the ruler guide to the desired position. Release the mouse button when the guide is positioned where you want it.
- Once positioned, you can move the ruler guide by dragging it to a new location. You may place as many ruler guides on the layout as you wish.
- To remove the ruler guide, click-and-drag it back to its original ruler.

When you turn grid snap on, new objects you create, move, or resize will align with the grid snap points defined in the Ruler Settings dialog box. To turn grid snap on or off:
- Choose View —> Grid Snap.
  Grid snap toggles on or off, depending on the prior setting. Settings remain in effect for the current Designer session only.

You can divide the layout into multiple views to focus on different parts of a display. The layout is initially shown in one large view, but you can create up to four views, each of which can focus on a different object, part of an object, or layer. In addition, each view can have its own layer, ruler, grid, and magnification settings.

Each view is independent and has its own settings, such as rulers units, grid visibility, magnification, etc. Each view can also scroll the layout independently.

The view that you are currently working in is called the “active” view. You can recognize the active view by its scroll bars being enabled; the scroll bars of the inactive view(s) will be disabled.

Any changes you make to ruler, grid, and other settings will be applied only to the active view. Any changes you make to objects that appear on the layout will be reflected in all views focusing on that part of the layout.

To create views, you use the split bars that are initially located to the right of the horizontal scroll bar and below the vertical scroll bar, as shown in the figure below:
Views appear on the layout in the Designer only, and are not visible when you run or print the display.

Creating Views  
To divide the layout into multiple views:
- Use the horizontal or vertical split bars on the scroll bar to create two, three, or four views:
  - To create two views, drag the horizontal or vertical split bar to a new position. This creates two views on one axis, each having a its own split bar on the alternate axis.
  - To create three views, create two views, then split one of the views using its split bar on the alternate axis.
  - To create four views, create two views, then split both views using their split bars on the alternate axis.

Views remain in effect for the current Designer session only, and are not visible at runtime.

Activating Views  
To work with a view, you need to activate it. Only one view can be active at a time. To activate a view:
- Click in it. The view is highlighted.

Resizing Views  
To resize a view:
- Drag its horizontal or vertical split bar to a new position.

Removing Views  
To remove a view:
- Drag its split bars back to their original positions on the scroll bars.
Creating Views

Activating Views

Resizing Views

Removing Views

4 – 36 Graphics Developer’s Guide

Split Bars

Views appear on the layout in the Designer only, and are not visible when you run or print the display.

To divide the layout into multiple views:

• To create two views, drag the horizontal or vertical split bar to a new position. This creates two views on one axis, each having a its own split bar on the alternate axis.

• To create three views, create two views, then split one of the views using its split bar on the alternate axis.

• To create four views, create two views, then split both views using their split bars on the alternate axis.

Views remain in effect for the current Designer session only, and are not visible at runtime.

To work with a view, you need to activate it. Only one view can be active at a time. To activate a view:

• Click in it. The view is highlighted.

To resize a view:

• Drag its horizontal or vertical split bar to a new position.

To remove a view:

• Drag its split bars back to their original positions on the scroll bars.
Specifying Drawing Options

You can specify default options for certain objects to be applied automatically when you draw them on the layout. For example, options for text include alignment, scalable bounding box, scalable fonts, and wraparound.

To specify drawing options:

1. Choose the Format—>Drawing Options, and then choose one of the following: General, Arc, Text, Rounded Rectangle, or Image. A dialog box appears for the option you selected.
2. Specify the desired options. Selections remain in effect for the current display until you change them again.

Using Windows

In Graphics, windows can function in two different ways: as work areas and as display areas.

In the Designer, windows are rectangular work areas that contain Graphics interface elements such as editors, tables, and browsers. You can open multiple windows, but only one window can be active at a time.

The default window configuration defines the position and number of windows that appear when you open a display. By default, the Layout editor window is the only one opened. However, you can set up a different window configuration and save it as the new default using the Preferences dialog box.

You can also create windows programmatically to present objects in different areas of the layout at runtime. By default, a display’s objects appear in the Main Layout window. Using PL/SQL, you can create additional windows in which to present objects. You can also define a window’s size and position and when it should open or close.

For more information about using windows in your runtime displays, see “Runtime Issues” on page 15 – 1.
You can specify default options for certain objects to be applied automatically when you draw them on the layout. For example, options for text include alignment, scalable bounding box, scalable fonts, and wraparound.

To specify drawing options:

1. Choose the **Format** —> **Drawing Options**, and then choose one of the following: **General**, **Arc**, **Text**, **Rounded Rectangle**, or **Image**. A dialog box appears for the option you selected.

2. Specify the desired options. Selections remain in effect for the current display until you change them again.

Using Windows

In Graphics, windows can function in two different ways: as work areas and as display areas.

In the Designer, windows are rectangular work areas that contain Graphics interface elements such as editors, tables, and browsers. You can open multiple windows, but only one window can be active at a time.

The default window configuration defines the position and number of windows that appear when you open a display. By default, the Layout editor window is the only one opened. However, you can set up a different window configuration and save it as the new default using the Preferences dialog box.

You can also create windows programmatically to present objects in different areas of the layout at runtime. By default, a display's objects appear in the Main Layout window. Using PL/SQL, you can create additional windows in which to present objects. You can also define a window's size and position and when it should open or close.

For more information about using windows in your runtime displays, see “Runtime Issues” on page 15 – 1.
CHAPTER

Using the Object Navigator

This chapter provides step-by-step instructions on how to use the Graphics Object Navigator to create and manipulate all of the objects in your Graphics displays.

The following topics are covered in this chapter:

- components of the Object Navigator – 5 – 2
- object hierarchy – 5 – 3
- creating objects – 5 – 4
- selecting objects – 5 – 4
- modifying objects – 5 – 5
- reordering objects – 5 – 6
- sharing objects between displays – 5 – 6
- renaming objects – 5 – 6
- deleting objects – 5 – 7
- expanding and collapsing nodes – 5 – 8
- marking objects – 5 – 9
- searching for objects – 5 – 9
CHAPTER 5

5 – 1 Using the Object Navigator

This chapter provides step-by-step instructions on how to use the Graphics Object Navigator to create and manipulate all of the objects in your Graphics displays.

The following topics are covered in this chapter:

• components of the Object Navigator – 5 – 2
• object hierarchy – 5 – 3
• creating objects – 5 – 4
• selecting objects – 5 – 4
• modifying objects – 5 – 5
• reordering objects – 5 – 6
• sharing objects between displays – 5 – 6
• renaming objects – 5 – 6
• deleting objects – 5 – 7
• expanding and collapsing nodes – 5 – 8
• marking objects – 5 – 9
• searching for objects – 5 – 9
Components of the Object Navigator

The Object Navigator components are described below.

**Nodes Display Area**
The Nodes display area is the main component of the Object Navigator. This is the area where all objects currently available to you are organized hierarchically.

**Toolbar**
The Toolbar contains common commands also available from the Navigator or File menu, such as Save, Run, Create, Delete, Expand, and Collapse. Online help is available for each tool by simply placing your mouse above the button.

**Location Indicator**
The location indicator reveals the name of the currently selected object in the nodes display area.

**Find Field**
The Find field allows you to search for an object in the Navigator by typing in a partial or full name of the object.
Components of the Object Navigator

The Object Navigator components are described below.

The Nodes display area is the main component of the Object Navigator. This is the area where all objects currently available to you are organized hierarchically.

The Toolbar contains common commands also available from the Navigator or File menu, such as Save, Run, Create, Delete, Expand, and Collapse. Online help is available for each tool by simply placing your mouse above the button.

The location indicator reveals the name of the currently selected object in the nodes display area.

The Find field allows you to search for an object in the Navigator by typing in a partial or full name of the object.
Object Hierarchy

Most objects in the Navigator hierarchy are automatically assigned one or more default subobjects, depending on the type of object. There are two basic types of these default subobjects:

- assignment
- reference

For example, each graphic object contains assignment subobjects for a button procedure, an execute query trigger, and a set parameter function. Each program unit contains reference subobjects for references contained in the program unit, and references to the program unit itself.

The following table shows the default subobjects for the first layer of objects in the Display node:

<table>
<thead>
<tr>
<th>Node</th>
<th>Object</th>
<th>Default Subobjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>layerx</td>
<td>Button Procedure: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Execute Query: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Parameter: None (or name)</td>
</tr>
<tr>
<td>Templates</td>
<td>templatex</td>
<td>Referenced By</td>
</tr>
<tr>
<td>Queries</td>
<td>queryx</td>
<td>Custom Exec Trigger: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Post Exec Trigger: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Referenced By</td>
</tr>
<tr>
<td>Parameters</td>
<td>PARAMx</td>
<td>Referenced By</td>
</tr>
<tr>
<td>Timers</td>
<td>timerx</td>
<td>Timer Procedure: None (or name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Referenced By</td>
</tr>
<tr>
<td>Program Units</td>
<td>name</td>
<td>Specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>References</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Referenced By</td>
</tr>
</tbody>
</table>

These default subobjects are designed to help you track the cross references between objects in your display. Specifically, between objects and program units, charts and templates, parameters and the objects that assign their values, etc.

For example, if you define a drill-down procedure on a graphic object to set the value of PARAM0, the Set Parameter node for the graphic object will be set to PARAM0. Similarly, the Reference By node of PARAM0 would be set to object_name.
Most objects in the Navigator hierarchy are automatically assigned one or more default subobjects, depending on the type of object. There are two basic types of these default subobjects:

- **Assignment**
- **Reference**

For example, each graphic object contains assignment subobjects for a button procedure, an execute query trigger, and a set parameter function. Each program unit contains reference subobjects for references contained in the program unit, and references to the program unit itself.

The following table shows the default subobjects for the first layer of objects in the Display node:

<table>
<thead>
<tr>
<th>Node</th>
<th>Object</th>
<th>Default Subobjects</th>
</tr>
</thead>
</table>
| Layout | Layer | Button Procedure: None (or name)  
Execute Query: None (or name)  
Set Parameter: None (or name) |
| Templates | Template | Referenced By |
| Parameters | PARAM | Referenced By |
| Timers | Timer | Referenced By |
| Program Units | Specifications | References |

These default subobjects are designed to help you track the cross references between objects in your display. Specifically, between objects and program units, charts and templates, parameters and the objects that assign their values, etc.

For example, if you define a drill-down procedure on a graphic object to set the value of PARAM0, the Set Parameter node for the graphic object will be set to PARAM0. Similarly, the Reference By node of PARAM0 would be set to `object_name`.
Creating Objects

To create a new object in the Object Navigator:

1. Select the top node for the type of object you wish to create, or an already existing object of that type.

2. Choose the Create button or **Navigator—>Create**.

The type of object you’re creating determines what events occur and how you proceed.

**Suggestion:** If there are no existing objects of the type you wish to create, double-click on the top node for the type of object you wish to create.

For more information, refer to the table below:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>displays</td>
<td>“Creating Displays” — 6 – 2</td>
</tr>
<tr>
<td>layers</td>
<td>“Adding Layers” — 6 – 8</td>
</tr>
<tr>
<td>templates</td>
<td>“Working with Chart Templates” — 9 – 5</td>
</tr>
<tr>
<td>queries</td>
<td>“Creating Queries” — 7 – 3</td>
</tr>
<tr>
<td>parameters</td>
<td>“Creating Parameters” — 10 – 2</td>
</tr>
<tr>
<td>sounds</td>
<td>“Recording Sounds” — 6 – 12</td>
</tr>
<tr>
<td>timers</td>
<td>“Creating Timer Triggers” — 11 – 9</td>
</tr>
<tr>
<td>program units</td>
<td>“Creating Program Units” — 12 – 2</td>
</tr>
<tr>
<td>libraries</td>
<td>“Attaching PL/SQL Libraries” — 12 – 9</td>
</tr>
</tbody>
</table>

Selecting Objects

Each node listed in the Navigator has three basic parts from left to right:

- **subobject indicator**: The highlighted plus sign indicates that you can expand this node to view its subobjects, an inactive plus sign indicates that the node has no subobjects and cannot be expanded. The minus sign indicates that the node is already expanded.

- **type icon**: Indicates the node type such as layer, chart, chart element, shape, etc.

- **type icon**: Indicates the node type such as program unit, library, breakpoint, etc.
Creating Objects

To create a new object in the Object Navigator:

1. Select the top node for the type of object you wish to create, or an already existing object of that type.
2. Choose the Create button or Navigator —> Create.

The type of object you're creating determines what events occur and how you proceed.

Suggestion: If there are no existing objects of the type you wish to create, double-click on the top node for the type of object you wish to create.

For more information, refer to the table below:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>displays</td>
<td>&quot;Creating Displays&quot; – 6 – 2</td>
</tr>
<tr>
<td>layers</td>
<td>&quot;Adding Layers&quot; – 6 – 8</td>
</tr>
<tr>
<td>templates</td>
<td>&quot;Working with Chart Templates&quot; – 9 – 5</td>
</tr>
<tr>
<td>queries</td>
<td>&quot;Creating Queries&quot; – 7 – 3</td>
</tr>
<tr>
<td>parameters</td>
<td>&quot;Creating Parameters&quot; – 10 – 2</td>
</tr>
<tr>
<td>sounds</td>
<td>&quot;Recording Sounds&quot; – 6 – 12</td>
</tr>
<tr>
<td>timers</td>
<td>&quot;Creating Timer Triggers&quot; – 11 – 9</td>
</tr>
<tr>
<td>program units</td>
<td>&quot;Creating Program Units&quot; – 12 – 2</td>
</tr>
<tr>
<td>libraries</td>
<td>&quot;Attaching PL/SQL Libraries&quot; – 12 – 9</td>
</tr>
</tbody>
</table>

Selecting Objects

Each node listed in the Navigator has three basic parts from left to right:

- The highlighted plus sign indicates that you can expand this node to view its subobjects.
- An inactive plus sign indicates that the node has no subobjects and cannot be expanded.
- The minus sign indicates that the node is already expanded.

- Indicates the node type such as layer, chart, chart element, shape, etc.
- Indicates the node type such as program unit, library, breakpoint, etc.

subobject indicator
type icon
subobject indicator
type icon
name Either the default name assigned by Graphics, or a name you have customized

To select an object in the Object Navigator:

■ Click once on the object’s type icon or name in the Object Navigator.

Notice that the name of the selected object appears in the Object Navigator window’s title bar, so you know which object is currently selected.

Notice also that selecting an existing node (e.g., SAL_bars) in the Object Navigator selects the corresponding object(s) in the Layout editor.

You can select multiple objects in the Navigator by holding down the Shift key while clicking on the object’s type icon.

Modifying Objects

Most objects are modified through a property sheet. You can access the property sheet for any object from the Object Navigator in one of three ways:

■ Double-click on the icon of the object you wish to edit.

■ Select the object and choose the Properties command from the pop-up menu.

■ Select the object and choose Tools→Properties.

The type of object you’re editing will determine what events occur and how you proceed. For more information, see the sections below:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>layers</td>
<td>“Using Layers” – 6 – 7</td>
</tr>
<tr>
<td>charts</td>
<td>“Working with Charts” – 8 – 3</td>
</tr>
<tr>
<td>templates</td>
<td>“Using Chart Templates” – 9 – 1</td>
</tr>
<tr>
<td>queries</td>
<td>“Using Queries” – 7 – 1</td>
</tr>
<tr>
<td>parameters</td>
<td>“Using Parameters” – 10 – 1</td>
</tr>
<tr>
<td>sounds</td>
<td>“Working with Sounds” – 6 – 12</td>
</tr>
<tr>
<td>program units</td>
<td>“Editing Program Units” – 12 – 3</td>
</tr>
<tr>
<td>libraries</td>
<td>“Attaching PL/SQL Libraries” – 12 – 9</td>
</tr>
</tbody>
</table>
Using the Object Navigator

Either the default name assigned by Graphics, or a name you have customized

To select an object in the Object Navigator:
- Click once on the object's type icon or name in the Object Navigator.

Notice that the name of the selected object appears in the Object Navigator window's title bar, so you know which object is currently selected.

Notice also that selecting an existing node (e.g., SAL_bars) in the Object Navigator selects the corresponding object(s) in the Layout editor. You can select multiple objects in the Navigator by holding down the Shift key while clicking on the object's type icon.

Modifying Objects

Most objects are modified through a property sheet. You can access the property sheet for any object from the Object Navigator in one of three ways:
- Double-click on the icon of the object you wish to edit.
- Select the object and choose the Properties command from the pop-up menu.
- Select the object and choose Tools —> Properties.

The type of object you're editing will determine what events occur and how you proceed. For more information, see the sections below:

- Object Type Section
  - layers: "Using Layers" – 6 – 7
  - charts: "Working with Charts" – 8 – 3
  - templates: "Using Chart Templates" – 9 – 1
  - queries: "Using Queries" – 7 – 1
  - parameters: "Using Parameters" – 10 – 1
  - sounds: "Working with Sounds" – 6 – 12
  - program units: "Editing Program Units" – 12 – 3
  - libraries: "Attaching PL/SQL Libraries" – 12 – 9
Reordering Objects

Objects are listed in the Navigator in the order in which they are created. You can change the order that objects are listed in the Navigator.

To move an object in the Object Navigator:

1. Click and drag the object to its new position in the object hierarchy.

Note: This action only changes the “order” of objects in the Navigator hierarchy. To change the physical position of the object in the layout, you must move the object in the Layout editor.

Sharing Objects Between Displays

In addition to reordering objects within a display, you can move queries, program units, parameters, timers, and sounds from one open display to another open display. You can also move program units to and from attached or open libraries.

When you move a program unit from one display to another, it loses its compiled state and must be recompiled in its new context.

To move a copy of any of the above objects from one display to another:

1. Select the query, parameter, timer, sound, or program unit you wish to move.

2. Hold down the select button on the object and drag it vertically to its new location. The cursor changes to a crosshair and a dashed horizontal line indicates the current node location of the object in the Navigator.

3. Release the mouse button to “drop” the object. You must “drop” the object into its correct node in the object hierarchy (i.e., queries must be dropped beneath the Query node, etc.).

Renaming Objects

You can rename any object you have created within a display layout. This includes layers, queries, graphic objects, parameters, timers, sounds, charts, and chart templates.

If you wish to rename a PL/SQL program unit, you must do so from the Program Unit editor. For more information, see the section “Renaming Program Units” on page 12 – 5.
Reordering Objects

Objects are listed in the Navigator in the order in which they are created. You can change the order that objects are listed in the Navigator.

To move an object in the Object Navigator:

1. Click and drag the object to its new position in the object hierarchy.

Note: This action only changes the "order" of objects in the Navigator hierarchy. To change the physical position of the object in the layout, you must move the object in the Layout editor.

Sharing Objects Between Displays

In addition to reordering objects within a display, you can move queries, program units, parameters, timers, and sounds from one open display to another open display. You can also move program units to and from attached or open libraries.

When you move a program unit from one display to another, it loses its compiled state and must be recompiled in its new context.

To move a copy of any of the above objects from one display to another:

1. Select the query, parameter, timer, sound, or program unit you wish to move.
2. Hold down the select button on the object and drag it vertically to its new location. The cursor changes to a crosshair and a dashed horizontal line indicates the current node location of the object in the Navigator.
3. Release the mouse button to "drop" the object. You must "drop" the object into its correct node in the object hierarchy (i.e., queries must be dropped beneath the Query node, etc.).

Renaming Objects

You can rename any object you have created within a display layout. This includes layers, queries, graphic objects, parameters, timers, sounds, charts, and chart templates.

If you wish to rename a PL/SQL program unit, you must do so from the Program Unit editor. For more information, see the section "Renaming Program Units" on page 12–5.
Renaming Objects

To rename an object in the Object Navigator:

1. Select the object you wish to rename.
2. Move the mouse pointer over the object name.
   The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.

   Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

If you rename an object that is referenced within a program unit, you must change those references as well to reflect the new name.

Changing References

If you change the name of an object that is referenced within one or more program units, you can change the name of the object in the program unit using the Program Unit editor, as follows:

1. Double-click on the program unit that references the object to show the program unit in the Program Unit editor.
2. Choose Edit—>Search/Replace to show the Search/Replace dialog.
3. Type the old object name in the Search field.
   You can specify a case sensitive search, or use a regular expression to search by.
4. Type the new object name in the Replace With field.
5. Choose the Replace All button to replace all occurrences of the old object name with the new object name in the current program unit.

To search for and replace a name in all program units in the current scope (i.e. the Program Units node, or an open library node), choose Edit—>Search/Replace All.

Deleting Objects

To delete an object in the Object Navigator:

1. Select the object you wish to delete.
2. Choose the Delete button or choose Navigator—>Delete.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.
Renaming Objects

Changing References

5 - 7 Using the Object Navigator

To rename an object in the Object Navigator:
1. Select the object you wish to rename.
2. Move the mouse pointer over the object name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.

Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

If you rename an object that is referenced within a program unit, you must change those references as well to reflect the new name.

If you change the name of an object that is referenced within one or more program units, you can change the name of the object in the program unit using the Program Unit editor, as follows:
1. Double-click on the program unit that references the object to show the program unit in the Program Unit editor.
2. Choose Edit —> Search/Replace to show the Search/Replace dialog.
3. Type the old object name in the Search field. You can specify a case sensitive search, or use a regular expression to search by.
4. Type the new object name in the Replace With field.
5. Choose the Replace All button to replace all occurrences of the old object name with the new object name in the current program unit.

To search for and replace a name in all program units in the current scope (i.e. the Program Units node, or an open library node), choose Edit —> Search/Replace All.

Deleting Objects

To delete an object in the Object Navigator:
1. Select the object you wish to delete.
2. Choose the Delete button or choose Navigator —> Delete.

An alert appears, prompting you to confirm the deletion.

3. Choose the Yes button to confirm.
Note: If you delete an item that is referenced within a program unit, you must edit the program unit to reflect the deletion. Notice that the program unit containing the reference loses its compiled status.

Expanding and Collapsing Nodes

The Object Navigator initially appears with only its top-level nodes visible. A plus sign (+) signifies that the node contains hidden subobjects (i.e., the object is *collapsed*). A minus sign (−) signifies that all subobjects are visible (i.e., the object is *expanded*).

You can expand and collapse nodes one level at a time, or all at once.

A node has subobjects if the plus sign is highlighted. For example, when you first open Graphics, the plus sign for the Libraries node appears inactive since you have not yet opened any PL/SQL libraries.

Expanding Nodes

To expand a node and show its first layer of subobjects:

1. Select a node containing subobjects.
2. Choose the Expand button or choose *Navigator—>Expand* to show the node’s first layer of subobjects.

   **Suggestion:** Alternatively, you can simply select the node’s highlighted plus sign to expand the first layer of subobjects.

To expand a node and show all of its subobjects:

1. Select a collapsed node.
2. Choose the Expand All button or *Navigator—>Expand All* to show all of the node’s subobjects.

Collapsing Nodes

To collapse a node and hide its first layer of subobjects:

1. Select the expanded node.
2. Choose the Collapse button or *Navigator—>Collapse*. The node’s subobjects are hidden.

   The collapsed node is prefaced with a highlighted plus sign.

   **Suggestion:** Alternatively, you can simply select the node’s highlighted minus sign to collapse the first layer of subobjects.

To collapse a node and hide all of its subobjects:

1. Select an expanded node.
Note: If you delete an item that is referenced within a program unit, you must edit the program unit to reflect the deletion. Notice that the program unit containing the reference loses its compiled status.

Expanding and Collapsing Nodes

The Object Navigator initially appears with only its top-level nodes visible. A plus sign (+) signifies that the node contains hidden subobjects (i.e., the object is collapsed). A minus sign (−) signifies that all subobjects are visible (i.e., the object is expanded).

You can expand and collapse nodes one level at a time, or all at once. A node has subobjects if the plus sign is highlighted. For example, when you first open Graphics, the plus sign for the Libraries node appears inactive since you have not yet opened any PL/SQL libraries.

To expand a node and show its first layer of subobjects:
1. Select a node containing subobjects.
2. Choose the Expand button or choose Navigator—>Expand to show the node's first layer of subobjects.
   Suggestion: Alternatively, you can simply select the node's highlighted plus sign to expand the first layer of subobjects.

To expand a node and show all of its subobjects:
1. Select a collapsed node.
2. Choose the Expand All button or Navigator—>Expand All to show all of the node's subobjects.

To collapse a node and hide its first layer of subobjects:
1. Select the expanded node.
2. Choose the Collapse button or Navigator—>Collapse. The node's subobjects are hidden.
   Suggestion: Alternatively, you can simply select the node's highlighted minus sign to collapse the first layer of subobjects.

To collapse a node and hide all of its subobjects:
1. Select an expanded node.
2. Choose the Collapse All button or **Navigator**—>**Collapse All** to hide all of the subobjects.

**Marking Objects**

Marking an object in the Object Navigator allows you to record the location of the currently selected object. Once you’ve marked an object you can perform operations in the Object Navigator or any other window, then quickly return to the previously “marked” object.

You can set one mark in the Object Navigator. Each subsequent mark that you create replaces the previously created mark.

To mark an object in the Object Navigator:

1. Select the object you wish to mark.
2. Choose **Navigator**—>**Set Mark**.

To go to a previously marked object:

Choose **Navigator**—>**Goto Mark** to select the object you previously marked.

**Searching for Objects**

You can search for any named object in the Object Navigator by entering an object name in the Find field at the top of the Object Navigator

1. Type the name, full or partial, of the object you wish to find in the Find field.

   As soon as you begin typing, Graphics begins an incremental forward search from the top of the Object Navigator. The incremental search continues until you stop typing.

   Graphics expands nodes as necessary to reveal its current location during the search.

2. Choose either the Search Forward button or the Search Backward button to find the next or previous occurrence of the search criteria in the Object Navigator.

Wild cards are not currently supported in the Find field.
5 – 9

Using the Object Navigator

2. Choose the Collapse All button or Navigator —> Collapse All to hide all of the subobjects.

Marking Objects

Marking an object in the Object Navigator allows you to record the location of the currently selected object. Once you've marked an object you can perform operations in the Object Navigator or any other window, then quickly return to the previously "marked" object.

You can set one mark in the Object Navigator. Each subsequent mark that you create replaces the previously created mark.

To mark an object in the Object Navigator:

1. Select the object you wish to mark.
2. Choose Navigator —> Set Mark.

To go to a previously marked object:

Choose Navigator —> Goto Mark to select the object you previously marked.

Searching for Objects

You can search for any named object in the Object Navigator by entering an object name in the Find field at the top of the Object Navigator.

1. Type the name, full or partial, of the object you wish to find in the Find field.

As soon as you begin typing, Graphics begins an incremental forward search from the top of the Object Navigator. The incremental search continues until you stop typing.

Graphics expands nodes as necessary to reveal its current location during the search.

2. Choose either the Search Forward button or the Search Backward button to find the next or previous occurrence of the search criteria in the Object Navigator.

Wild cards are not currently supported in the Find field.
Managing Displays

This chapter discusses how to work with the displays you will create in Graphics Designer. These displays can be used in Graphics Runtime, or embedded within other applications.

The following topics are covered in this chapter:

- components of displays – 6 – 2
- creating displays – 6 – 2
- opening displays – 6 – 3
- importing and exporting artwork – 6 – 4
- importing displays – 6 – 6
- using layers in displays – 6 – 7
- working with sounds – 6 – 12
- saving displays – 6 – 15
- closing displays – 6 – 17
- testing displays – 6 – 17
- generating displays – 6 – 18
- printing displays – 6 – 19
- granting and revoking access to displays – 6 – 19
- renaming displays – 6 – 20
- deleting displays – 6 – 21
CHAPTER 6
Managing Displays

This chapter discusses how to work with the displays you will create in Graphics Designer. These displays can be used in Graphics Runtime, or embedded within other applications.

The following topics are covered in this chapter:

• components of displays – 6 – 2
• creating displays – 6 – 2
• opening displays – 6 – 3
• importing and exporting artwork – 6 – 4
• importing displays – 6 – 6
• using layers in displays – 6 – 7
• working with sounds – 6 – 12
• saving displays – 6 – 15
• closing displays – 6 – 17
• testing displays – 6 – 17
• generating displays – 6 – 18
• printing displays – 6 – 19
• granting and revoking access to displays – 6 – 19
• renaming displays – 6 – 20
• deleting displays – 6 – 21
Components of Displays

The displays you create with Graphics may consist of many components. Not all display components are covered in this chapter. Use the following list for more information:

<table>
<thead>
<tr>
<th>Component</th>
<th>See This Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>charts</td>
<td>“Creating Charts” – 8 – 8</td>
</tr>
<tr>
<td>graphic objects</td>
<td>“Working with Objects” – 4 – 6</td>
</tr>
<tr>
<td>imported artwork</td>
<td>“Importing and Exporting Artwork” – 6 – 4</td>
</tr>
<tr>
<td>sounds</td>
<td>“Working with Sounds” – 6 – 12</td>
</tr>
<tr>
<td>layers</td>
<td>“Using Layers” – 6 – 7</td>
</tr>
</tbody>
</table>

Page Breaks

Page breaks are solid dark lines that appear on the layout, each one representing a boundary of a printed page. The page size is determined by the settings in the Print Setup dialog.

Page breaks appear in the Designer only, and are not visible when the display is run or printed.

Copyright Information

You may want to add copyright information such as your name and the last date of update to a display you have created. This information is useful to other users who may only receive the runtime display but wish to know more about the display such as who created it and when.

Creating Displays

Every time you invoke the Designer, a new display initially named Disp1 appears automatically. This new display is created for your convenience.

You can specify that no initial display is created when you start Graphics Designer. Choose Tools—>Tools Options to display the Tools Options property sheet. Choose the Preferences tab. In the Startup block, select No Display from the Open drop-down list. Choose the Save button to save this setting to your preferences file.

You can create new displays either from the Object Navigator, or using the menu. You can create and work on more than one display at a time.

Creating New Displays in the Navigator

To create a new display in the Object Navigator:

1. Select the Displays node and choose the Create button or Navigator—>Create.
The displays you create with Graphics may consist of many components. Not all display components are covered in this chapter. Use the following list for more information:

<table>
<thead>
<tr>
<th>Component</th>
<th>See This Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>charts</td>
<td>“Creating Charts” – 8 – 8</td>
</tr>
<tr>
<td>graphic objects</td>
<td>“Working with Objects” – 4 – 6</td>
</tr>
<tr>
<td>imported artwork</td>
<td>“Importing and Exporting Artwork” – 6 – 4</td>
</tr>
<tr>
<td>sounds</td>
<td>“Working with Sounds” – 6 – 12</td>
</tr>
<tr>
<td>layers</td>
<td>“Using Layers” – 6 – 7</td>
</tr>
</tbody>
</table>

Page breaks are solid dark lines that appear on the layout, each one representing a boundary of a printed page. The page size is determined by the settings in the Print Setup dialog.

Page breaks appear in the Designer only, and are not visible when the display is run or printed.

You may want to add copyright information such as your name and the last date of update to a display you have created. This information is useful to other users who may only receive the runtime display but wish to know more about the display such as who created it and when.

Creating Displays

Every time you invoke the Designer, a new display initially named Disp1 appears automatically. This new display is created for your convenience.

You can specify that no initial display is created when you start Graphics Designer. Choose Tools —> Tools Options to display the Tools Options property sheet. Choose the Preferences tab. In the Startup block, select No Display from the Open drop-down list. Choose the Save button to save this setting to your preferences file.

You can create new displays either from the Object Navigator, or using the menu. You can create and work on more than one display at a time.

To create a new display in the Object Navigator:

- Select the Displays node and choose the Create button or Navigator —> Create.
Creating New Displays with the Menu

Suggestion: If no displays are currently open, double-click on the Displays node in the Navigator to create a new display.

A new display appears in its own Layout editor. The default name is $Disp_x$, where $x$ is an instance number that increments by one, each time a new display is created during the current session.

Opening Displays

To create a new display from the menu:

1. Choose File $\rightarrow$ New.

A new display appears in its own Layout editor. The default name is $Disp_x$, where $x$ is an instance number that increments by one, each time a new display is created during the current session.

Opening Displays from the File System

You can open an existing display stored in either the file system or the database.

To open a display that is stored on the file system:

1. Choose File $\rightarrow$ Open to display the Filter dialog.
2. Make sure the radio button for File System is selected in the Object Access field.
3. Select the Displays or All Files check box in the Display field.
4. Choose the OK button to accept the Filter dialog and open your system file dialog.
5. Select a display to open and accept the dialog box.

Displays use the .ogd file extension by default.

The display is opened in the Designer.

Opening Displays from the Database

To open a display that is stored in the database:

1. Choose File $\rightarrow$ Open to display the Filter dialog.
2. Select the radio button for Database in the Object Access field.
3. Select the Displays or All Files check box in the Display field.
4. Choose the OK button to accept the Filter dialog and display the Open from Database dialog.

If you are not connected to a database, Graphics presents the Connect dialog.
Creating New Displays

Opening Displays from the File System

Opening Displays from the Database

Managing Displays

Suggestion:
If no displays are currently open, double-click on the Displays node in the Navigator to create a new display.

A new display appears in its own Layout editor. The default name is Dispx, where x is an instance number that increments by one each time a new display is created during the current session.

To create a new display from the menu:

- Choose File —> New.

A new display appears in its own Layout editor. The default name is Dispx, where x is an instance number that increments by one each time a new display is created during the current session.

Opening Displays

You can open an existing display stored in either the file system or the database.

To open a display that is stored on the file system:

1. Choose File —> Open to display the Filter dialog.
2. Make sure the radio button for File System is selected in the Object Access field.
3. Select the Displays or All Files check box in the Display field.
4. Choose the OK button to accept the Filter dialog and open your system file dialog.
5. Select a display to open and accept the dialog box.

Displays use the .ogd file extension by default. The display is opened in the Designer.

To open a display that is stored in the database:

1. Choose File —> Open to display the Filter dialog.
2. Select the radio button for Database in the Object Access field.
3. Select the Displays or All Files check box in the Display field.
4. Choose the OK button to accept the Filter dialog and display the Open from Database dialog.

If you are not connected to a database, Graphics presents the Connect dialog.
If your display does not appear in the list, select Update List.

5. Select a display to open from the list.

6. Choose the OK button to accept the dialog box and open the display.

The display is opened in the Designer.

**Importing and Exporting Artwork**

You can import artwork stored in the database or file system into the current display. Likewise, you can export artwork from your display to the database or file system. Artwork includes:

- line-art drawings created in Graphics
- line-art drawings created outside of Graphics
- bit-mapped images

**Importing a Drawing**

Graphics accepts drawing files saved in the CGM format or artwork files from other Oracle applications saved in Oracle Format. Graphics imports the drawing into the active layer unless otherwise specified.

To import a drawing file:

1. Choose **Edit**—>**Import**—>**Drawing** to display the Import dialog.

2. Indicate the data source for the file by selecting either the Database or File System radio button.

3. Type the file/module name in the corresponding field, or select the appropriate Browse button to display your operating system file dialog, or the Database dialog where you can select the file/module.

4. Select the drawing format type from the Format drop-down list.

5. Choose the OK button to import the drawing file.

The imported artwork appears on the layout.

**Note:** If you have grouped objects in a file and then converted the file to CGM format, those objects will no longer be grouped.

**Importing an Image**

Graphics accepts image files in a number of common graphic formats.

To import an image file:

1. Choose **Edit**—>**Import**—>**Image** to display the Import dialog.

2. Indicate the data source for the file by selecting either the Database or File System radio button.
Importing a Drawing

If your display does not appear in the list, select Update List.

5. Select a display to open from the list.
6. Choose the OK button to accept the dialog box and open the display.

The display is opened in the Designer.

Importing and Exporting Artwork

You can import artwork stored in the database or file system into the current display. Likewise, you can export artwork from your display to the database or file system. Artwork includes:

• line-art drawings created in Graphics
• line-art drawings created outside of Graphics
• bit-mapped images

Graphics accepts drawing files saved in the CGM format or artwork files from other Oracle applications saved in Oracle Format. Graphics imports the drawing into the active layer unless otherwise specified.

To import a drawing file:

1. Choose Edit —> Import —> Drawing to display the Import dialog.
2. Indicate the data source for the file by selecting either the Database or File System radio button.
3. Type the file/module name in the corresponding field, or select the appropriate Browse button to display your operating system file dialog, or the Database dialog where you can select the file/module.
4. Select the drawing format type from the Format drop-down list.
5. Choose the OK button to import the drawing file.

The imported artwork appears on the layout.

Note: If you have grouped objects in a file and then converted the file to CGM format, those objects will no longer be grouped.

Graphics accepts image files in a number of common graphic formats.

To import an image file:

1. Choose Edit —> Import —> Image to display the Import dialog.
2. Indicate the data source for the file by selecting either the Database or File System radio button.
3. Type the file/module name in the corresponding field, or select the appropriate Browse button to display your operating system file dialog, or the Database dialog where you can select the file/module.

4. Select the image format type from the Format drop-down list.

5. Choose the OK button to import the image file.

The imported artwork appears on the layout.

You can also import artwork using PL/SQL. The following example imports an image onto the active layer in the active window:

```plsql
PROCEDURE import_the_image IS
    the_image   OG_OBJECT;
    the_layer   OG_LAYER;
BEGIN
    the_layer:=OG_GET_LAYER('layer1');
    OG_ACTIVATE_LAYER(the_layer);
    the_image:=OG_IMPORT_IMAGE('my_image',OG_FILESYSTEM,
                                   OG_TIFF_IFORMAT);
END;
```


Exporting Artwork

You can export an image from a display, or export all visible layers in the display as a drawing. You can export artwork to the file system, or to a database. To export artwork from the current display:

**Note:** If you have grouped objects in the display, those objects will not retain their grouped properties once exported to a CGM file.

1. If you are exporting an image, select the image. If you are exporting the entire display as a drawing, Graphics will automatically include all artwork on the visible layers of the display.

2. Choose Edit—>Export, and then choose the type of artwork you want to export.

   For example, to export objects you’ve drawn in Graphics, choose Drawing. The Export dialog appears.

3. Specify the desired options such as the file format to export the artwork in, filename and location, etc.

4. Choose the OK button to confirm the information and close the Export dialog.

   The artwork is exported to the location you specified.
3. Type the file/module name in the corresponding field, or select the appropriate Browse button to display your operating system file dialog, or the Database dialog where you can select the file/module.

4. Select the image format type from the Format drop-down list.

5. Choose the OK button to import the image file. The imported artwork appears on the layout.

You can also import artwork using PL/SQL. The following example imports an image onto the active layer in the active window:

```plsql
PROCEDURE import_the_image IS
    the_image   OG_OBJECT;
    the_layer   OG_LAYER;
BEGIN
    the_layer:=OG_GET_LAYER('layer1');
    OG_ACTIVATE_LAYER(the_layer);
    the_image:=OG_IMPORT_IMAGE('my_image',OG_FILESYSTEM,OG_TIFF_IFORMAT);
END;
```


You can export an image from a display, or export all visible layers in the display as a drawing. You can export artwork to the file system, or to a database. To export artwork from the current display:

1. If you are exporting an image, select the image. If you are exporting the entire display as a drawing, Graphics will automatically include all artwork on the visible layers of the display.

2. Choose Edit —> Export, and then choose the type of artwork you want to export. For example, to export objects you've drawn in Graphics, choose Drawing. The Export dialog appears.

3. Specify the desired options such as the file format to export the artwork in, filename and location, etc.

4. Choose the OK button to confirm the information and close the Export dialog. The artwork is exported to the location you specified.

Note: If you have grouped objects in the display, those objects will not retain their grouped properties once exported to a CGM file.
**Note:** The procedures for granting and revoking access, renaming, and deleting artwork are the same as those for displays.

For more information about administering displays, see the following sections:

- “Granting and Revoking Access to Displays” – 6 – 19
- “Renaming Displays” – 6 – 20
- “Deleting Displays” – 6 – 21

You can also export artwork using PL/SQL. The following example exports the contents of a layer to a CGM file:

```plsql
PROCEDURE export_the_drawing IS
  the_layer   OG_OBJECT;
BEGIN
  the_layer:=OG_GET_LAYER('layer0');
  OG_EXPORT_DRAWING('my_draw', OG_FILESYSTEM,
                     OG_CGM16_DFORMAT, the_layer);
END;
```


### Importing Displays

You can import a display that is stored on the file system or in a database into your current display. This is a way to copy the contents of one display into another existing display and effectively combine the two displays.

To import a display:

1. Choose **Edit—>Import—>Display** to display the Import Display dialog.
2. Specify the name and location of the display, whether or not you want the active layers from each display merged or not, and how to resolve naming conflicts.
3. Choose the Import button to accept the dialog and import the display.
Note: The procedures for granting and revoking access, renaming, and deleting artwork are the same as those for displays. For more information about administering displays, see the following sections:

- “Granting and Revoking Access to Displays” – 6 – 19
- “Renaming Displays” – 6 – 20
- “Deleting Displays” – 6 – 21

You can also export artwork using PL/SQL. The following example exports the contents of a layer to a CGM file:

```plsql
PROCEDURE export_the_drawing IS
    the_layer   OG_OBJECT;
BEGIN
    the_layer:=OG_GET_LAYER('layer0');
    OG_EXPORT_DRAWING('my_draw', OG_FILESYSTEM, OG_CGM16_DFORMAT, the_layer);
END;
```


**Importing Displays**

You can import a display that is stored on the file system or in a database into your current display. This is a way to copy the contents of one display into another existing display and effectively combine the two displays.

To import a display:

1. Choose **Edit** —> **Import** —> **Display** to display the Import Display dialog.
2. Specify the name and location of the display, whether or not you want the active layers from each display merged or not, and how to resolve naming conflicts.
3. Choose the Import button to accept the dialog and import the display.
Using Layers in Displays

You can divide the display layout into multiple surfaces called layers. Every display’s layout is composed of one or more distinct layers. Layers are useful for hiding objects you want to show only under certain conditions. For example, you can place a button on one layer that a user can click on to display an object on another layer. Similarly, you may want to hide a layer so that you can display its objects in a window other than the Main Layout window.

The active layer is your working layer, and is always showing. However, you can create multiple layers to show along with the active layer, and you can also hide multiple layers. When you create a new display, it contains one layer called layer0. Since this is the only layer, it’s also the active layer by default.

Using PL/SQL, you can manipulate existing layers at runtime. The active layer is the only layer that end-users can interact with, but you can change the active layer using PL/SQL, depending on where you want user input. You can also hide, show, and change the stacking order of layers, to wholly or partially obscure individual objects.

For example, your layout may appear as a single surface:

But it may actually be composed of multiple layers:
Using Layers in Displays

You can divide the display layout into multiple surfaces called layers. Every display's layout is composed of one or more distinct layers. Layers are useful for hiding objects you want to show only under certain conditions. For example, you can place a button on one layer that a user can click on to display an object on another layer. Similarly, you may want to hide a layer so that you can display its objects in a window other than the Main Layout window.

The active layer is your working layer, and is always showing. However, you can create multiple layers to show along with the active layer, and you can also hide multiple layers. When you create a new display, it contains one layer called layer0. Since this is the only layer, it's also the active layer by default.

Using PL/SQL, you can manipulate existing layers at runtime. The active layer is the only layer that end-users can interact with, but you can change the active layer using PL/SQL, depending on where you want user input. You can also hide, show, and change the stacking order of layers, to wholly or partially obscure individual objects.

For example, your layout may appear as a single surface:

But it may actually be composed of multiple layers:
You can set any number of layers to be showing or hidden at any one time. The active layer must always be showing.

You can perform all actions on layers through the Layer Settings dialog. To display this dialog at any time, choose Tools—>Layers from the menu. Many actions can also be performed using the Object Navigator.

Creating New Layers

To create a new layer in the layout:

1. Select the layer object in the Object Navigator.

2. Choose the Create button, or choose Navigator—>Create to display the Layer Settings dialog.

   Alternatively, you can choose Tools—>Layers from the menu.

3. Choose the New button.

   In the Existing Layers list, a new layer appears called layerx, where x is an instance number that increments by one each time a new layer is created. It also becomes the active layer.

4. Choose the OK button to confirm the changes and close the Layer Settings dialog.

   Note: If the layout is divided into multiple views, the new layer is added in all views. However, it becomes the active layer only in the active view.

Renaming Layers

You can rename a layer either in the Layer Settings dialog box, or directly in the Object Navigator.
Creating New Layers

You can set any number of layers to be showing or hidden at any one time. The active layer must always be showing.

You can perform all actions on layers through the Layer Settings dialog. To display this dialog at any time, choose Tools —> Layers from the menu. Many actions can also be performed using the Object Navigator.

To create a new layer in the layout:

1. Select the layer object in the Object Navigator.
2. Choose the Create button, or choose Navigator —> Create to display the Layer Settings dialog. Alternatively, you can choose Tools —> Layers from the menu.
3. Choose the New button. In the Existing Layers list, a new layer appears called layerx, where x is an instance number that increments by one each time a new layer is created. It also becomes the active layer.
4. Choose the OK button to confirm the changes and close the Layer Settings dialog.

Note: If the layout is divided into multiple views, the new layer is added in all views. However, it becomes the active layer only in the active view.

You can rename a layer either in the Layer Settings dialog box, or directly in the Object Navigator.
**Note:** If you rename a layer that you have referenced in a PL/SQL program unit, you must change all references to reflect the new name.

To rename a layer in the Object Navigator:

1. Select the layer object you wish to rename.
2. Move the mouse pointer over the layer name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.
   
   **Suggestion:** If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To rename a layer in the Layer Settings dialog box:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
   
   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose **Tools—>Properties** from the menu.
2. Select the layer from the Existing Layers list.
   
   The layer appears in the Rename field.
3. Type a new name for the layer in the Rename field.
4. Choose the Rename button.
   
   In the Existing Layers list, the new name replaces the old name.
5. Choose the OK button to confirm the changes and close the Layer Settings dialog.

**Activating Layers**

To work on a layer, you need to make it the active layer. Only one layer can be active at a time. To activate a layer:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
   
   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose **Tools—>Properties** from the menu.
2. Select a layer from the Existing Layers list.
3. Choose the Activate button.
   
   The layer is listed in the Active Layer field. If the layer was previously hidden, it will now be showing.
Activating Layers

Managing Displays

Note:
If you rename a layer that you have referenced in a PL/SQL program unit, you must change all references to reflect the new name.

To rename a layer in the Object Navigator:
1. Select the layer object you wish to rename.
2. Move the mouse pointer over the layer name.
The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.
Suggestion:
If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To rename a layer in the Layer Settings dialog box:
1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools —> Properties from the menu.
2. Select the layer from the Existing Layers list.
The layer appears in the Rename field.
3. Type a new name for the layer in the Rename field.
4. Choose the Rename button.
The layer is listed in the Active Layer field. If the layer was previously hidden, it will now be showing.
5. Choose the OK button to confirm the changes and close the Layer Settings dialog.

To work on a layer, you need to make it the active layer. Only one layer can be active at a time. To activate a layer:
1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools —> Properties from the menu.
2. Select a layer from the Existing Layers list.
3. Choose the Activate button.
The layer is listed in the Active Layer field. If the layer was previously hidden, it will now be showing.
4. Choose the OK button to confirm the changes and close the Layer Settings dialog.

You can also activate a layer using PL/SQL. The following example activates the specified layer in the specified window:

```plsql
PROCEDURE activate_a_layer(layer_num IN NUMBER, the_window IN og_window) IS
    layer_name VARCHAR2(6);
    the_layer OG_LAYER;
BEGIN
    layer_name:='layer'||TO_CHAR(layer_num);
    the_layer:=OG_GET_LAYER(layer_name);
    OG_ACTIVATE_LAYER(the_layer, the_window);
END;
```


**Hiding and Showing Layers**

You can hide or show any number of layers. You cannot hide the active layer. If you wish to hide a layer that is currently active, you must first activate a different layer.

To hide or show a layer:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.

   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools—>Properties from the menu.

2. Select the layer from the Existing Layers list.

3. Choose the Hide/Show button.
   
   - If the layer was hidden, a plus (+) sign appears in front of the layer, indicating that it’s now showing.
   
   - If the layer was showing, a minus (–) sign appears in front of the layer, indicating that it’s now hidden.

**Note:** You can hide any layer that is showing except the active layer. To hide the layer that is currently active, first activate a different layer.

4. Choose the OK button to confirm the changes and close the Layer Settings dialog.
Choose the OK button to confirm the changes and close the Layer Settings dialog.

You can also activate a layer using PL/SQL. The following example activates the specified layer in the specified window:

```sql
PROCEDURE activate_a_layer(layer_num IN NUMBER, the_window IN og_window) IS
  layer_name   VARCHAR2(6);
  the_layer    OG_LAYER;
BEGIN
  layer_name:='layer'||TO_CHAR(layer_num);
  the_layer:=OG_GET_LAYER(layer_name);
  OG_ACTIVATE_LAYER(the_layer, the_window);
END;```


You can hide or show any number of layers. You cannot hide the active layer. If you wish to hide a layer that is currently active, you must first activate a different layer.

To hide or show a layer:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools —> Properties from the menu.
2. Select the layer from the Existing Layers list.
3. Choose the Hide/Show button.
   – If the layer was hidden, a plus (+) sign appears in front of the layer, indicating that it’s now showing.
   – If the layer was showing, a minus (–) sign appears in front of the layer, indicating that it’s now hidden.

Note: You can hide any layer that is showing except the active layer. To hide the layer that is currently active, first activate a different layer.

4. Choose the OK button to confirm the changes and close the Layer Settings dialog.
You can also hide and show layers using PL/SQL. The first example hides the specified layer and the second example shows the specified layer:

```plsql
PROCEDURE make_layer1_invis(the_window) IS
  my_layer   OG_LAYER;
BEGIN
  my_layer:=OG_GET_LAYER('layer1');
  OG_HIDE_LAYER(my_layer, the_window);
END;

PROCEDURE make_layer_visible(the_window) IS
  my_layer   OG_LAYER;
BEGIN
  my_layer:=OG_GET_LAYER('layer1');
  OG_SHOW_LAYER(my_layer, the_window);
END;
```


### Re-ordering Layers

To change the order in which layers are stacked:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.

   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose **Tools—>Properties** from the menu.

2. Select the layer you want to place in a different order from the Existing Layers list.

3. Choose either the Up (up a layer), Down (down a layer), Top (to the top of the stack), or Bottom (to the bottom of the stack) buttons.

   The layer moves to its new position in the stack.

4. Choose the OK button to confirm the changes and close the Layer Settings dialog.

### Deleting Layers

You can delete any non-active layers, whether hiding or showing. Deleting a layer also deletes all objects on that layer. You cannot delete the active layer. If you wish to delete the active layer, you must first activate another layer.
You can also hide and show layers using PL/SQL. The first example hides the specified layer and the second example shows the specified layer:

```
PROCEDURE make_layer1_invis(the_window) IS
my_layer   OG_LAYER;
BEGIN
my_layer:=OG_GET_LAYER('layer1');
OG_HIDE_LAYER(my_layer, the_window);
END;

PROCEDURE make_layer_visible(the_window) IS
my_layer   OG_LAYER;
BEGIN
my_layer:=OG_GET_LAYER('layer1');
OG_SHOW_LAYER(my_layer, the_window);
END;
```


To change the order in which layers are stacked:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.

   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools —> Properties from the menu.

2. Select the layer you want to place in a different order from the Existing Layers list.

3. Choose either the Up (up a layer), Down (down a layer), Top (to the top of the stack), or Bottom (to the bottom of the stack) buttons. The layer moves to its new position in the stack.

4. Choose the OK button to confirm the changes and close the Layer Settings dialog.

You can delete any non-active layers, whether hiding or showing. Deleting a layer also deletes all objects on that layer. You cannot delete the active layer. If you wish to delete the active layer, you must first activate another layer.
To delete a layer and all of its objects:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools—>Properties from the menu.

2. Select the layer you want to delete from the Existing Layers list.

3. Choose the Delete button.

4. Choose the Yes button to confirm the alert that appears.
   The layer is removed from the Existing Layers List.

5. Choose the OK button to confirm the changes and close the Layer Settings dialog.
   The layer and its objects are deleted from the display.

Working with Sounds

If your system supports sound, you can make audio recordings to use in your displays. Sounds are useful for alerting end-users to specific events at runtime. For example, you may want to record an alarm to be sounded when data in a chart falls below a certain threshold value. Playing sounds at runtime must be done programmatically, using PL/SQL.

You can either record a sound in the Designer, or import an existing sound from the file system or the database.

Recording Sounds

To record a sound in the Designer:

1. In the Object Navigator, select the Sounds subnode and choose the Create button or Navigator—>Create.

   **Suggestion:** If no sounds exist in the current display, double-click on the Sounds node in the Navigator to display a new sound in the Sound Player dialog box.

   A new sound (initially named soundx) is created and the Sound Player dialog box appears.

2. To record the new sound, choose the Record button.

3. When you are done recording, choose the Stop button.

4. Choose the OK button to accept the sound and close the dialog.
To delete a layer and all of its objects:

1. Double-click on the layer object in the Navigator to display the Layer Settings dialog.
   Alternatively, you can select the layer object in the Navigator and choose the Properties command from the pop-up menu, or choose Tools —> Properties from the menu.

2. Select the layer you want to delete from the Existing Layers list.

3. Choose the Delete button.

4. Choose the Yes button to confirm the alert that appears. The layer is removed from the Existing Layers List.

5. Choose the OK button to confirm the changes and close the Layer Settings dialog. The layer and its objects are deleted from the display.

Working with Sounds

If your system supports sound, you can make audio recordings to use in your displays. Sounds are useful for alerting end-users to specific events at runtime. For example, you may want to record an alarm to be sounded when data in a chart falls below a certain threshold value. Playing sounds at runtime must be done programmatically, using PL/SQL.

You can either record a sound in the Designer, or import an existing sound from the file system or the database.

To record a sound in the Designer:

1. In the Object Navigator, select the Sounds subnode and choose the Create button or Navigator —> Create.

   Suggestion: If no sounds exist in the current display, double-click on the Sounds node in the Navigator to display a new sound in the Sound Player dialog box.

2. To record the new sound, choose the Record button.

3. When you are done recording, choose the Stop button.

4. Choose the OK button to accept the sound and close the dialog.
The sound is added to your display. You can access the sound through the PL/SQL built-in procedure OG_PLAY_SOUND.

You can also prompt the user to record a sound, using PL/SQL. The following example shows the sound dialog box in the Runtime display, and allows the user to record a new sound:

```plsql
PROCEDURE record_warning IS
  warn_sound   OG_SOUND;
BEGIN
  warn_sound:=OG_GET_SOUND('warning');
  IF NOT OG_ISNULL(warn_sound) THEN
    OG_RECORD_SOUND(warn_sound);
  END IF;
END;
```


### Importing a Sound

You can import an existing sound file from either the file system or the database.

1. Choose **Edit**—>**Import**—>**Sound** to display the Import Sound dialog.
2. Select either the File System or Database radio button to indicate the source of the file.
3. Type the name of the file in the File field, or select the Browse button to display a list of files to choose from.
4. Select the appropriate sound file format from the Format drop-down list.
5. Choose the OK button to accept the dialog and import the sound.

The sound is added to your display. You can access the sound through the PL/SQL built-in procedure OG_PLAY_SOUND.

### Playing and Editing Sounds

To play or edit a sound:

1. Double-click on an existing sound to display the Sound Player dialog box.
2. Play or record over the existing sound by choosing the Play or Record buttons, respectively.

You can also use the Forward and Rewind buttons, or the slide bar to find a position in the sound file.
The sound is added to your display. You can access the sound through the PL/SQL built-in procedure OG_PLAY_SOUND.


You can import an existing sound file from either the file system or the database.

1. Choose Edit —> Import —> Sound to display the Import Sound dialog.
2. Select either the File System or Database radio button to indicate the source of the file.
3. Type the name of the file in the File field, or select the Browse button to display a list of files to choose from.
4. Select the appropriate sound file format from the Format drop-down list.
5. Choose the OK button to accept the dialog and import the sound.

The sound is added to your display. You can access the sound through the PL/SQL built-in procedure OG_PLAY_SOUND.

To play or edit a sound:

1. Double-click on an existing sound to display the Sound Player dialog box.
2. Play or record over the existing sound by choosing the Play or Record buttons, respectively.
3. You can also use the Forward and Rewind buttons, or the slide bar to find a position in the sound file.
3. Choose the OK button to close the Sound Player dialog box.

You can also play a sound using PL/SQL built-in procedure OG_PLAY SOUND, and the function OG_GET_SOUND. The following example plays the specified sound through the sound output device specified in your preferences:

```plsql
PROCEDURE warning IS
    the_sound   OG_SOUND;
BEGIN
    the_sound:=OG_GET_SOUND('warning_snd');
    OG_PLAY_SOUND(the_sound);
END;
```


**Exporting Sounds**

You can export sounds that you have created in Graphics and save them on the file system, or in the database.

To export a sound:

1. Select the sound object in the Navigator.
2. Choose Edit—>Export—>Sound to display the Export Sound dialog.
3. Specify the desired options such as filename and location, the file format and compression to export the sound in, etc.
4. Choose the OK button to confirm the information and close the Export dialog.

The artwork is exported to the location you specified.

**Managing Sounds**

Sounds that you have created in the Designer or imported into the display are considered to be internal to the display. Sounds that are stored in the file system or the database are considered external to the display. The location of the sound determines what methods you use to grant and revoke access, rename, or delete the sound.

**Internal Sounds**

This section describes how to rename and delete sounds within the display.
3. Choose the OK button to close the Sound Player dialog box. You can also play a sound using PL/SQL built-in procedure `OG_PLAY_SOUND`, and the function `OG_GET_SOUND`. The following example plays the specified sound through the sound output device specified in your preferences:

```plsql
PROCEDURE warning IS
    the_sound   OG_SOUND;
BEGIN
    the_sound:=OG_GET_SOUND('warning_snd');
    OG_PLAY_SOUND(the_sound);
END;
```


You can export sounds that you have created in Graphics and save them on the file system, or in the database.

1. Select the sound object in the Navigator.
2. Choose Edit —> Export —> Sound to display the Export Sound dialog.
3. Specify the desired options such as filename and location, the file format and compression to export the sound in, etc.
4. Choose the OK button to confirm the information and close the Export dialog.

The artwork is exported to the location you specified.

Sounds that you have created in the Designer or imported into the display are considered to be internal to the display. Sounds that are stored in the file system or the database are considered external to the display. The location of the sound determines what methods you use to grant and revoke access, rename, or delete the sound.

This section describes how to rename and delete sounds within the display.
To rename a sound in the Object Navigator:

**Note:** If you rename a sound that you have referenced in a PL/SQL program unit, you must change all references to reflect the new name.

1. Select the sound object you wish to rename.
2. Move the mouse pointer over the sound name.
   - The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.
   - **Suggestion:** If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To delete a sound from the Object Navigator:

1. Select the sound object you wish to delete.
2. Choose the Delete button, or **Navigator —> Delete**.
3. Choose Yes to accept the delete confirmation that appears.
   - The sound is removed from the display.

External Sounds

The procedures for granting and revoking access, renaming, and deleting sounds in the file system or a database are the same as those for displays.

For more information about administering displays, see the following sections:

- “Granting and Revoking Access to Displays” — 6 – 19
- “Renaming Displays” — 6 – 20
- “Deleting Displays” — 6 – 21

If you have created a sound in the Designer, or imported a sound into your display, you can rename and delete the sound object in the Navigator.

**Saving Displays**

When you save a display for the first time, you need to specify whether to store it in the file system or the database.

**Saving Displays in the File System**

To save a display in the file system:

1. Choose **File —> Save As —> File System** to display your system file dialog.
To rename a sound in the Object Navigator:

1. Select the sound object you wish to rename.
2. Move the mouse pointer over the sound name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode. Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To delete a sound from the Object Navigator:

1. Select the sound object you wish to delete.
2. Choose the Delete button, or Navigator —> Delete.
3. Choose Yes to accept the delete confirmation that appears. The sound is removed from the display.

The procedures for granting and revoking access, renaming, and deleting sounds in the file system or a database are the same as those for displays.

For more information about administering displays, see the following sections:
- “Granting and Revoking Access to Displays” — 6 – 19
- “Renaming Displays” — 6 – 20
- “Deleting Displays” — 6 – 21

If you have created a sound in the Designer, or imported a sound into your display, you can rename and delete the sound object in the Navigator.

Saving Displays

When you save a display for the first time, you need to specify whether to store it in the file system or the database.

To save a display in the file system:

1. Choose File —> Save As —> File System to display your system file dialog.
2. Specify a unique filename for the display.
   You can keep the Graphics default filename or replace it with a
   unique name.
3. Accept the dialog box to save the display.
   The contents of the Layout editor and other windows within the
display are saved as you left them.

---

**Saving Displays in the Database**

You must be connected to the database, and have the appropriate access
to save displays in the database. In addition, your database must have
the Graphics tables set up. Check with your database administrator for
more information.

To save a display in the database:

1. Choose **File**—>**Save As**—>**Database** to display the Save To
   Database dialog box.
2. Type a unique name for the display in the Name field.
   You can keep the Graphics default module name or replace it with a
   unique name. Graphics will automatically prefix the module name
   with your database username (i.e. SCOTT.DISP1).
3. Choose the Save button to save the display.
   The contents of the Layout editor and other windows within the
display are saved as you left them.

---

**Saving Changes to Displays**

While you’re working, it’s a good idea to save your display periodically
to avoid losing data should your system fail. To save changes to an
open display that you’ve already saved:

1. Make sure the display you want to save is selected in the Object
   Navigator.
2. Choose the Save command from the pop-up menu, or choose the
   Save button, or choose **File**—>**Save** to save your display.
   You can access the Save button in both the Navigator and Layout
   editor Tool bars.
   Your display is saved under the same name and location as you
   previously specified.

---

**Discarding Changes to Displays**

To discard changes you’ve made to a display since the last time you
saved it:

- Choose **File**—>**Revert** to discard the most recent changes.
  Your display is restored to the previously saved version.
Saving Displays in the Database

Saving Changes to Displays

Discarding Changes to Displays

6 – 16 Graphics Developer's Guide

2. Specify a unique filename for the display. You can keep the Graphics default filename or replace it with a unique name.

3. Accept the dialog box to save the display. The contents of the Layout editor and other windows within the display are saved as you left them.

You must be connected to the database, and have the appropriate access to save displays in the database. In addition, your database must have the Graphics tables set up. Check with your database administrator for more information.

To save a display in the database:

1. Choose File —> Save As —> Database to display the Save To Database dialog box.

2. Type a unique name for the display in the Name field. You can keep the Graphics default module name or replace it with a unique name. Graphics will automatically prefix the module name with your database username (i.e. SCOTT.DISP1).

3. Choose the Save button to save the display. The contents of the Layout editor and other windows within the display are saved as you left them.

While you're working, it's a good idea to save your display periodically to avoid losing data should your system fail. To save changes to an open display that you've already saved:

1. Make sure the display you want to save is selected in the Object Navigator.

2. Choose the Save command from the pop-up menu, or choose the Save button, or choose File —> Save to save your display.

You can access the Save button in both the Navigator and Layout editor Tool bars.

Your display is saved under the same name and location as you previously specified.

To discard changes you've made to a display since the last time you saved it:

Choose File —> Revert to discard the most recent changes. Your display is restored to the previously saved version.
Closing Displays

To close a display:

- Choose **File—>Close** to close your display.
  
The display window closes leaving the Object Navigator, the Designer main menu and any other open displays on your screen.

**Note:** If you try to close a display containing changes that haven’t been saved, Graphics will prompt you to save the display first.

Ending Your Session

To end your Designer session:

- Choose **File—>Exit** to exit from Graphics Designer.
  
The Graphics Designer session ends and you return to your operating system.

**Note:** If you try to end your Designer session before saving your displays, Graphics will prompt you to do so first.

Testing Displays

In the Designer, you can test the execution of your display by accessing the Graphics Debugger. When you select a display to run in the debugger, Graphics runs the stored version of the display. Therefore, if your display is open in the Designer, be sure to save any recent changes you’ve made before running it.

For more information, see “Debugging in Graphics” on page 14 – 1.

Running an Open Display

If the display you wish to run is open, you can run it from the menu, or from the Object Navigator.

To run an open display:

1. Select the display object in the Navigator.

2. Choose the Run command from the pop-up menu, or choose the Run button, or choose **File—>Run**.
   
   You can access the Run button in both the Navigator and Layout editor Tool bars.

3. The display is opened in the Graphics Debugger Main Layout window, which simulates runtime execution of your display.
   
   All other windows, except the Object Navigator, are minimized.
Ending Your Session

Running an Open Display

Closing Displays

To close a display:

Choose File —> Close to close your display.

The display window closes leaving the Object Navigator, the Designer main menu and any other open displays on your screen.

Note: If you try to close a display containing changes that haven't been saved, Graphics will prompt you to save the display first.

To end your Designer session:

Choose File —> Exit to exit from Graphics Designer.

The Graphics Designer session ends and you return to your operating system.

Note: If you try to end your Designer session before saving your displays, Graphics will prompt you to do so first.

Testing Displays

In the Designer, you can test the execution of your display by accessing the Graphics Debugger. When you select a display to run in the debugger, Graphics runs the stored version of the display. Therefore, if your display is open in the Designer, be sure to save any recent changes you've made before running it.

For more information, see "Debugging in Graphics" on page 14 – 1.

If the display you wish to run is open, you can run it from the menu, or from the Object Navigator.

To run an open display:

1. Select the display object in the Navigator.
2. Choose the Run command from the pop-up menu, or choose the Run button, or choose File —> Run.

You can access the Run button in both the Navigator and Layout editor Tool bars.

3. The display is opened in the Graphics Debugger Main Layout window, which simulates runtime execution of your display. All other windows, except the Object Navigator, are minimized.
Running a Closed Display

You can run any closed displays that are saved on the file system or in the database.

To run a closed display:

1. Choose **File**—➤**Run** to display the filter dialog.

2. Select the appropriate settings for the filter dialog and then choose the OK button to open either the Open from Database or your operating system file dialog.

3. Select a display from the resulting list.

4. Accept the dialog to close it and run the selected display.

5. The display is opened in the Graphics Debugger Main Layout window, which simulates runtime execution of your display.

6. Choose **File**—➤**Close** to close the debugger when you are done.

Generating Displays

When you generate a runtime version of a display, it contains only the elements needed to run it with the Graphics Runtime or Batch executables. For example, a generated display does not contain PL/SQL source code.

Generating Displays to the File System

To generate a runtime version of a display stored in the file system:

1. Choose **File**—➤**Administration**—➤**Generate**—➤**File System** to display your system file dialog.

2. Type the name of the Runtime display file or accept the default name.

   Graphics Runtime displays use the file extension .ogr by default.

3. Accept the dialog box to generate the display.

   Graphics generates a runtime version of the display.
Note: If you change a display in the Designer while it is open in the debugger, the changes will not be visible in the debugger until you save the display and run it again. If you edit PL/SQL program units in the debugger, you can incorporate your changes back into the Designer using Edit —> Synchronize Program Units.

4. Choose File —> Close to close the debugger when you are done.

You can run any closed displays that are saved on the file system or in the database.

To run a closed display:
1. Choose File —> Run to display the filter dialog.
2. Select the appropriate settings for the filter dialog and then choose the OK button to open either the Open from Database or your operating system file dialog.
3. Select a display from the resulting list.
4. Accept the dialog to close it and run the selected display.
5. The display is opened in the Graphics Debugger Main Layout window, which simulates runtime execution of your display.
6. Choose File —> Close to close the debugger when you are done.

Generating Displays

When you generate a runtime version of a display, it contains only the elements needed to run it with the Graphics Runtime or Batch executables. For example, a generated display does not contain PL/SQL source code.

To generate a runtime version of a display stored in the file system:
2. Type the name of the Runtime display file or accept the default name.
3. Graphics Runtime displays use the file extension .ogr by default.
3. Accept the dialog box to generate the display.

Graphics generates a runtime version of the display.
Generating Displays from the Database

To generate a runtime version of a display stored in the database:
1. Choose **File**—>**Administration**—>**Generate**—>**Database** to display the Save to Database dialog.
2. Specify the name of the display.
3. Choose the OK button to confirm and generate the display.

Graphics generates a runtime version of the display to the database.

Printing Displays

To print a hard copy of your display:
1. Choose **File**—>**Print** to display your system’s print dialog box.
2. Specify print options for your system.
3. Accept the dialog box to begin printing.

**Note:** If your display contains multiple layers, only the layers showing are printed.

Setting Up Pages

To set page size, orientation, and other output-related information:
1. Choose **File**—>**Print Setup** to display your system’s print setup dialog.
2. Specify the print settings for your system.
3. Accept the dialog box.

The settings remain in effect for the current Designer session.

Granting and Revoking Access to Displays

After you have created a display, you may want to protect it from update by other users, or grant access to other database users so they can modify it.

Granting and Revoking Access in the File System

Some operating systems permit users to grant read, write, and execute permissions on individual files. For more information about how to set file permissions on your operating system, see your operating system documentation or check with your system administrator.
Generating Displays from the Database

Setting Up Pages

Granting and Revoking Access in the File System

To generate a runtime version of a display stored in the database:
1. Choose File —> Administration —> Generate —> Database to display the Save to Database dialog.
2. Specify the name of the display.
3. Choose the OK button to confirm and generate the display.

Graphics generates a runtime version of the display to the database.

Printing Displays

To print a hard copy of your display:
1. Choose File —> Print to display your system's print dialog box.
2. Specify print options for your system.
3. Accept the dialog box to begin printing.

Note: If your display contains multiple layers, only the layers showing are printed.

To set page size, orientation, and other output-related information:
1. Choose File —> Print Setup to display your system's print setup dialog.
2. Specify the print settings for your system.
3. Accept the dialog box. The settings remain in effect for the current Designer session.

Granting and Revoking Access to Displays

After you have created a display, you may want to protect it from update by other users, or grant access to other database users so they can modify it.

Some operating systems permit users to grant read, write, and execute permissions on individual files. For more information about how to set file permissions on your operating system, see your operating system documentation or check with your system administrator.
Granting and Revoking Access in the Database

You can grant another user access (read/write/execute privileges) to a display you’ve created and stored in the database. However, the user must also have access to the Graphics database tables. For more information, consult your database administrator.

To grant access to a display stored in the database:

1. Choose File—>Administration—>Access—>Display to display the Module Access in Database dialog box.
   This dialog contains a list of all displays to which you have access that are stored in the database.
2. Select a display from the list.
3. Choose the OK button to display the Module Access dialog.
   This dialog contains a list of users who have access to the display.
4. Type the database userid of the person to whom you want to grant access in the User field.
5. Choose the Add button to add the name to the Access list.
6. Select the username of the person from whom you wish to revoke access in the Access list.
7. Choose the Remove button to remove the username from the Access list.
8. Choose the OK button to accept your changes and close the Module Access dialog.
   Access privileges are updated for your display.

Renaming Displays

Renaming Displays in the File System

You can use your operating system to rename a display that is stored as a file. You must have the appropriate file permissions to be able to rename the file.

See your operating system documentation for the correct method of renaming files.

Renaming Displays in the Database

You can rename a display stored in the database if you created it or were granted access to it.
Granting and Revoking Access in the Database

Renaming Displays in the File System

You can grant another user access (read/write/execute privileges) to a display you've created and stored in the database. However, the user must also have access to the Graphics database tables. For more information, consult your database administrator.

To grant access to a display stored in the database:

1. Choose File —> Administration —> Access —> Display to display the Module Access in Database dialog box. This dialog contains a list of all displays to which you have access that are stored in the database.
2. Select a display from the list.
3. Choose the OK button to display the Module Access dialog. This dialog contains a list of users who have access to the display.
4. Type the database userid of the person to whom you want to grant access in the User field.
5. Choose the Add button to add the name to the Access list.
6. Select the username of the person from whom you wish to revoke access in the Access list.
7. Choose the Remove button to remove the username from the Access list.
8. Choose the OK button to accept your changes and close the Module Access dialog. Access privileges are updated for your display.

Renaming Displays

You can use your operating system to rename a display that is stored as a file. You must have the appropriate file permissions to be able to rename the file.

See your operating system documentation for the correct method of renaming files.

You can rename a display stored in the database if you created it or were granted access to it.
To rename a display stored in the database:

1. Choose **File**—>**Administration**—>**Rename**—>**Display** to display the Rename dialog box.
2. Specify the old name and a new name for the display.
3. Choose the Rename button to apply the new name.

Deleting Displays

Deleting Displays in the Object Navigator

You can delete an open display in the Object Navigator regardless of where it is stored. You must have the appropriate access to the display to delete it. If you do not have file write permission or database access, you cannot delete the display.

To delete a display in the Object Navigator:

1. Select the display object in the Object Navigator.
2. Choose the Delete button, or choose **Navigator**—>**Delete**.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.
   The display is deleted.

Deleting Displays from the File System

You can use your operating system to delete a display that is stored as a file. You must have the appropriate file permissions to delete the display.

See your operating system documentation for the correct method of deleting files.

Deleting Displays from the Database

You can delete a display stored in the database if you created or were granted access to it.

To delete a display stored in the database:

1. Choose **File**—>**Administration**—>**Delete**—>**Display** to display the Delete From Database dialog.
2. Specify the name of the display.
3. Choose the Delete button to delete the display from the database.
Deleting Displays in the Object Navigator

Deleting Displays from the File System

Deleting Displays from the Database

To rename a display stored in the database:

1. Choose File —> Administration —> Rename —> Display to display the Rename dialog box.
2. Specify the old name and a new name for the display.
3. Choose the Rename button to apply the new name.

Deleting Displays

You can delete an open display in the Object Navigator regardless of where it is stored. You must have the appropriate access to the display to delete it. If you do not have file write permission or database access, you cannot delete the display.

To delete a display in the Object Navigator:

1. Select the display object in the Object Navigator.
2. Choose the Delete button, or choose Navigator —> Delete.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.
   The display is deleted.

You can use your operating system to delete a display that is stored as a file. You must have the appropriate file permissions to delete the display.

See your operating system documentation for the correct method of deleting files.

You can delete a display stored in the database if you created or were granted access to it.

To delete a display stored in the database:

1. Choose File —> Administration —> Delete —> Display to display the Delete From Database dialog.
2. Specify the name of the display.
3. Choose the Delete button to delete the display from the database.
Using Queries

This chapter discusses how to make use of queries in Graphics. Queries are SQL SELECT statements or files that you can use to retrieve data for a display. In most cases, you’ll use queries to retrieve data for charts, but you can also execute queries programmatically and use the data to create and modify objects.

The following topics are covered in this chapter:

• connecting to the database – 7 – 2
• notes on queries – 7 – 2
• creating queries – 7 – 3
• using external SQL queries – 7 – 7
• creating custom queries – 7 – 8
• accessing queries – 7 – 11
• modifying queries – 7 – 12
• exporting queries to text files – 7 – 13
• executing queries – 7 – 13
• renaming queries – 7 – 15
• deleting queries – 7 – 16
• specifying date formats for file queries – 7 – 16
• using query filter functions – 7 – 17
This chapter discusses how to make use of queries in Graphics. Queries are SQL SELECT statements or files that you can use to retrieve data for a display. In most cases, you'll use queries to retrieve data for charts, but you can also execute queries programmatically and use the data to create and modify objects.

The following topics are covered in this chapter:

• connecting to the database – 7 – 2
• notes on queries – 7 – 2
• creating queries – 7 – 3
• using external SQL queries – 7 – 7
• creating custom queries – 7 – 8
• accessing queries – 7 – 11
• modifying queries – 7 – 12
• exporting queries to text files – 7 – 13
• executing queries – 7 – 13
• renaming queries – 7 – 15
• deleting queries – 7 – 16
• specifying date formats for file queries – 7 – 16
• using query filter functions – 7 – 17
Connecting to the Database

Before you can query a database, you need to connect to it. Then you can create a SQL SELECT statement to retrieve the data.

To connect to the database:

1. Choose File—>Connect to show the Connect dialog box.

2. Type your username and password. (Your password will not appear when you type.) If you’re not sure which username and password to use, consult your database administrator.

   Leave the Database field blank, unless you’re connecting to a remote database. If so, you’ll need to enter the SQL*Net communications protocol, database node, and database name, separated by colons. For example:

   t:boston:payroll

   You can specify a default database connection string to use for all connections. See the Developer/2000 Installation Guide for more information, or consult your system administrator.

3. Choose the Connect button to log on.

   If you successfully log on to the database, the Connect dialog box closes. If you get an error message, consult your database administrator and confirm that you’re using the correct logon ID and database specification.

Notes on Queries

Naming Queries

Regardless of whether you create your queries at chart creation time, or separately, it is a good idea to give each query a unique and representative name. Each query you create is given a default name of queryn by Graphics. Assigning indicative names can help you easily locate the query in the Object Navigator, or pick a query from a drop-down list when creating a chart.

Custom Axis Labels

If you are creating axis charts, the X and Y axis labels are derived from the column names assigned to the table in your query. You can either customize those axis labels using the Axis property sheet, or you can specify a column name alias in your query. For example:

```sql
select warehouse_id "Warehouse ID",
amount_in_stock "Inventory Levels"
from s_inventory
```
Before you can query a database, you need to connect to it. Then you can create a SQL SELECT statement to retrieve the data.

To connect to the database:

1. Choose File —> Connect to show the Connect dialog box.
2. Type your username and password. (Your password will not appear when you type.) If you're not sure which username and password to use, consult your database administrator. Leave the Database field blank, unless you're connecting to a remote database. If so, you'll need to enter the SQL*Net communications protocol, database node, and database name, separated by colons. For example:
   t:boston:payroll
   You can specify a default database connection string to use for all connections. See the Developer/2000 Installation Guide for more information, or consult your system administrator.
3. Choose the Connect button to log on.

If you successfully log on to the database, the Connect dialog box closes. If you get an error message, consult your database administrator and confirm that you're using the correct logon ID and database specification.

Notes on Queries

Regardless of whether you create your queries at chart creation time, or separately, it is a good idea to give each query a unique and representative name. Each query you create is given a default name of queryn by Graphics. Assigning indicative names can help you easily locate the query in the Object Navigator, or pick a query from a drop-down list when creating a chart.

If you are creating axis charts, the X and Y axis labels are derived from the column names assigned to the table in your query. You can either customize those axis labels using the Axis property sheet, or you can specify a column name alias in your query. For example:

```
select warehouse_id "Warehouse ID",
amount_in_stock "Inventory Levels"
from s_inventory
```
If you plan to use chart templates, it’s a good idea to assign customized axis labels in your query as show above.

An Alternative to WHERE Clauses

If you plan to create multiple charts that use basically the same data source, you can create one generic query and use query filters to determine the data plotted. Query filters perform the same function as a WHERE clause.

However, query filters have performance advantages. If you create three charts that use the same basic query, but have different WHERE clause conditions set, each of those specific queries must be executed against the database server when the chart is updated.

If, instead, you create one generic query, and then define a query filter for each of the three charts, only that one query needs to be executed against the server. The query filter is performed on the client side, reducing network overhead and calls against the database.

Creating Queries

Options for Creating Queries

Graphics gives you several ways to create queries:

- create standalone queries that you can assign to charts later
- create queries when you create a chart
- create queries programmatically

Creating queries at chart creation time combines two actions into one process so you can quickly create a chart without having to define your query separately.

You can create three types of queries, depending on your data source. If your data source is a database, you’ll use a SQL SELECT statement. If your data source is a file, such as a spreadsheet, you’ll specify the entire file as the query. Further, you can create custom queries using a combination of SQL statements and PL/SQL code.

When you execute a query, Graphics retrieves the specified set of data columns and their rows, and places them in the Data property page in the Query property sheet. The Data property page always contains the data retrieved from the current query. When you create a chart, Graphics uses the data from the current query unless you specify otherwise.
If you plan to use chart templates, it's a good idea to assign customized axis labels in your query as shown above.

If you plan to create multiple charts that use basically the same data source, you can create one generic query and use query filters to determine the data plotted. Query filters perform the same function as a WHERE clause.

However, query filters have performance advantages. If you create three charts that use the same basic query, but have different WHERE clause conditions set, each of those specific queries must be executed against the database server when the chart is updated.

If, instead, you create one generic query, and then define a query filter for each of the three charts, only that one query needs to be executed against the server. The query filter is performed on the client side, reducing network overhead and calls against the database.

Creating Queries

Graphics gives you several ways to create queries:

- create standalone queries that you can assign to charts later
- create queries when you create a chart
- create queries programmatically

Creating queries at chart creation time combines two actions into one process so you can quickly create a chart without having to define your query separately.

You can create three types of queries, depending on your data source. If your data source is a database, you'll use a SQL SELECT statement. If your data source is a file, such as a spreadsheet, you'll specify the entire file as the query. Further, you can create custom queries using a combination of SQL statements and PL/SQL code.

When you execute a query, Graphics retrieves the specified set of data columns and their rows, and places them in the Data property page in the Query property sheet. The Data property page always contains the data retrieved from the current query. When you create a chart, Graphics uses the data from the current query unless you specify otherwise.
Creating Standalone Queries

If you wish to create your queries up front, and create your charts at a later time, use this method. With the Query property sheet, you can create multiple queries at one time.

Creating a SQL SELECT Statement

To create a SQL SELECT statement query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator—>Create to show the Query property sheet.

   Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

   Alternatively, you can choose Tools—>Queries to show the Query property sheet and then choose the New button to create a new query.

   The default name is queryx, where x is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name or type a new name in the Name field.

3. Type a SQL SELECT statement in the SQL Statement field.

   For more information about SQL SELECT statements, see the Oracle7 Server SQL Language Reference Manual.

4. Choose the Execute button to execute the query.

   The data appears in the Data property page.

5. Choose the OK button or the Close button to close the Query property sheet.

   The query remains the current query until you create or edit another one.

Specifying Files as Queries

To specify a file (SYLK, WKS, or PRN) as a query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator—>Create to show the Query property sheet.

   Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

   Alternatively, you can choose Tools—>Queries to show the Query property sheet, then choose the New button to create a new query.
Creating Standalone Queries

Creating a SQL SELECT Statement

Specifying Files as Queries

If you wish to create your queries up front, and create your charts at a later time, use this method. With the Query property sheet, you can create multiple queries at one time.

To create a SQL SELECT statement query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator — Create to show the Query property sheet.

Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

Alternatively, you can choose Tools — Queries to show the Query property sheet and then choose the New button to create a new query.

The default name is queryx, where x is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name or type a new name in the Name field.

3. Type a SQL SELECT statement in the SQL Statement field. For more information about SQL SELECT statements, see the Oracle7 Server SQL Language Reference Manual.

4. Choose the Execute button to execute the query. The data appears in the Data property page.

5. Choose the OK button or the Close button to close the Query property sheet. The query remains the current query until you create or edit another one.

To specify a file (SYLK, WKS, or PRN) as a query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator — Create to show the Query property sheet.

Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

Alternatively, you can choose Tools — Queries to show the Query property sheet, then choose the New button to create a new query.
Creating Queries at Chart Creation Time

The default name is \textit{queryx}, where \textit{x} is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name, or type a new name in the Name field.
3. Select a file type from the Query Type drop-down list.
4. Choose the Find button to show your system’s file dialog.
5. Select a valid file and accept the dialog box.
   Graphics places the directory path and filename in the File name field.
   \textbf{Note:} If your file contains dates, see the section “Setting Date Formats for File Queries” on page 7 – 16.
6. Choose the Execute button to execute the query.
   The data appears in the Data property page.
7. Choose the OK button or the Close button to close the Query property sheet.
   The query remains current until you create or edit another.

This section describes how to create a new query during the chart creation process. Use the instructions in the previous section for creating a specific type of query such as SQL, file, or custom.

After you create the new query with your chart, both objects are listed in the Object Navigator. You can edit the query’s properties as described in the following sections of this chapter.

To create a query during chart creation:

1. Choose \textbf{Chart—>Create Chart}.
   Alternatively, you can select the Chart tool in the Layout editor Tool palette and either click in the display area, or drag out a region for the chart.
   If you have no existing queries, the New Query dialog appears. If you have existing queries, the Chart Genie appears.

2. If the Chart Genie appears, select the New Query radio button and choose the OK button to show the New Query property sheet.

3. Type a SQL SELECT statement in the SQL Statement field, or select an alternative query type.

4. Choose the Execute button to execute the query.
   You can view the data returned in the Data property page.
Creating Queries at Chart Creation Time

7.5 Using Queries

The default name is queryx, where x is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name, or type a new name in the Name field.

3. Select a file type from the Query Type drop-down list.

4. Choose the Find button to show your system's file dialog.

5. Select a valid file and accept the dialog box. Graphics places the directory path and filename in the File name field.

Note: If your file contains dates, see the section "Setting Date Formats for File Queries" on page 7 – 16.

6. Choose the Execute button to execute the query. The data appears in the Data property page.

7. Choose the OK button or the Close button to close the Query property sheet. The query remains current until you create or edit another.

This section describes how to create a new query during the chart creation process. Use the instructions in the previous section for creating a specific type of query such as SQL, file, or custom.

After you create the new query with your chart, both objects are listed in the Object Navigator. You can edit the query's properties as described in the following sections of this chapter.

To create a query during chart creation:

1. Choose Chart —> Create Chart. Alternatively, you can select the Chart tool in the Layout editor Tool palette and either click in the display area, or drag out a region for the chart.

If you have no existing queries, the New Query dialog appears. If you have existing queries, the Chart Genie appears.

2. If the Chart Genie appears, select the New Query radio button and choose the OK button to show the New Query property sheet.

3. Type a SQL SELECT statement in the SQL Statement field, or select an alternative query type.

4. Choose the Execute button to execute the query. You can view the data returned in the Data property page.
5. Choose OK button to close the New Query property sheet and show the Chart property sheet.

6. Select your chart options and then choose the OK button to create the chart.

See the chapter “Using Charts” on page 8–1 for more information about setting chart properties.

Creating a Query Using PL/SQL

You can use parameters and Graphics PL/SQL built-ins to create and execute queries programmatically. For example, you may want to use a parameter to programmatically set a where clause for the query. You could do this by creating a query (query0) that includes a lexical reference to a parameter of the type CHAR: (the_param).

The following procedure sets a value for the parameter based on the argument provided, then executes the query and updates the associated chart:

```plsql
PROCEDURE add_param (the_name IN VARCHAR2) IS
    the_query  OG_QUERY;
    the_chart  OG_OBJECT;
BEGIN
    :the_param:='WHERE ename = '|| the_name;
    the_query:=OG_GET_QUERY('query0');
    OG_EXECUTE_QUERY(the_query);
    the_chart:=OG_GET_OBJECT('name_chart');
    OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;
```

Alternatively, you can create the entire query programmatically using the OG_MAKE_QUERY function. For example:

```plsql
FUNCTION make_ch RETURN og_query IS
BEGIN
    the_query:=OG_MAKE_QUERY(OG_SQL_QTYPE, 'select ename,
    sal from emp');
END;
```

Once you’ve created a query with the above function, you can execute the resulting query to create a chart.

5. Choose OK button to close the New Query property sheet and show the Chart property sheet.
6. Select your chart options and then choose the OK button to create the chart.

See the chapter “Using Charts” on page 8 – 1 for more information about setting chart properties.

You can use parameters and Graphics PL/SQL built-ins to create and execute queries programmatically. For example, you may want to use a parameter to programmatically set a where clause for the query. You could do this by creating a query (query0) that includes a lexical reference to a parameter of the type CHAR. (the_param).

The following procedure sets a value for the parameter based on the argument provided, then executes the query and updates the associated chart:

```plsql
PROCEDURE add_param (the_name IN VARCHAR2) IS
    the_query  OG_QUERY;
    the_chart  OG_OBJECT;
BEGIN
    :the_param:='WHERE ename = '|| the_name;
    the_query:=OG_GET_QUERY('query0');
    OG_EXECUTE_QUERY(the_query);
    the_chart:=OG_GET_OBJECT('name_chart');
    OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;
```

Alternatively, you can create the entire query programmatically using the OG_MAKE_QUERY function. For example:

```plsql
FUNCTION make_ch RETURN og_query IS
BEGIN
    the_query:=OG_MAKE_QUERY(OG_SQL_QTYPE, 'select ename, sal from emp');
END;
```

Once you’ve created a query with the above function, you can execute the resulting query to create a chart.

Using External SQL Queries

If you have existing SQL queries stored in files, you can either reference them, or import them into Graphics to use for creating charts. For more information about SQL SELECT statements and how to store them in text files, see the Oracle7 Server SQL Language Reference Manual.

Referencing External Queries

Referencing an external query file in Graphics provides a link to that file, rather than including the query in the display. Use this method if you do not want to edit the query, or include it in the display.

Note: If you distribute the display at runtime, you must also distribute the external query file so that Graphics can find it to execute.

To reference an external query stored on the file system:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator—>Create to show the Query property sheet.

   Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

   Alternatively, you can choose Tools—>Queries to show the Query property sheet, then choose the New button to create a new query.

2. Select External SQL File from the Type drop-down list.
   The File field and Browse button are activated.

3. Type the name and location of the SQL file in the File field or choose the Browse button to show your operating system file dialog.

4. Select the SQL file you wish to reference and accept the file dialog to close it.
   Graphics stores the path location to that file.

5. Choose Execute to execute the query in the external file.

6. Choose the OK button to close the Query property sheet.

Importing Queries

You can import a query stored in an external file on the file system. Once the query is imported, you can edit it, if necessary.

To import a query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator—>Create to show the Query property sheet.
Referencing an external query file in Graphics provides a link to that file, rather than including the query in the display. Use this method if you do not want to edit the query, or include it in the display.

Note: If you distribute the display at runtime, you must also distribute the external query file so that Graphics can find it to execute.

To reference an external query stored on the file system:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator —> Create to show the Query property sheet.

   Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

   Alternatively, you can choose Tools —> Queries to show the Query property sheet, then choose the New button to create a new query.

2. Select External SQL File from the Type drop-down list.

   The File field and Browse button are activated.

3. Type the name and location of the SQL file in the File field or choose the Browse button to show your operating system file dialog.

4. Select the SQL file you wish to reference and accept the file dialog to close it.

   Graphics stores the path location to that file.

5. Choose Execute to execute the query in the external file.

6. Choose the OK button to close the Query property sheet.

You can import a query stored in an external file on the file system. Once the query is imported, you can edit it, if necessary.

To import a query:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator —> Create to show the Query property sheet.
Creating the Query

Creating Custom Queries

Custom queries allow you to create columns and populate them with data using just PL/SQL, or a combination of SQL and PL/SQL. There are two stages to creating a custom query: creating the columns, and populating the columns with data.

After you’ve created a custom query, you can reference it when creating a chart.

Creating the Query Columns

You create the columns using the Schema property page of the Query property sheet, and populate the columns with data using a Custom Execution procedure.

To create a custom query using Graphics PL/SQL built-ins:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator—>Create to show the Query property sheet.

   Suggestion: If no queries currently exist, double-click on the Queries node in the Navigator to show the Query property sheet.

   Alternatively, you can choose Tools—>Queries to show the Query property sheet, then choose the New button to create a new query.

   2. Click the cursor in the SQL Statement field.

   3. Choose the Import SQL button to show your system’s file dialog.

   4. Select a text file containing a SQL SELECT statement and accept the dialog box.

      Graphics imports the SQL statement stored in the file into the SQL Statement field.

   5. Choose the Execute button to execute the imported query.

   6. Choose the OK button to close the dialog.
Creating the Query

Columns

Creating Custom Queries

Custom queries allow you to create columns and populate them with data using just PL/SQL, or a combination of SQL and PL/SQL. There are two stages to creating a custom query: creating the columns, and populating the columns with data.

After you've created a custom query, you can reference it when creating a chart.

You create the columns using the Schema property page of the Query property sheet, and populate the columns with data using a Custom Execution procedure.

To create a custom query using Graphics PL/SQL built-ins:

1. Select the Queries node in the Object Navigator and choose the Create button or Navigator —> Create to show the Query property sheet.
2. Choose the Import SQL button to show your system's file dialog.
3. Select a text file containing a SQL SELECT statement and accept the dialog box. Graphics imports the SQL statement stored in the file into the SQL Statement field.
4. Choose the Execute button to execute the imported query.
5. Choose the OK button to close the dialog.
The default name is $queryx$, where $x$ is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name or type a new name in the Name field.

3. Select Custom from the Query Type drop-down list.

4. Choose the Schema tab to show the Schema property page.

5. Choose the New Column button to create a column.

6. Change the name, type, and other settings, where applicable.

Continue creating all of the columns you wish to graph.

7. Continue to the next section for information on how to create a custom execution procedure to populate the columns with data, or choose the Close button to close the Query property sheet.

**Populating the Custom Query Columns**

Once you have created the empty columns, you can populate them with data in the following ways:

- using a combination of SQL and PL/SQL in a custom execution procedure
- from an external source such as a query from another Oracle product that invokes Graphics

**Using a Custom Execution Procedure**

You can populate the custom query columns you created using a custom execution procedure.

To create a custom execution procedure:

1. Double-click on the (custom) query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.

   Alternatively, you can choose Tools $\rightarrow$ Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Choose the Options tab to show the Options property page.

4. Select an existing custom execution trigger from the drop-down list, or choose the New button to create a new custom execution procedure.

   The Program Unit editor appears with a template OGQUERYPROC0 procedure.

5. Edit the template procedure to populate the custom columns.
Populating the Custom Query Columns
Using a Custom Execution Procedure

7. Using Queries

The default name is queryx, where x is an instance number that increments by one each time a new query is created during the current session.

2. Keep the default name or type a new name in the Name field.
3. Select Custom from the Query Type drop-down list.
4. Choose the Schema tab to show the Schema property page.
5. Choose the New Column button to create a column.
6. Change the name, type, and other settings, where applicable.
Continue creating all of the columns you wish to graph.
7. Continue to the next section for information on how to create a custom execution procedure to populate the columns with data, or choose the Close button to close the Query property sheet.

Once you have created the empty columns, you can populate them with data in the following ways:
• using a combination of SQL and PL/SQL in a custom execution procedure
• from an external source such as a query from another Oracle product that invokes Graphics

You can populate the custom query columns you created using a custom execution procedure.

To create a custom execution procedure:
1. Double-click on the (custom) query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.

Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Choose the Options tab to show the Options property page.
4. Select an existing custom execution trigger from the drop-down list, or choose the New button to create a new custom execution procedure.

The Program Unit editor appears with a template OGQUERYPROC0 procedure.
5. Edit the template procedure to populate the custom columns.
Use the OG_SET_xxxCELL and OG_APPEND_ROW PL/SQL built-in procedures. See the example below.

6. Choose the OK button to apply your changes and close the Query property sheet.

You can save and run your display, or choose Chart—>Update Chart to test the custom execution procedure.

Example

The following procedure is an example of a custom execution procedure. This procedure assumes that three columns were created in the Schema property page. The Name column is of type CHAR, the Salary column is of type NUMBER, and the Hired columns is of type DATE.

Graphics maintains a buffer row, or virtual row for the table. You populate the virtual row using OG_SET_xxxCELL, and then append the virtual row to the table using OG_APPEND_ROW. Notice that since there is only one virtual row, you can only populate one row of the table at a time.

```plsql
PROCEDURE fill_columns IS
    the_query  OG_QUERY;
    the_chart  OG_OBJECT;
BEGIN
    the_query:=OG_GET_QUERY('query0');

    /* Set the data for the first row */
    OG_SET_CHARCELL(the_query, 'Name', 'Smith');
    OG_SET_NUMCELL(the_query, 'Salary', '50000');
    OG_SET_DATECELL(the_query, 'Hired', '12–NOV–88');

    /* Append the row of data to the table*/
    OG_APPEND_ROW(the_query);

    /* Set the data for the next row*/
    OG_SET_CHARCELL(the_query, 'Name', 'Jones');
    OG_SET_NUMCELL(the_query, 'Salary', '55000');
    OG_SET_DATECELL(the_query, 'Hired', '19–APR–92');

    /* Append the row of data to the table*/
    OG_APPEND_ROW(the_query);

    /* Execute the query based on the data */
    OG_EXECUTE_QUERY(the_query);
END;
```
Use the OG_SET_xxxCELL and OG_APPEND_ROW PL/SQL built-in procedures. See the example below.

6. Choose the OK button to apply your changes and close the Query property sheet.

You can save and run your display, or choose Chart —> Update Chart to test the custom execution procedure.

The following procedure is an example of a custom execution procedure. This procedure assumes that three columns were created in the Schema property page. The Name column is of type CHAR, the Salary column is of type NUMBER, and the Hired column is of type DATE.

Graphics maintains a buffer row, or virtual row for the table. You populate the virtual row using OG_SET_xxxCELL, and then append the virtual row to the table using OG_APPEND_ROW. Notice that since there is only one virtual row, you can only populate one row of the table at a time.

```sql
PROCEDURE fill_columns IS
    the_query  OG_QUERY;
    the_chart  OG_OBJECT;
    BEGIN
        the_query:=OG_GET_QUERY('query0');
        /* Set the data for the first row */
        OG_SET_CHARCELL(the_query, 'Name', 'Smith');
        OG_SET_NUMCELL(the_query, 'Salary', '50000');
        OG_SET_DATECELL(the_query, 'Hired', '12–NOV–88');
        /* Append the row of data to the table*/
        OG_APPEND_ROW(the_query);
        /* Set the data for the next row*/
        OG_SET_CHARCELL(the_query, 'Name', 'Jones');
        OG_SET_NUMCELL(the_query, 'Salary', '55000');
        OG_SET_DATECELL(the_query, 'Hired', '19–APR–92');
        /* Append the row of data to the table*/
        OG_APPEND_ROW(the_query);
        /* Execute the query based on the data */
        OG_EXECUTE_QUERY(the_query);
    END;
```
Using an External Source  When you include a Graphics chart in another Developer/2000 component such as Forms, or Reports, the chart will use the data from the calling product’s query as long as the query columns used to create the chart and the columns used in the calling product’s query match.

If the chart you are building will only be called by another Oracle product, you can use a custom query to create an “empty” chart. This chart will be populated with data when it is invoked by the calling product. Use the custom query Schema property page to create columns with the same names as the query used by the calling product. Then define the chart’s properties as you normally would, assigning category and value data, etc.

The chart will appear in the layout with no data in it.

When the Graphics display is called by the other product, the calling product’s query is substituted for the custom query, and because the column names are the same, the chart is drawn to your specifications.

In this case, there is no need to create a custom execution procedure to populate the columns with data.

For more information about integrating Graphics displays in other Oracle products, see “Integration” on page 16 – 1.

Accessing Queries

Once you have created queries, you may want to access them to perform any of the following tasks:

• modifying the query properties or contents
• renaming the query
• deleting the query
• setting query execution options such as update events, query filters, or timers
• exporting a query to a file
• setting date format properties

There are two basic ways to access existing queries:

• using the Object Navigator
• using the menu
Using an External Source

When you include a Graphics chart in another Developer/2000 component such as Forms, or Reports, the chart will use the data from the calling product's query as long as the query columns used to create the chart and the columns used in the calling product's query match.

If the chart you are building will only be called by another Oracle product, you can use a custom query to create an "empty" chart. This chart will be populated with data when it is invoked by the calling product. Use the custom query Schema property page to create columns with the same names as the query used by the calling product. Then define the chart's properties as you normally would, assigning category and value data, etc.

The chart will appear in the layout with no data in it. When the Graphics display is called by the other product, the calling product's query is substituted for the custom query, and because the column names are the same, the chart is drawn to your specifications. In this case, there is no need to create a custom execution procedure to populate the columns with data.

For more information about integrating Graphics displays in other Oracle products, see "Integration" on page 16-1.

Accessing Queries

Once you have created queries, you may want to access them to perform any of the following tasks:

• modifying the query properties or contents
• renaming the query
• deleting the query
• setting query execution options such as update events, query filters, or timers
• exporting a query to a file
• setting date format properties

There are two basic ways to access existing queries:

• using the Object Navigator
• using the menu


**Accessing Queries with the Object Navigator**

Depending on the action you wish to perform, you can access the query’s property sheet using the Object Navigator or perform the action directly on the selected query object in the Navigator.

To show a specific query’s property sheet:

- Double-click on the query object in the Navigator, or select the query object and choose the Properties command from the pop-up menu.
  
  Once the Query property sheet is open, you can select other queries from the Query drop-down list.

To perform an action on a query using the buttons in the Object Navigator, or an option in the Navigator menu:

- Select the query object, then choose a button in the Object Navigator (e.g., Create or Delete), or choose an option from the Navigator menu.

**Accessing Queries with the Menu**

This method opens the Query property sheet for the current query. Once it’s opened, you can select an alternate query from the Query drop-down list. The current query is the most recently created or modified in the Query property sheet.

1. Choose **Tools**—>**Queries** to show the Query property sheet.
2. Select a query from the Query drop-down list.
3. Perform the desired action.

**Modifying Queries**

To modify the contents of an existing query:

1. Double-click on the query object you wish to modify, or select the query object in the Navigator and choose the Properties command from the pop-up menu.

   Alternatively, you can choose **Tools**—>**Queries** to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Edit the query as follows:

   - If the query is a SQL SELECT statement, edit the SQL SELECT statement in the SQL Statement field.
   - If the query is based on a file, your system’s file dialog appears. Select a file and accept the dialog. Edit the file as necessary.
Accessing Queries with the Object Navigator

Depending on the action you wish to perform, you can access the query's property sheet using the Object Navigator or perform the action directly on the selected query object in the Navigator.

To show a specific query's property sheet:
- Double-click on the query object in the Navigator,
- or select the query object and choose the Properties command from the pop-up menu.

Once the Query property sheet is open, you can select other queries from the Query drop-down list.

To perform an action on a query using the buttons in the Object Navigator, or an option in the Navigator menu:
- Select the query object, then choose a button in the Object Navigator (e.g., Create or Delete), or choose an option from the Navigator menu.

This method opens the Query property sheet for the current query. Once it's opened, you can select an alternate query from the Query drop-down list. The current query is the most recently created or modified in the Query property sheet.

1. Choose **Tools** —> **Queries** to show the Query property sheet.
2. Select a query from the Query drop-down list.
3. Perform the desired action.

Modifying Queries

To modify the contents of an existing query:
- Double-click on the query object you wish to modify,
- or select the query object in the Navigator and choose the Properties command from the pop-up menu.

Alternatively, you can choose **Tools** —> **Queries** to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Edit the query as follows:
   - If the query is a SQL SELECT statement, edit the SQL SELECT statement in the SQL Statement field.
   - If the query is based on a file, your system's file dialog appears. Select a file and accept the dialog. Edit the file as necessary.
4. Choose the Execute button to re-execute the query.
   The new data appears in the Data property page.

5. Choose the OK button to close the Query property sheet.
   The query remains current until you create or edit another.

**Exporting Queries to Text Files**

Once you have created a SQL query in Graphics, you can export that query to a text file for use in other Oracle products. You can re-use queries within Graphics by dragging a query from one display to another in the Object Navigator.

To export a query to a text file:

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.
   Alternatively, you can choose **Tools**→**Queries** to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Click the cursor in the SQL Statement field.

4. Select the Export SQL button to show your system’s file dialog.

5. Select a text file to contain the SQL statement and accept the dialog box.

6. Choose the OK button to close the Query property sheet.
   Graphics exports the SQL statement to the named file.

**Executing Queries**

You can execute queries in the following ways:

- automatically upon opening the display
- at specific time intervals
- using PL/SQL
Using Queries

4. Choose the Execute button to re-execute the query.
The new data appears in the Data property page.

5. Choose the OK button to close the Query property sheet.
The query remains current until you create or edit another.

Exporting Queries to Text Files

Once you have created a SQL query in Graphics, you can export that query to a text file for use in other Oracle products. You can re-use queries within Graphics by dragging a query from one display to another in the Object Navigator.

To export a query to a text file:

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Click the cursor in the SQL Statement field.

4. Select the Export SQL button to show your system's file dialog.

5. Select a text file to contain the SQL statement and accept the dialog box.

6. Choose the OK button to close the Query property sheet.
Graphics exports the SQL statement to the named file.

Executing Queries

You can execute queries in the following ways:

• automatically upon opening the display
• at specific time intervals
• using PL/SQL
Executing Queries Upon Opening a Display

To execute a query when you open a display:

1. Double-click on the query object you want to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.
   Alternatively, you can choose Tools→Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Choose the Options tab to show the Options property page.

4. Activate Execute on Opening Display by clicking on its check box.

Executing Queries at Specific Time Intervals

To execute a query at a specific time:

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.
   Alternatively, you can choose Tools→Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Choose the Options tab to show the Options property page.

4. Activate Execute on Timer by clicking on its check box.

5. Select an existing timer from the drop-down list, or choose the New button to create a new timer.
   For more information about using timers, see “Using Timers” on page 11–9.

Executing Queries Using PL/SQL

You can also execute a query using PL/SQL. The following example executes the specified query and updates the associated chart:

```plsql
PROCEDURE every_30_secs IS
  the_query  OG_QUERY;
  the_chart  OG_OBJECT;
BEGIN
  the_query:=OG_GET_QUERY('emp_query');
  the_chart:=OG_GET_OBJECT('emp_chart');
  OG_EXECUTE_QUERY(the_query);
```

---

7 – 14 Graphics Developer’s Guide
To execute a query when you open a display:
1. Double-click on the query object you want to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Choose the Options tab to show the Options property page.
4. Activate Execute on Opening Display by clicking on its check box.

To execute a query at a specific time:
1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Choose the Options tab to show the Options property page.
4. Activate Execute on Timer by clicking on its check box.
5. Select an existing timer from the drop-down list, or choose the New button to create a new timer.
For more information about using timers, see "Using Timers" on page 11–9.

You can also execute a query using PL/SQL. The following example executes the specified query and updates the associated chart:

```plsql
PROCEDURE every_30_secs IS
    the_query   OG_QUERY;
    the_chart   OG_OBJECT;
BEGIN
    the_query:=OG_GET_QUERY('emp_query');
    the_chart:=OG_GET_OBJECT('emp_chart');
    OG_EXECUTE_QUERY(the_query);
END;
```
OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;


Renaming Queries

You can rename queries directly in the Object Navigator, or from the Query property sheet.

Note: If you rename queries that are referenced in any PL/SQL program units or drill-down relationships, you must change all references to reflect the new query name.

Renaming Queries in the Object Navigator

Use this method to quickly rename one or more queries directly in the object navigator. If you want to view the contents of the query before renaming it, use the menu method below.

To rename a query in the Object Navigator:
1. Select the query object you wish to rename.
2. Move the mouse pointer over the query name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.

Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

Renaming Queries in the Query Property Sheet

Use this method if you want to view the contents of the query before renaming it.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.

Alternatively, you can choose Tools—Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.
3. Type a new name for the query in the Name field.
4. Choose the OK button to apply the new name and close the Query property sheet.
Renaming Queries in the Object Navigator

Renaming Queries in the Query Property Sheet


Renaming Queries

You can rename queries directly in the Object Navigator, or from the Query property sheet.

Note: If you rename queries that are referenced in any PL/SQL program units or drill-down relationships, you must change all references to reflect the new query name.

Use this method to quickly rename one or more queries directly in the object navigator. If you want to view the contents of the query before renaming it, use the menu method below.

To rename a query in the Object Navigator:

1. Select the query object you wish to rename.
2. Move the mouse pointer over the query name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode. Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

Use this method if you want to view the contents of the query before renaming it.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Type a new name for the query in the Name field.
4. Choose the OK button to apply the new name and close the Query property sheet.
Deleting Queries

You can delete an existing query directly from the Object Navigator, or from the Query property sheet.

Note: If you delete queries that are referenced in any PL/SQL program units or drill-down relationships, you must change all references to reflect the deleted query.

Deleting Queries in the Object Navigator

Use this method to quickly delete one or more queries listed in the Object Navigator. If you wish to view the query before deleting it, use the Query property sheet method described below.

1. Select the query object you wish to delete.
2. Choose the Delete button in the Object Navigator, or choose Navigator—>Delete.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

Deleting Queries in the Query Property Sheet

Use this method if you wish to view the contents of a query prior to deleting it.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.
   Alternatively, you can choose Tools—>Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Choose the Delete button.
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.

Setting Date Formats for File Queries

If you are querying a file (SYLK, WKS, or PRN) that includes dates, and you want Graphics to recognize them as dates and not as individual characters, you need to specify date formats in the Query property sheet.
Deleting Queries in the Object Navigator

Deleting Queries in the Query Property Sheet

You can delete an existing query directly from the Object Navigator, or from the Query property sheet.

Note: If you delete queries that are referenced in any PL/SQL program units or drill-down relationships, you must change all references to reflect the deleted query.

Use this method to quickly delete one or more queries listed in the Object Navigator. If you wish to view the query before deleting it, use the Query property sheet method described below.

1. Select the query object you wish to delete.
2. Choose the Delete button in the Object Navigator, or choose Navigator —> Delete.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

Use this method if you wish to view the contents of a query prior to deleting it.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.
   Alternatively , you can choose Tools —> Queries to show the Query property sheet for the current query.
2. If necessary, select an alternate query from the Query drop-down list.
3. Choose the Delete button.
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.

Setting Date Formats for File Queries

If you are querying a file (SYLK, WKS, or PRN) that includes dates, and you want Graphics to recognize them as dates and not as individual characters, you need to specify date formats in the Query property sheet.
Note: This date format is used only for dates on the Data property page in the Query property sheet. For charts, you can set date formats in the Layout or Chart Template editors.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu.

   Alternatively, you can choose **Tools—>Queries** to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Choose the Options tab to show the Options property page.

4. Type a date format in the Date Format field, or choose List and select an existing format from the Date Format dialog box.

   For more information about valid formats, see the *Oracle7 Server SQL Language Reference Manual*.

5. Choose the OK button to accept the dialog box and return to the Query property sheet.

6. Choose the Execute button to re-execute the query and plot the data in the new format.

Using Query Filter Functions

Using Graphics, you can create a single query that can be assigned to multiple charts. By creating a query filter function for a chart, you can restrict the data plotted in the chart (e.g., all salaries greater than 5000) without the need to re-execute your query.

1. Double-click on the chart, or select the chart to which you wish to add a query filter (either in the Layout editor or the Object Navigator) and choose the Properties command from the pop-up menu.

   Alternatively, you can choose **Tools—>Properties** to show the Chart property sheet.

2. Choose the Data tab to show the Data property page.

3. Select an existing query filter function from the Filter Function drop-down list, or choose the New button to create a new query filter.
Using Queries

Note:
This date format is used only for dates on the Data property page in the Query property sheet. For charts, you can set date formats in the Layout or Chart Template editors.

1. Double-click on the query object you wish to modify, or select the query object in the Object Navigator and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Queries to show the Query property sheet for the current query.

2. If necessary, select an alternate query from the Query drop-down list.

3. Choose the Options tab to show the Options property page.

4. Type a date format in the Date Format field, or choose List and select an existing format from the Date Format dialog box. For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.

5. Choose the OK button to accept the dialog box and return to the Query property sheet.

6. Choose the Execute button to re-execute the query and plot the data in the new format.

Using Query Filter Functions

Using Graphics, you can create a single query that can be assigned to multiple charts. By creating a query filter function for a chart, you can restrict the data plotted in the chart (e.g., all salaries greater than 5000) without the need to re-execute your query.

1. Double-click on the chart, or select the chart to which you wish to add a query filter (either in the Layout editor or the Object Navigator) and choose the Properties command from the pop-up menu. Alternatively, you can choose Tools —> Properties to show the Chart property sheet.

2. Choose the Data tab to show the Data property page.

3. Select an existing query filter function from the Filter Function drop-down list, or choose the New button to create a new query filter.
For more information about query filter functions, see “Filtering Chart Data” on page 8 – 11.

4. Choose the OK button to apply your changes and close the Chart property sheet.

You can use Chart—>Update Chart to test the query filter function.
For more information about query filter functions, see "Filtering Chart Data" on page 8–11.

4. Choose the OK button to apply your changes and close the Chart property sheet.

You can use Chart —> Update Chart to test the query filter function.
Using Charts

This chapter discusses how to create and use charts in Graphics to visually display data.

The following topics are covered in this chapter:

- working with charts – 8 – 2
- creating charts – 8 – 8
- accessing chart properties – 8 – 10
- modifying chart properties – 8 – 11
- customizing charts – 8 – 15
- working with field templates – 8 – 22
- updating charts – 8 – 25
- working with drill-down charts – 8 – 26
- converting charts to artwork – 8 – 28
- renaming charts – 8 – 28
- deleting charts – 8 – 29
Chapter 8

This chapter discusses how to create and use charts in Graphics to visually display data.

The following topics are covered in this chapter:

- Working with charts
- Creating charts
- Accessing chart properties
- Modifying chart properties
- Customizing charts
- Working with field templates
- Updating charts
- Working with drill-down charts
- Converting charts to artwork
- Renaming charts
- Deleting charts
Working with Charts

Charts are graphic diagrams you can create in a display to visually represent data. This section describes various aspects of charts, including:

- chart components
- chart properties
- data classification
- axis charts
- chart types
- chart templates
- dynamic vs. artwork only charts

Chart Components

A chart appears on the layout as a single object, but it is made up of these main components:

**Query**

Is a SQL SELECT statement or file that Graphics needs in order to retrieve data for a chart. A chart can be associated with only one query at a time. When you draw the chart, Graphics uses the data from the current query, unless you specify a different one.

**Chart region**

Is the space that the chart occupies. It defines the chart’s size and position on the layout. A chart can be associated with only one region at a time. However, you can duplicate a chart to appear multiple times in a display.

**Chart template**

Defines the format of a chart and its various properties. A chart can be associated with only one chart template at a time. Each chart template can be associated with one or more field templates, which define the properties of the individual fields, such as bars, columns, pie slices, or lines.

Creating Chart Components

You can create chart components in the Layout editor by specifying a query and chart template to use and defining its region on the layout. Alternately, you can create chart components programmatically using PL/SQL.

For example, you can programmatically define a query, select a chart template, make a field into a button, or “explode” a field by setting it apart from the rest of the chart.
Charts are graphic diagrams you can create in a display to visually represent data. This section describes various aspects of charts, including:

- Chart components
- Chart properties
- Data classification
- Axis charts
- Chart types
- Chart templates
- Dynamic vs. artwork only charts

A chart appears on the layout as a single object, but it is made up of these main components:

- **Query**: A SQL SELECT statement or file that Graphics needs in order to retrieve data for a chart. A chart can be associated with only one query at a time. When you draw the chart, Graphics uses the data from the current query, unless you specify a different one.

- **Chart region**: The space that the chart occupies. It defines the chart's size and position on the layout. A chart can be associated with only one region at a time. However, you can duplicate a chart to appear multiple times in a display.

- **Chart template**: Defines the format of a chart and its various properties. A chart can be associated with only one chart template at a time. Each chart template can be associated with one or more field templates, which define the properties of the individual fields, such as bars, columns, pie slices, or lines.

You can create chart components in the Layout editor by specifying a query and chart template to use and defining its region on the layout. Alternately, you can create chart components programmatically using PL/SQL.

For example, you can programmatically define a query, select a chart template, make a field into a button, or "explode" a field by setting it apart from the rest of the chart.
You can also use PL/SQL to define the graphic properties of fields by setting their pattern, color, or line style. Then you can set up a condition, such as a specific threshold value for a data category, under which a property should be applied. At runtime, Graphics will apply the properties to the specified fields depending on the data categories or conditions present.

Sample PL/SQL program units are provided throughout this chapter to demonstrate some of these tasks.


**Chart Properties**

Chart properties define options for the chart as a whole, including its associated chart template and query. Chart properties are accessible through the Chart property sheet and are organized as follows:

- **Chart**: Includes the chart’s name, title, chart type, and template.
- **Data**: Includes query–related properties such as a query filter function and the range of data to plot.
- **Categories**: Includes the chart categories and any sub-categories to plot in the chart. Category (independent) data is plotted at fixed intervals. Category data is not considered to be mathematically related and is usually plotted along what is called the Discrete axis.
- **Values**: Includes the chart values and associated field templates. Value (dependent) data generally starts at one value and continues until another value. Value data is considered to be mathematically related and is usually plotted along what is called the Continuous axis.
- **PL/SQL**: Includes any mouse events and associated procedures for the chart.

In general, you define chart properties by selecting the chart, opening its property sheet, and choosing the options you want to define for it. You can also define many of these options programmatically, using PL/SQL.

**Data Classification**

Each column in the query contains either category or value data.
You can also use PL/SQL to define the graphic properties of fields by setting their pattern, color, or line style. Then you can set up a condition, such as a specific threshold value for a data category, under which a property should be applied. At runtime, Graphics will apply the properties to the specified fields depending on the data categories or conditions present.

Sample PL/SQL program units are provided throughout this chapter to demonstrate some of these tasks.


Chart properties define options for the chart as a whole, including its associated chart template and query. Chart properties are accessible through the Chart property sheet and are organized as follows:

- **Chart** includes the chart's name, title, chart type, and template.
- **Data Categories** includes query–related properties such as a query filter function and the range of data to plot.
- **Values** includes the chart categories and any sub-categories to plot in the chart. Category (independent) data is plotted at fixed intervals. Category data is not considered to be mathematically related and is usually plotted along what is called the Discrete axis.
- **Values** includes the chart values and associated field templates. Value (dependent) data generally starts at one value and continues until another value. Value data is considered to be mathematically related and is usually plotted along what is called the Continuous axis.
- **Mouse Events** includes any mouse events and associated procedures for the chart.

In general, you define chart properties by selecting the chart, opening its property sheet, and choosing the options you want to define for it. You can also define many of these options programmatically, using PL/SQL.
Categories

A category is a piece of data that has a value associated with it. For example, if you are interested in knowing the salaries of various employees, the employee name would be the category. Then you could chart the salary for each employee (i.e., the value for each category).

Columns containing categories are often called independent because the category does not depend on any other piece of data. Using the above example, the employee name does not depend on any other piece of data.

Categories are usually plotted along a discrete or date axis, which may then also be called a “category axis.”

Values

A value is a piece of data that is associated with a category. For example, a quantity of 5000 must be associated with a specific product (the product is the category).

Columns containing values are often called dependent because the value depends on another piece of data. Using the above example, knowing the quantity of 5000 is meaningless unless you also know the product that that quantity refers to. The quantity value depends on a specific product.

Values are usually plotted along a continuous axis, which may then also be called a “value axis.”

Field Mapping

Field mapping is the process by which you specify which columns in the query to include in the chart, how the columns will be represented as chart fields, and the order in which the fields are plotted. Field mapping also involves associating Value (dependent) data fields with field templates such as column, or bar.

When you first create a chart, Graphics does the field mapping automatically, based on the way and order that columns appear in the query properties Data page, and the current field template. You can change the defaults, by selecting the chart, opening its property sheet, and modifying the settings in the Values and Categories pages.

For example, you may want to add or remove a data field from the chart, or change the order in which the data fields are plotted. You can create a break chart by specifying a sub-category data field to plot on the chart.

Axis Charts

Most of the charts you can create in Graphics are axis charts. An axis chart is a chart in which the data is plotted according to where it lies along the edges of the chart, called axes. Column, Bar, and Line charts are all Axis charts. An axis chart may contain two or three axes, each of which may be one of the following types:
A category is a piece of data that has a value associated with it. For example, if you are interested in knowing the salaries of various employees, the employee name would be the category. Then you could chart the salary for each employee (i.e., the value for each category).

Columns containing categories are often called independent because the category does not depend on any other piece of data. Using the above example, the employee name does not depend on any other piece of data.

Categories are usually plotted along a discrete or date axis, which may then also be called a "category axis."

A value is a piece of data that is associated with a category. For example, a quantity of 5000 must be associated with a specific product (the product is the category).

Columns containing values are often called dependent because the value depends on another piece of data. Using the above example, knowing the quantity of 5000 is meaningless unless you also know the product that that quantity refers to. The quantity value depends on a specific product.

Values are usually plotted along a continuous axis, which may then also be called a "value axis."

Field mapping is the process by which you specify which columns in the query to include in the chart, how the columns will be represented as chart fields, and the order in which the fields are plotted. Field mapping also involves associating Value (dependent) data fields with field templates such as column, or bar.

When you first create a chart, Graphics does the field mapping automatically, based on the way and order that columns appear in the query properties Data page, and the current field template. You can change the defaults, by selecting the chart, opening its property sheet, and modifying the settings in the Values and Categories pages.

For example, you may want to add or remove a data field from the chart, or change the order in which the data fields are plotted. You can create a break chart by specifying a sub-category data field to plot on the chart.

Most of the charts you can create in Graphics are axis charts. An axis chart is a chart in which the data is plotted according to where it lies along the edges of the chart, called axes. Column, Bar, and Line charts are all Axis charts. An axis chart may contain two or three axes, each of which may be one of the following types:
• discrete
• continuous
• date

Discrete Axis
A discrete axis is one that plots distinct values at fixed intervals. These values are not mathematically related. Category (independent) data is usually plotted along this axis, also called the category axis.

For example, employee names may be plotted along a discrete axis. However, their salaries cannot (since monetary values are related by one being mathematically greater than another).

Below is an example of a discrete axis:

| SMITH | ALLEN | WARD | JONES | MARTIN |

Continuous Axis
A continuous axis is one that plots mathematically related values. The axis begins at one value, and “continues” until it reaches another value. The plotting interval depends on where the data values fall between the lower and upper limits of the axis. Value (dependent) data is usually plotted along this axis, also called the value axis.

For example, employee salaries may be plotted along a continuous axis. However, employee names cannot.

Below is an example of a continuous axis:

$5000
$4000
$3000
$2000
$1000
$0

Date Axis
A date axis is one that plots date values. The axis begins at one date, and continues until it reaches another date. (In this respect, a date axis resembles a continuous axis; however, Graphics treats them very differently.)
A discrete axis is one that plots distinct values at fixed intervals. These values are not mathematically related. Category (independent) data is usually plotted along this axis, also called the category axis. For example, employee names may be plotted along a discrete axis. However, their salaries cannot (since monetary values are related by one being mathematically greater than another).

Below is an example of a discrete axis:

SMITH ALLEN WARD JONES MARTIN

A continuous axis is one that plots mathematically related values. The axis begins at one value, and "continues" until it reaches another value. The plotting interval depends on where the data values fall between the lower and upper limits of the axis. Value (dependent) data is usually plotted along this axis, also called the value axis. For example, employee salaries may be plotted along a continuous axis. However, employee names cannot.

Below is an example of a continuous axis:

$0 $1000 $2000 $3000 $4000 $5000

A date axis is one that plots date values. The axis begins at one date, and continues until it reaches another date. (In this respect, a date axis resembles a continuous axis; however, Graphics treats them very differently.)
The plotting interval depends on where the data values fall between the lower and upper limits of the axis.

For example, employees’ hire dates may be plotted along a date axis.

Below is an example of a date axis:

```
JAN  FEB  MAR  APR
```

**Chart Types**

The following list describes the available types of charts you can create, and their suggested usage:

- **Column**: Use this chart type to compare sets of data ranges vertically. For example, use a column chart to show the quarterly sales revenues generated by each sales representative.

- **Pie**: Use this chart type to compare the ratios or percentages of parts of a whole. For example, use a pie chart to compare annual revenue by department or quarter.

- **Bar**: Use this chart type to compare sets of data ranges horizontally. For example, use a bar chart to compare monthly revenues.

- **Table**: Use this chart type to show data in a table format. For example, use a table chart to show an employee organization chart.

- **Line**: Use this chart type to show vertical changes for a specific set of data. For example, use a line chart to gauge daily or hourly changes in a stock’s value.

- **Scatter**: Use this chart type to show data along two value axes. Scatter charts are well suited for showing standard deviations. For example, use a scatter chart to plot the ages and salaries for your employees. If there is a correlation between the two sets of data, the points will be grouped together fairly closely. One or more points well outside the group could indicate a salary disparity.

- **Mixed**: Combines multiple plot types such as column and line. For example, use a mixed chart to show daily sales revenue, with a line plot type to provide a summary view of the data.
The plotting interval depends on where the data values fall between the lower and upper limits of the axis. For example, employees’ hire dates may be plotted along a date axis.

Below is an example of a date axis:

JAN  FEB  MAR  APR

The following list describes the available types of charts you can create, and their suggested usage:

- **Column**: Use this chart type to compare sets of data ranges vertically. For example, use a column chart to show the quarterly sales revenues generated by each sales representative.

- **Pie**: Use this chart type to compare the ratios or percentages of parts of a whole. For example, use a pie chart to compare annual revenue by department or quarter.

- **Bar**: Use this chart type to compare sets of data ranges horizontally. For example, use a bar chart to compare monthly revenues.

- **Table**: Use this chart type to show data in a table format. For example, use a table chart to show an employee organization chart.

- **Line**: Use this chart type to show vertical changes for a specific set of data. For example, use a line chart to gauge daily or hourly changes in a stock’s value.

- **Scatter**: Use this chart type to show data along two value axes. Scatter charts are well suited for showing standard deviations. For example, use a scatter chart to plot the ages and salaries for your employees. If there is a correlation between the two sets of data, the points will be grouped together fairly closely. One or more points well outside the group could indicate a salary disparity.

- **Mixed**: Combines multiple plot types such as column and line. For example, use a mixed chart to show daily sales revenue, with a line plot type to provide a summary view of the data.
High–low

Use this chart type to show values fields that correspond to high, low, and close (i.e. stock prices). For each row in your query, a high–low range is plotted on the chart.

Double–Y

This chart type provides two independent Y–axes on which to plot data. Each Y–axis can show a different range of values. For example, you could use one axis to represent revenue, the second to represent income, and plot both over a course of time.

Gantt

Use this chart type to show sets of project data over a given amount of time. Gantt charts are generally used to show project milestone timelines.

Chart Templates

Chart templates define the format of a chart. They are made up of various components for which you can define properties as you can for objects.

Using a single chart template, you can create multiple charts with the same set basic of characteristics such as grid line settings or tick label rotation. For example, instead of specifying the properties for each chart individually, you can create a single chart template and associate it with multiple queries. To work with chart templates, you use the Chart Template editor.

Types of Templates

Graphics provides a set of 56 default chart templates divided into 10 categories, including bar, line, and pie. When you create a chart of a specific type and sub-type, Graphics makes a copy of the default chart template associated with that type. You can customize this copy.

For more information about chart templates and the Chart Template editor, see “Using Chart Templates” on page 9 – 1.

Dynamic vs. Artwork Only Charts

A dynamic chart is a chart linked to a data source. Initially, every chart is dynamic because Graphics needs data with which to draw it. Keeping a chart dynamic allows you to update it whenever you re-execute its query or change its chart template. When you update a chart, Graphics looks at the contents of the Data property page and the Chart Template editor, and redraws the chart based on the new data.

If you want to make a chart into a static object, you can convert it to artwork. This freezes the chart with one “snapshot” of the data and breaks its connection to the data source so that you can no longer update it. The advantage of an artwork-only chart is that its graphic components become separate artwork objects that you can break apart and edit individually.
Chart Templates

Types of Templates

Dynamic vs. Artwork

Only Charts

Using Charts

Use this chart type to show values fields that correspond to high, low, and close (i.e. stock prices). For each row in your query, a high–low range is plotted on the chart.

This chart type provides two independent Y–axes on which to plot data. Each Y–axis can show a different range of values. For example, you could use one axis to represent revenue, the second to represent income, and plot both over a course of time.

Use this chart type to show sets of project data over a given amount of time. Gantt charts are generally used to show project milestone timelines.

Chart templates define the format of a chart. They are made up of various components for which you can define properties as you can for objects.

Using a single chart template, you can create multiple charts with the same set basic of characteristics such as grid line settings or tick label rotation. For example, instead of specifying the properties for each chart individually, you can create a single chart template and associate it with multiple queries. To work with chart templates, you use the Chart Template editor.

Graphics provides a set of 56 default chart templates divided into 10 categories, including bar, line, and pie. When you create a chart of a specific type and sub-type, Graphics makes a copy of the default chart template associated with that type. You can customize this copy. For more information about chart templates and the Chart Template editor, see “Using Chart Templates” on page 9 – 1.

A dynamic chart is a chart linked to a data source. Initially, every chart is dynamic because Graphics needs data with which to draw it. Keeping a chart dynamic allows you to update it whenever you re-execute its query or change its chart template. When you update a chart, Graphics looks at the contents of the Data property page and the Chart Template editor, and redraws the chart based on the new data.

If you want to make a chart into a static object, you can convert it to artwork. This freezes the chart with one “snapshot” of the data and breaks its connection to the data source so that you can no longer update it. The advantage of an artwork-only chart is that its graphic components become separate artwork objects that you can break apart and edit individually.
Creating Charts

Creating a chart is essentially a two-step process that involves specifying a query and drawing the chart on the layout. Once you create a chart you can modify it in a variety of ways.

Creating a Chart in the Layout Editor

Creating a Chart from a New Query

To create a new query and associated chart in one step:

1. Choose File—>Connect to connect to a database if you are not already connected.
2. Select the Chart tool in the Tool palette and either click in the layout or drag out a region for the chart.
   Alternatively, you can choose Chart—>Create Chart from the menu.
   If you do not drag out a region for the chart, Graphics will use a default size.
   If you have existing queries in your display, the Chart Genie dialog appears. If you have no existing queries, the New Query dialog appears.
3. If the Chart Genie appears, select the New Query radio button then choose the OK button to show the New Query dialog.
4. Type a SQL SELECT statement in the SQL Statement field, or select an alternative query type.
5. Choose the Execute button to execute the query.
   You can view the data returned in the Data property page.
6. Choose the OK button to close the New Query property sheet and show the Chart property sheet.
7. If you wish, enter a name and title for your chart.
8. Select a Chart Type and Subtype.
9. Choose the OK button to create the chart.
   The chart is drawn in the Layout editor. The chart object appears in the Object Navigator.

Note: The chart region is the area covered by the chart. The tick labels and legend may extend beyond this.

You can now manipulate the chart object (select, move, resize, delete, etc.) as you would other object on the layout, or modify its properties.
Creating a Chart in the Layout Editor

Creating a Chart from a New Query

Creating charts is essentially a two-step process that involves specifying a query and drawing the chart on the layout. Once you create a chart, you can modify it in a variety of ways. You can create a chart based on a query you have already defined, or you can define the query at the same time you create the chart.

To create a new query and associated chart in one step:

1. Choose **File** —> **Connect** to connect to a database if you are not already connected.
2. Select the Chart tool in the Tool palette and either click in the layout or drag out a region for the chart. Alternatively, you can choose **Chart** —> **Create Chart** from the menu.
3. If you do not drag out a region for the chart, Graphics will use a default size.
4. If you have existing queries in your display, the Chart Genie dialog appears. If you have no existing queries, the New Query dialog appears.
5. If the Chart Genie appears, select the New Query radio button then choose the OK button to show the New Query dialog.
6. Type a SQL SELECT statement in the SQL Statement field, or select an alternative query type.
7. Choose the Execute button to execute the query. You can view the data returned in the Data property page.
8. Choose the OK button to close the New Query property sheet and show the Chart property sheet.
9. If you wish, enter a name and title for your chart.
10. Select a Chart Type and Subtype.
11. Choose the OK button to create the chart.

The chart is drawn in the Layout editor. The chart object appears in the Object Navigator.

**Note:**

The chart region is the area covered by the chart. The tick labels and legend may extend beyond this.

You can now manipulate the chart object (select, move, resize, delete, etc.) as you would other objects on the layout, or modify its properties.
Creating a Chart from an Existing Query

To create a chart from a previously defined query:

1. Select the Chart tool in the Tool palette and either click in the layout or drag out a region for the chart.
   Alternatively, you can choose Chart—Create Chart from the menu.
   If you do not drag out a region for the chart, Graphics will use a default size.
   The Chart Genie dialog appears.

2. Select an existing query from the Existing Query drop-down list.

3. Choose the OK button to accept the dialog and show the Chart property sheet.

4. If you wish, enter a name and title for your chart.

5. Select a Chart Type and Subtype.

6. Choose the OK button to create the chart.
   The chart is drawn in the Layout editor. The chart object appears in the Object Navigator.

Note: The chart region is the area covered by the chart. The tick labels and legend may extend beyond this.
You can now manipulate the chart object (select, move, resize, delete, etc.) as you would other object on the layout, or modify its properties.

Creating a Chart Programmatically

You can also create a chart using PL/SQL. The following example executes the specified query and stores the results internally. The following procedure assumes that the template and query already have been defined:

```plsql
PROCEDURE create_chart IS
    /* Declare the variables */
    the_chart      OG_OBJECT;
    ch_pos         OG_POINT;
    ch_hy          NUMBER;
    ch_wi          NUMBER;
    the_query      OG_QUERY;
    the_template   OG_TEMPLATE;
    indep_field    OG_FIELD;
    dep_field      OG_FIELD;
    BEGIN
        /* Get the template and query that have already been defined in the Designer */
```

Using Charts 8–9
Creating a Chart from an Existing Query

Creating a Chart Programatically

PL/SQL

To create a chart from a previously defined query:

1. Select the Chart tool in the Tool palette and either click in the layout or drag out a region for the chart.
2. Alternatively, you can choose Chart —> Create Chart from the menu.
3. If you do not drag out a region for the chart, Graphics will use a default size.
4. The Chart Genie dialog appears.
5. Select an existing query from the Existing Query drop-down list.
6. Choose the OK button to accept the dialog and show the Chart property sheet.
7. If you wish, enter a name and title for your chart.
8. Select a Chart Type and Subtype.
9. Choose the OK button to create the chart.

The chart is drawn in the Layout editor. The chart object appears in the Object Navigator.

Note: The chart region is the area covered by the chart. The tick labels and legend may extend beyond this.

You can now manipulate the chart object (select, move, resize, delete, etc.) as you would other objects on the layout, or modify its properties.

You can also create a chart using PL/SQL. The following example executes the specified query and stores the results internally. The following procedure assumes that the template and query already have been defined:

```plsql
PROCEDURE create_chart IS
    /* Declare the variables */
    the_chart      OG_OBJECT;
    ch_pos         OG_POINT;
    ch_hy          NUMBER;
    ch_wi          NUMBER;
    the_query      OG_QUERY;
    the_template   OG_TEMPLATE;
    indep_field    OG_FIELD;
    dep_field      OG_FIELD;
    BEGIN
        /* Get the template and query that have already been defined in the Designer */
        /* The rest of the procedure... */
    END;
```
the_template:=OG_GET_TEMPLATE('template0');
the_query:=OG_GET_QUERY('query0');
/* Set the position of the chart in X/Y coordinates */
ch_pos.x:=0*OG_INCH;
ch_pos.y:=0*OG_INCH;
/* Set the height and width of the chart */
ch_ht:=3*OG_INCH;
ch_wi:=3*OG_INCH;
/* Make the chart */
the_chart:=OG_MAKE_CHART(chpos, ch_ht, ch_wi,
          the_template, the_query);
/* Define a field for ENAME */
indep_field.field_type:=OG_INDEPENDENT;
indep_field.colname:='ENAME';
/* Insert it into the chart in the 0th position */
OG_INSERT_FIELD(the_chart, indep_field, 0);
/* Define a field for SAL */
dep_field.field_type:=OG_DEPENDENT;
dep_field.colname:='SAL';
dep_field.ftname:='column';
/* Insert it into the chart in the second position */
OG_INSERT_FIELD(the_chart, dep_field, 1);
/* Define a field for COMM */
dep_field.field_type:=OG_DEPENDENT;
dep_field.colname:='COMM';
dep_field.ftname:='column';
/* Insert it into the chart in the last position */
OG_INSERT_FIELD(the_chart, dep_field, OG_LAST);
/* Execute the query for the first time */
OG_EXECUTE_QUERY(the_query);
/* Update the chart so the new field and query information takes effect */
OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;


Accessing Chart Properties

There are two basic ways to access a chart’s property sheet:
the_template:=OG_GET_TEMPLATE('template0');
the_query:=OG_GET_QUERY('query0');

/*Set the position of the chart in X/Y coordinates*/
ch_pos.x:=0*OG_INCH;
ch_pos.y:=0*OG_INCH;

/*Set the height and width of the chart */
ch_ht:=3*OG_INCH;
ch_wi:=3*OG_INCH;

/* Make the chart */
the_chart:=OG_MAKE_CHART(chpos, ch_ht, ch_wi,
the_template, the_query);

/* Define a field for ENAME */
indep_field.field_type:=OG_INDEPENDENT;
indep_field.colname:='ENAME';

/* Insert it into the chart in the 0th position */
OG_INSERT_FIELD(the_chart, indep_field, 0);

/* Define a field for SAL */
dep_field.field_type:=OG_DEPENDENT;
dep_field.colname:='SAL';
dep_field.ftname:='column';

/* Insert it into the chart in the second position */
OG_INSERT_FIELD(the_chart, dep_field, 1);

/* Define as field for COMM */
dep_field.field_type:=OG_DEPENDENT;
dep_field.colname:='COMM';
dep_field.ftname:='column';

/* Insert it into the chart in the last position */
OG_INSERT_FIELD(the_chart, dep_field, OG_LAST);

/* Execute the query for the first time*/
OG_EXECUTE_QUERY(the_query);

/* Update the chart so the new field and query information takes effect */
OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);

END;


Accessing Chart Properties

There are two basic ways to access a chart's property sheet:
Using the Layout Editor

To access a chart’s property sheet in the Layout editor:
- Double-click on the chart in the Layout editor, or
- Select the chart in the Layout editor and then choose the Properties command from the pop-up menu.

Using the Object Navigator

To access a chart’s property sheet using the Object Navigator:
- Double-click on the chart object in the Object Navigator, or
- Select the chart object in the Object Navigator and choose the Properties command from the pop-up menu, or
- Select the chart object in the Object Navigator and choose Tools—>Properties.

Modifying Chart Properties

You can specify a different query, and chart template to associate with a chart, as well as other options for the chart as a whole. For example, you may want to name a chart in order to reference it programmatically. You may also want to associate a chart with a button procedure.

To modify chart properties:
1. Double-click on the chart to show the Chart property sheet.
2. Specify the desired options.
3. Choose the OK button to apply the changes and close the Chart property sheet, or choose the Apply button to apply your changes to the chart without closing the Chart property sheet.

Filtering Chart Data

When you want to restrict the amount of data plotted in a chart, you normally filter the data by specifying a WHERE clause in your query (e.g., “WHERE DEPTNO = :DNO”). However, this method requires that each chart be assigned its own unique query. In addition, re-executing a query every time its chart is updated can negatively impact your chart’s performance.

To avoid this problem, you can create a query filter on a chart. Query filters are PL/SQL functions that restrict the amount of data plotted in a chart.
Using the Layout Editor

Using the Object Navigator

Filtering Chart Data

8 – 11 Using Charts

• through the Layout editor
• through the Object Navigator

To access a chart’s property sheet in the Layout editor:

n Double-click on the chart in the Layout editor,

n Select the chart in the Layout editor and then choose the Properties command from the pop-up menu.

To access a chart’s property sheet using the Object Navigator:

n Double-click on the chart object in the Object Navigator,

n Select the chart object in the Object Navigator and choose the Properties command from the pop-up menu,

n Select the chart object in the Object Navigator and choose Tools—>Properties.

Modifying Chart Properties

You can specify a different query, and chart template to associate with a chart, as well as other options for the chart as a whole.  For example, you may want to name a chart in order to reference it programmatically.  You may also want to associate a chart with a button procedure.

To modify chart properties:

1. Double-click on the chart to show the Chart property sheet.
2. Specify the desired options.
3. Choose the OK button to apply the changes and close the Chart property sheet, or choose the Apply button to apply your changes to the chart without closing the Chart property sheet.

When you want to restrict the amount of data plotted in a chart, you normally filter the data by specifying a WHERE clause in your query (e.g., "WHERE DEPTNO = :DNO").  However, this method requires that each chart be assigned its own unique query.  In addition, re-executing a query every time its chart is updated can negatively impact your chart’s performance.

To avoid this problem, you can create a query filter on a chart.  Query filters are PL/SQL functions that restrict the amount of data plotted in a chart.
Because each chart can be assigned its own query filter, a single query—executed only once—can provide the data used by multiple charts.

The query filter function reads through each row in your query and evaluates that row of data according to the condition(s) you set in the query filter itself. Each row that meets the condition(s) is plotted. Each row that does not meet the condition(s) is not plotted but still remains in the Data property page of the Query property sheet.

Creating a Query Filter

Follow the steps below to create a simple query filter function on an existing chart.

For more information about creating charts, see “Creating Charts” on page 8–8.

1. Decide what condition you wish to set on the query data.
   You may want to set a maximum limit on a column, compare values of multiple columns, or select a specific range of allowable data.

2. Double-click on the chart to show the Chart property sheet.

3. Choose the Data tab to show the Data property page.

4. Choose the New button that appears to the right of the Filter Function drop-down list.
   
   The PL/SQL Program Unit editor appears, containing a default query filter function template (initially named OGQUERYFILTER0):

   ```pl/sql
   -- Query Filter Functions. Called for each row of a
   -- query that is associated with the chart object.
   -- ARGUMENTS:
   --   CHARTOBJ The current chart object
   --   QUERY    The query associated with the chart object
   -- RETURN:
   --   TRUE   keep the row
   --   FALSE  remove the row
   FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query) RETURN BOOLEAN IS
   BEGIN
   END;
   
   You will need to supply a condition based on one or more columns in your query.

5. Define the condition to filter by.
Creating a Query Filter

Because each chart can be assigned its own query filter, a single query—executed only once—can provide the data used by multiple charts.

The query filter function reads through each row in your query and evaluates that row of data according to the condition(s) you set in the query filter itself. Each row that meets the condition(s) is plotted. Each row that does not meet the condition(s) is not plotted but still remains in the Data property page of the Query property sheet.

Follow the steps below to create a simple query filter function on an existing chart.

For more information about creating charts, see “Creating Charts” on page 8–8.

1. Decide what condition you wish to set on the query data. You may want to set a maximum limit on a column, compare values of multiple columns, or select a specific range of allowable data.

2. Double-click on the chart to show the Chart property sheet.

3. Choose the Data tab to show the Data property page.

4. Choose the New button that appears to the right of the Filter Function drop-down list.

The PL/SQL Program Unit editor appears, containing a default query filter function template (initially named OGQUERYFILTER0):

```plsql
-- Query Filter Functions. Called for each row of a query that is associated with the chart object.
-- ARGUMENTS:
--   CHARTOBJ  The current chart object
--   QUERY    The query associated with the chart object
-- RETURN:
--   TRUE   keep the row
--   FALSE  remove the row
FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query) RETURN BOOLEAN IS
BEGIN
END;
```

You will need to supply a condition based on one or more columns in your query.

5. Define the condition to filter by.
The following example uses the OG_GET_NUMCELL function to retrieve the value of the DEPTNO column in the current query. The condition states that if that value is 10, RETURN TRUE (plot the row), otherwise, RETURN FALSE (do not plot the row).

```plsql
FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query) RETURN BOOLEAN IS
BEGIN
    IF OG_GET_NUMCELL(query,OG_NEWDATA,'deptno')=10 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
```

6. Choose the Compile button in the Program Unit editor to compile the query filter.

   If you encounter any errors, correct them and re-compile. The query filter must be successfully compiled before it can be used.

7. Choose the Close button to close the Program Unit editor.

8. Choose the OK button to close the Chart property sheet.

   You can choose Chart—>Update Chart to see the results of the query filter function on your chart.

   Although the query selects every row from the table, the query filter ensures that only certain rows are charted.

For more information about setting conditions using PL/SQL, see chapter “Control Structures” of the PL/SQL User’s Guide and Reference.

For a lesson on how to create a sample query filter, see “Lesson 8: Create a Query Filter” on page 2 – 26 in chapter “Tutorial”.

**Creating a Break Chart**

Break charts allow you to specify a subcategory of data along the category axis. This feature greatly facilitates the task of charting groups of columns retrieved by a query.

For example, suppose you wanted to chart the revenue brought in by each sales representative per quarter. You could choose the sales representative as the category, the quarter (Q1, Q2, etc.) as the detail (sub)category, and revenue as the value. For each sales representative, you would see four bars—one for each quarter.

Follow the steps below to create a break chart:

1. Create a column or bar chart or open the Chart property sheet for an existing column or bar chart.
Creating a Break Chart

8 – 13

The following example uses the OG_GET_NUMCELL function to retrieve the value of the DEPTNO column in the current query. The condition states that if that value is 10, RETURN TRUE (plot the row), otherwise, RETURN FALSE (do not plot the row).

FUNCTION OGQUERYFILTER0(chartobj IN og_object, query IN og_query) RETURN BOOLEAN IS
BEGIN
  IF OG_GET_NUMCELL(query, OG_NEWDATA, 'deptno') = 10 THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END;

6. Choose the Compile button in the Program Unit editor to compile the query filter. If you encounter any errors, correct them and re-compile. The query filter must be successfully compiled before it can be used.

7. Choose the Close button to close the Program Unit editor.

8. Choose the OK button to close the Chart property sheet. You can choose Chart —> Update Chart to see the results of the query filter function on your chart.

Although the query selects every row from the table, the query filter ensures that only certain rows are charted.

For more information about setting conditions using PL/SQL, see chapter "Control Structures" of the PL/SQL User's Guide and Reference.

For a lesson on how to create a sample query filter, see "Lesson 8: Create a Query Filter" on page 2 – 26 in chapter "Tutorial".

Break charts allow you to specify a subcategory of data along the category axis. This feature greatly facilitates the task of charting groups of columns retrieved by a query.

For example, suppose you wanted to chart the revenue brought in by each sales representative per quarter. You could choose the sales representative as the category, the quarter (Q1, Q2, etc.) as the detail (sub)category, and revenue as the value. For each sales representative, you would see four bars—one for each quarter.

Follow the steps below to create a break chart:

1. Create a column or bar chart or open the Chart property sheet for an existing column or bar chart.
Note that the query for a break chart must contain an ORDER BY clause to ensure that the data is ordered correctly. The column specified in the ORDER BY clause should be the same column as the category plotted on the chart axis.

2. Choose the Categories tab to show the Categories property page.
3. Make sure only one category is listed in the Category list.
4. Select a sub-category from the Sub-category drop-down list.
5. Choose the OK button to apply the change and close the Chart property sheet, or choose the Apply button to apply the changes without closing the Chart property sheet.

For a lesson on how to create a sample break chart, see “Lesson 7: Create a Break Chart” on page 2–24 in chapter “Tutorial”.

### Assign Category and Value Columns

When you first create a chart, Graphics maps the query columns to the categories and values plotted in the chart. However, you may need to change the category and value columns whenever you change information the chart is currently using—specifically, when you do any of the following:

- change column names by adding aliases in the query
- delete a column from the query
- add a sub-category to the chart
- rename or delete a field template
- add a field template to be used

To modify chart category and value columns:

1. Double-click on the chart to show the Chart property sheet.
2. Select the Categories tab to show the Categories property page, or Select the Values tab to show the Values property page.
3. Change options as follows:
   - To add to the list of chart categories or values, select a column from the Query Columns list and then choose the Insert button.
   - To change the order in which chart categories and values are plotted, select the category or value column and then choose either the Top (to the top of the list), Bottom (to the bottom of the list), Up (up one item), or Down (down one item) buttons.
Note that the query for a break chart must contain an ORDER BY clause to ensure that the data is ordered correctly. The column specified in the ORDER BY clause should be the same column as the category plotted on the chart axis.

2. Choose the Categories tab to show the Categories property page.
3. Make sure only one category is listed in the Category list.
4. Select a sub-category from the Sub-category drop-down list.
5. Choose the OK button to apply the change and close the Chart property sheet, or choose the Apply button to apply the changes without closing the Chart property sheet.

For a lesson on how to create a sample break chart, see “Lesson 7: Create a Break Chart” on page 2–24 in chapter “Tutorial.”

When you first create a chart, Graphics maps the query columns to the categories and values plotted in the chart. However, you may need to change the category and value columns whenever you change information the chart is currently using—specifically, when you do any of the following:

• change column names by adding aliases in the query
• delete a column from the query
• add a sub-category to the chart
• rename or delete a field template
• add a field template to be used

To modify chart category and value columns:
1. Double-click on the chart to show the Chart property sheet.
2. Select the Categories tab to show the Categories property page, or select the Values tab to show the Values property page.
3. Change options as follows:
   – To add to the list of chart categories or values, select a column from the Query Columns list and then choose the Insert button.
   – To change the order in which chart categories and values are plotted, select the category or value column and then choose either the Top (to the top of the list), Bottom (to the bottom of the list), Up (up one item), or Down (down one item) buttons.
– To delete a category or value column from its respective list, select the column and then choose the Delete button.

**Note:** Deleting a category or value column from the list does not delete the column from the query.

– To associate a chart value column with a field template, select the value column from the list and then choose a field template from the Field Template drop-down list.

4. Choose the OK button to apply the changes and close the Chart property sheet.

**Customizing Charts**

You can customize charts in the following ways:

- set a baseline value
- set shadow and depth for bars or columns
- hide or show the legend
- add custom axis labels
- specify an axis data type
- set tick labels
- hide or show the chart grid
- move the legend
- use reference lines
- specify number and date formats
- set text and line properties
- set colors and patterns

You can perform most of these same actions on the chart template if you wish to apply the properties to multiple charts. For more information, see “Using Chart Templates” on page 8 – 1.

**Specifying Axis Frame Properties**

You can modify the appearance of axis charts using the Frame property sheet. Frame properties include baseline values, category width, depth size, shadow size, and shadow direction. In addition, you can use the Frame property sheet to hide/show the legend.
To delete a category or value column from its respective list, select the column and then choose the Delete button. Note: Deleting a category or value column from the list does not delete the column from the query.

To associate a chart value column with a field template, select the value column from the list and then choose a field template from the Field Template drop-down list.

4. Choose the OK button to apply the changes and close the Chart property sheet.

Customizing Charts

You can customize charts in the following ways:

• set a baseline value
• set shadow and depth for bars or columns
• hide or show the legend
• add custom axis labels
• specify an axis data type
• set tick labels
• hide or show the chart grid
• move the legend
• use reference lines
• specify number and date formats
• set text and line properties
• set colors and patterns

You can perform most of these same actions on the chart template if you wish to apply the properties to multiple charts. For more information, see "Using Chart Templates" on page 8 – 1.

You can modify the appearance of axis charts using the Frame property sheet. Frame properties include baseline values, category width, depth size, shadow size, and shadow direction. In addition, you can use the Frame property sheet to hide/show the legend.
Set a Baseline Value

By default, an axis chart’s baseline is set to the minimum value returned by your query for the value column (usually the X axis). You can change the baseline setting to reflect a different point of reference.

To change the baseline value of the axis chart:

1. Select the chart or a single chart object such as the chart title.
2. Choose the Frame command from the pop-up menu, or choose Chart—>Frame to show the Frame property sheet.
3. Choose the Axis Frame tab to show the Axis Frame property page.
4. Specify the desired options for the baseline value.
5. Choose the OK button to apply the changes to the chart.

Set Shadow and Depth for Bar or Columns

To specify depth size, shadow size, and shadow direction for an axis chart:

1. Select the chart or a single chart object.
2. Choose the Frame command from the pop-up menu, or choose Chart—>Frame to show the Frame property sheet.
3. If necessary, choose the Frame tab to show the Frame property page.
4. Choose one of the options for depth size, shadow size, or shadow direction.
5. Choose the OK button to apply the change and close the Frame property sheet.

Whether you see this change reflected in the chart frame or field template depends on the type of chart you’ve selected.

Hide or Show Legend

A chart legend appears by default. You can choose to hide the legend and later show it again. Additionally, you can split the legend into multiple columns:

1. Select the chart or a single chart object such as the chart title.
2. Choose the Frame command from the pop-up menu, or choose Chart—>Frame to show the Frame property sheet.
3. Click the Show Legend checkbox to toggle it on or off.

When the checkbox is selected, the legend will appear with the chart. When the checkbox is empty (deselected), the legend will not appear.

4. Choose the OK button to apply the changes to the frame and close the Frame property sheet.
Set a Baseline Value

By default, an axis chart's baseline is set to the minimum value returned by your query for the value column (usually the X axis). You can change the baseline setting to reflect a different point of reference.

To change the baseline value of the axis chart:
1. Select the chart or a single chart object such as the chart title.
2. Choose the Frame command from the pop-up menu, or choose Chart —> Frame to show the Frame property sheet.
3. Choose the Axis Frame tab to show the Axis Frame property page.
4. Specify the desired options for the baseline value.
5. Choose the OK button to apply the changes to the chart.

To specify depth size, shadow size, and shadow direction for an axis chart:
1. Select the chart or a single chart object.
2. Choose the Frame command from the pop-up menu, or choose Chart —> Frame to show the Frame property sheet.
3. If necessary, choose the Frame tab to show the Frame property page.
4. Choose one of the options for depth size, shadow size, or shadow direction.
5. Choose the OK button to apply the change and close the Frame property sheet.

Whether you see this change reflected in the chart frame or field template depends on the type of chart you've selected.

A chart legend appears by default. You can choose to hide the legend and later show it again. Additionally, you can split the legend into multiple columns:
1. Select the chart or a single chart object such as the chart title.
2. Choose the Frame command from the pop-up menu, or choose Chart —> Frame to show the Frame property sheet.
3. Click the Show Legend checkbox to toggle it on or off.
   - When the checkbox is selected, the legend will appear with the chart. When the checkbox is empty (deselected), the legend will not appear.
4. Choose the OK button to apply the changes to the frame and close the Frame property sheet.
Specifying Axis Properties

For axis charts, axis properties include options specific to the type of axis you select: discrete, continuous, or date. You can access the Axis property sheet by selecting the chart in the Layout editor and then selecting Axes from the pop-up menu.

Add Custom Axis Labels

By default, the axis labels displayed with your chart are based on the column names selected in your query. You can customize axis labels by specifying columns aliases in your query, or by setting them in the Axis property sheet.

To set custom axis labels in the Axis properties sheet:

1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart→Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Type a name in the Custom Label field.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.

Specify an Axis Data Type

An axis can be one of the following types: discrete, continuous, or date. In general, the axis that displays your category (independent) data is of type discrete. The axis that displays your value (or dependent) data, is of type continuous.

To set a data type for an axis:

1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart→Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Select a data type from the Data Type drop-down list.
   
   **Note:** If the Axis type you select is not compatible with the datatype of the column plotted on that axis, you will receive an alert prompting you to confirm your choice.

6. Choose the OK button to apply the changes to the selected axis.
Using Charts

For axis charts, axis properties include options specific to the type of axis you select: discrete, continuous, or date. You can access the Axis property sheet by selecting the chart in the Layout editor and then selecting Axes from the pop-up menu.

By default, the axis labels displayed with your chart are based on the column names selected in your query. You can customize axis labels by specifying columns aliases in your query, or by setting them in the Axis property sheet.

To set custom axis labels in the Axis properties sheet:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart —> Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Type a name in the Custom Label field.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.

An axis can be one of the following types: discrete, continuous, or date. In general, the axis that displays your category (independent) data is of type discrete. The axis that displays your value (or dependent) data, is of type continuous.

To set a data type for an axis:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart —> Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Select a data type from the Data Type drop-down list.
6. Choose the OK button to apply the changes to the selected axis.

Note: If the Axis type you select is not compatible with the datatype of the column plotted on that axis, you will receive an alert prompting you to confirm your choice.
You can toggle to the other axis by selecting it from the Axis drop-down list.

**Set Tick Label Properties**

You can set a variety of tick label properties such as position on the axis, and rotation of labels. In addition, you can hide or show the ticks or tick labels.

To set tick and tick label properties:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart→Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Select tick and tick label options from the radio groups and check boxes.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.

**Hide or Show Chart Grid**

The chart grid provides major and minor reference lines at consistent intervals along both chart axes. You can hide or show major or minor grid lines for either or both axes.

To hide or show chart grids:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart→Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Click the Show Major Grid and Show Minor Grid check boxes to toggle them on or off.
   When the checkbox is selected, the grid will appear with the chart. When the checkbox is empty (deselected), the grid will not appear.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.
You can toggle to the other axis by selecting it from the Axis drop-down list.

You can set a variety of tick label properties such as position on the axis, and rotation of labels. In addition, you can hide or show the ticks or tick labels.

To set tick and tick label properties:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart —> Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Select tick and tick label options from the radio groups and check boxes.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.

The chart grid provides major and minor reference lines at consistent intervals along both chart axes. You can hide or show major or minor grid lines for either or both axes.

To hide or show chart grids:
1. Select the chart or a single chart object such as one of the axes.
2. Choose the Axes command from the pop-up menu, or choose Chart —> Axes to show the Axis property sheet.
3. If not already displayed, click the Axis tab to show the Axis property page.
4. Choose an axis (e.g., X, Y1) from the Axis drop-down list.
5. Click the Show Major Grid and Show Minor Grid check boxes to toggle them on or off. When the checkbox is selected, the grid will appear with the chart. When the checkbox is empty (deselected), the grid will not appear.
6. Choose the OK button to apply the changes to the selected axis.

You can toggle to the other axis by selecting it from the Axis drop-down list.
Move the Legend

To move the legend to a new location:

- Click-and-drag the legend frame to a new location.

If necessary, choose Update from the pop-up menu, or choose **Chart**—>**Update Chart** to update the chart’s appearance.

Use Reference Lines

You can add a reference line that represents a significant value on the chart frame. Once you’ve added a reference line to a chart, you can edit the reference line properties, or delete the reference line from the chart. You can move a reference line by editing its value in the Reference Line property sheet.

To add a reference line:

1. Select the chart or a single chart object.
2. Choose the Reference Lines command from the pop-up menu, or choose **Chart**—>**Reference Lines** to show the Reference Line property sheet.
   
   If no reference lines currently exist, an alert appears asking if you want to create a new one. If you choose the Yes button, a new reference line is automatically created for you in the Reference Line property sheet.

3. Choose the New button in the property sheet to create a new reference line.
4. Accept the default legend label and value, or modify their settings.
5. Choose the OK button to draw the new reference line on the chart frame and close the Reference Line property sheet.

To edit reference line properties:

1. Select the chart or a single chart object.
2. Choose the Reference Lines command from the pop-up menu, or choose **Chart**—>**Reference Lines** to show the Reference Line property sheet.

3. Choose the reference line whose properties you wish to edit from the Reference Line drop-down list.
4. Specify the desired options.
5. Choose the OK button to apply the changes to the reference line and close the Reference Line property sheet.

To delete a reference line:

1. Select the chart or a single chart object.
To move the legend to a new location:

1. Click-and-drag the legend frame to a new location.
2. If necessary, choose Update from the pop-up menu, or choose Chart —> Update Chart to update the chart's appearance.

You can add a reference line that represents a significant value on the chart frame. Once you've added a reference line to a chart, you can edit the reference line properties, or delete the reference line from the chart.

You can move a reference line by editing its value in the Reference Line property sheet.

To add a reference line:

1. Select the chart or a single chart object.
2. Choose the Reference Lines command from the pop-up menu, or choose Chart —> Reference Lines to show the Reference Line property sheet.
3. If no reference lines currently exist, an alert appears asking if you want to create a new one. If you choose the Yes button, a new reference line is automatically created for you in the Reference Line property sheet.
4. Choose the New button in the property sheet to create a new reference line.
5. Accept the default legend label and value, or modify their settings.
6. Choose the OK button to draw the new reference line on the chart frame and close the Reference Line property sheet.

To edit reference line properties:

1. Select the chart or a single chart object.
2. Choose the Reference Lines command from the pop-up menu, or choose Chart —> Reference Lines to show the Reference Line property sheet.
3. Choose the reference line whose properties you wish to edit from the Reference Line drop-down list.
4. Specify the desired options.
5. Choose the OK button to apply the changes to the reference line and close the Reference Line property sheet.

To delete a reference line:

1. Select the chart or a single chart object.
Specify Number and Date Formats

You can set number and date formats in the following ways depending on your needs:

- specify the format in your preferences file for all charts in all sessions
- set the format directly in the Layout editor for an individual chart
- set the format for a specific chart template in the Chart Template editor

You will need to set Date formats in your preferences file if you are using date data from external files such as PRN files. This ensures that any dates referenced in those external files will map correctly into dates in the chart. If a matching date format is not specified in your preferences file, Graphics treats the external date as character data.

Setting Number and Date Formats in Your Preferences File

You can set one or more number or date formats in your preferences file. When you create a chart, the last format added to the format list is used. To set the Number and Date formats for all charts and all Designer sessions:

1. Choose Tools—>Tools Options to show the Tools Options property sheet.
2. Click on either the Number or Date button to show the Number Format or Date Format dialog.
3. Enter your preferred date or number format in the Format field.
   For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.
4. Choose the Add button to add the format to the list.
5. Choose the OK button to accept your changes to the format list and close the dialog.

2. Choose Reference Lines from the pop-up menu, or choose Chart—>Reference Lines to show the Reference Line property sheet.

3. Choose the reference line you wish to delete from the Reference Line drop-down list.

4. Choose the Delete button
   An alert appears, prompting you to confirm the deletion.

5. Choose the Yes button to confirm.

6. Choose the OK button to apply the change and close the Reference Line property sheet.
Specify Number and Date Formats

Setting Number and Date Formats in Your Preferences File

You can set number and date formats in the following ways depending on your needs:

- specify the format in your preferences file for all charts in all sessions
- set the format directly in the Layout editor for an individual chart
- set the format for a specific chart template in the Chart Template editor

You will need to set Date formats in your preferences file if you are using date data from external files such as PRN files. This ensures that any dates referenced in those external files will map correctly into dates in the chart. If a matching date format is not specified in your preferences file, Graphics treats the external date as character data.

You can set one or more number or date formats in your preferences file. When you create a chart, the last format added to the format list is used.

To set the Number and Date formats for all charts and all Designer sessions:

1. Choose Tools —> Tools Options to show the Tools Options property sheet.
2. Click on either the Number or Date button to show the Number Format or Date Format dialog.
3. Enter your preferred date or number format in the Format field.
4. Choose the Add button to add the format to the list.
5. Choose the OK button to accept your changes to the format list and close the dialog.
6. Choose the Save button to save your settings to your preferences file, or choose the OK button to save your settings for the current session only.

Setting Number and Date Formats in the Chart Template

To set the Number and Date formats for a specific chart template:
1. Choose Tools—>Templates to show the Chart Template editor.
2. Select the template you wish to set a number or date format for in the Template drop-down list.
   The template appears in the editor’s layout.
3. Select the chart template element that represents the number or date format you wish to set.
4. Choose Format—>Number or Format—>Date to show the Number Format or Date Format dialog.
   The current number or date format appears in the Format field.
5. You can edit the current format, select an alternate format from the list, or enter a new format and add it to the list.
   If you want to add a format to the list permanently, see the section above on setting number and date formats in your preferences file.
   For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.
6. Choose the OK button to accept your changes to the format list and close the dialog.
7. Close the Chart Template editor.
   Your changes are reflected in all charts that use the template you edited.

Setting Number and Date Formats in the Layout Editor

To set Number and Date formats directly on the chart in the Layout editor:
1. Select the chart element whose number or date format you wish to edit.
   All elements of that set are selected.
2. Choose Number Format or Date Format from the pop-up menu, or choose Format—>Number or Format—>Date to show the Number Format or Date Format dialog.
3. Choose the Date button or the Number button to show the Date Format dialog box or Number Format dialog box, respectively.
   The current number or date format appears in the Format field.
To set the Number and Date formats for a specific chart template:

1. Choose **Tools** → **Templates** to show the Chart Template editor.
2. Select the template you wish to set a number or date format for in the Template drop-down list. The template appears in the editor's layout.
3. Select the chart template element that represents the number or date format you wish to set.
4. Choose **Format** → **Number** or **Format** → **Date** to show the Number Format or Date Format dialog.
   - The current number or date format appears in the Format field.
5. You can edit the current format, select an alternate format from the list, or enter a new format and add it to the list.
   - If you want to add a format to the list permanently, see the section above on setting number and date formats in your preferences file.
   - For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.
6. Choose the OK button to accept your changes to the format list and close the dialog.
7. Close the Chart Template editor.
   - Your changes are reflected in all charts that use the template you edited.

To set Number and Date formats directly on the chart in the Layout editor:

1. Select the chart element whose number or date format you wish to edit.
2. All elements of that set are selected.
3. Choose Number Format or Date Format from the pop-up menu, or choose **Format** → **Number** or **Format** → **Date** to show the Number Format or Date Format dialog.
4. Choose the Date button or the Number button to show the Date Format dialog box or Number Format dialog box, respectively.
   - The current number or date format appears in the Format field.
4. You can edit the current format, select an alternate format from the list, or enter a new format and add it to the list.

   If you want to add a format to the list permanently, see the section above on setting number and date formats in your preferences file.

   For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.

5. Choose the OK button to accept your changes to the format list and close the dialog.

   Your changes are reflected in all charts that use the template you edited.

   For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.

---

**Set Text and Line Properties**

The procedures for editing text properties (font, size, weight, style, spacing, and justification) and line properties (line, dash, arrow) for components of the chart frame are the same as those for objects.

For more information about setting text properties for objects in the layout, see “Creating Text and Text Fields” on page 4–9. For more information about setting line properties, see “Setting Line and Edge Properties” on page 4–29.

**Set Colors and Patterns**

The procedures for applying colors and patterns to components of the chart frame are the same as those for objects.

For more information about applying patterns and colors to chart elements in the layout, see “Setting Patterns and Color for Objects” on page 4–25.

---

**Working with Field Templates**

You can create multiple field templates for a chart if you want to assign different properties to value columns. Each field template can be assigned to a different value, assigned to multiple values, or remain unused until you need it. You can also delete field templates.

You can customize field templates in the following ways:

- adding and deleting field templates
- specifying field properties
- applying colors and patterns
You can edit the current format, select an alternate format from the list, or enter a new format and add it to the list. If you want to add a format to the list permanently, see the section above on setting number and date formats in your preferences file. For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.

Choose the OK button to accept your changes to the format list and close the dialog. Your changes are reflected in all charts that use the template you edited. For more information about valid formats, see the Oracle7 Server SQL Language Reference Manual.

The procedures for editing text properties (font, size, weight, style, spacing, and justification) and line properties (line, dash, arrow) for components of the chart frame are the same as those for objects. For more information about setting text properties for objects in the layout, see "Creating Text and Text Fields" on page 4 – 9. For more information about setting line properties, see "Setting Line and Edge Properties" on page 4 – 29.

The procedures for applying colors and patterns to components of the chart frame are the same as those for objects. For more information about applying patterns and colors to chart elements in the layout, see "Setting Patterns and Color for Objects" on page 4 – 25.

Working with Field Templates

You can create multiple field templates for a chart if you want to assign different properties to value columns. Each field template can be assigned to a different value, assigned to multiple values, or remain unused until you need it. You can also delete field templates.

You can customize field templates in the following ways:

• adding and deleting field templates
• specifying field properties
• applying colors and patterns
**Adding Field Templates**

To add a field template to the current chart:

1. Select the chart or a single chart object.
2. Choose Field Templates from the pop-up menu, or choose `Chart—>Field Templates` to show the Field Template property sheet.
3. Choose the New button to create a new field template, initially named `ftemp x`.
   You can rename the new field template or keep the default.
4. Choose the Axis Field tab to set properties for axis chart field templates.
5. Choose the OK button to apply the new field template and close the Field Templates dialog.

   **Note:** To see a field template appear in your chart, you must associate it with a value column. For more information about modifying value columns, see “Modifying Category and Value Columns” on page 8 – 14.

**Specifying Properties for Field Templates**

Field properties include options such as plot type, line style, and label rotation, depending on the chart type. To specify field properties for the field template:

1. Select the chart or a single chart object.
2. Choose the Field Templates command from the pop-up menu, or choose `Chart—>Field Templates` to show the Field Template property sheet.
3. Choose the Axis Field tab to show the Axis Field property page.
4. Specify the desired options.
5. Choose the OK button to apply the changes to the field template and close the Field Templates property sheet.

**Applying Colors and Patterns to Field Templates**

The procedures for applying colors and patterns to field templates are the same as those for objects.

For more information about applying patterns and colors to chart elements in the layout, see “Setting Patterns and Color for Objects” on page 4 – 25.
To add a field template to the current chart:

1. Select the chart or a single chart object.
2. Choose Field Templates from the pop-up menu, or choose Chart —> Field Templates to show the Field Template property sheet.
3. Choose the New button to create a new field template, initially named ftemp x. You can rename the new field template or keep the default.
4. Choose the Axis Field tab to set properties for axis chart field templates.
5. Choose the OK button to apply the new field template and close the Field Templates dialog.

Note: To see a field template appear in your chart, you must associate it with a value column. For more information about modifying value columns, see "Modifying Category and Value Columns" on page 8 – 14.

Field properties include options such as plot type, line style, and label rotation, depending on the chart type. To specify field properties for the field template:

1. Select the chart or a single chart object.
2. Choose the Field Templates command from the pop-up menu, or choose Chart —> Field Templates to show the Field Template property sheet.
3. Choose the Axis Field tab to show the Axis Field property page.
4. Specify the desired options.
5. Choose the OK button to apply the changes to the field template and close the Field Templates property sheet.

The procedures for applying colors and patterns to field templates are the same as those for objects. For more information about applying patterns and colors to chart elements in the layout, see "Setting Patterns and Color for Objects" on page 4 – 25.
Assigning Field Templates

Once you have created a field template and customized it, you can assign it to a value column in the chart. To assign a field template to a value column field:

1. Select the chart in the Layout editor or Object Navigator.
2. Choose Properties from the pop-up menu, or choose Tools—>Properties to show the chart’s property sheet.
3. Choose the Value tab to show the chart’s Value property page.
4. From the Chart Values list, select the value column to which you wish to assign a field template.
5. Select the field template from the Field Templates drop-down list.
6. Choose the OK button to apply your changes and close the Chart property sheet.

Deleting Field Templates

To delete a field template from a chart:

1. Select the chart or a single chart object.
2. Choose the Field Templates command from the pop-up menu, or choose Chart—>Field Templates to show the Field Template property sheet.
3. Select the field template you want to delete from the Field Template drop-down list.
4. Choose the Delete button.
   An alert appears, prompting you to confirm the deletion.
5. Choose the Yes button to confirm.
6. Choose the OK button to accept the changes and close the Field Template property sheet.

Note: If you’ve created a chart that contains a value column associated with the deleted field template, you must associate the value column with another field template before deleting it. For more information about modifying value columns, see “Modifying Chart and Value Columns” on page 8 – 14.
Once you have created a field template and customized it, you can assign it to a value column in the chart. To assign a field template to a value column field:

1. Select the chart in the Layout editor or Object Navigator.
2. Choose Properties from the pop-up menu, or choose Tools —> Properties to show the chart's property sheet.
3. Choose the Value tab to show the chart's Value property page.
4. From the Chart Values list, select the value column to which you wish to assign a field template.
5. Select the field template from the Field Templates drop-down list.
6. Choose the OK button to apply your changes and close the Chart property sheet.

To delete a field template from a chart:

1. Select the chart or a single chart object.
2. Choose the Field Templates command from the pop-up menu, or choose Chart —> Field Templates to show the Field Template property sheet.
3. Select the field template you want to delete from the Field Template drop-down list.
4. Choose the Delete button. An alert appears, prompting you to confirm the deletion.
5. Choose the Yes button to confirm.
6. Choose the OK button to accept the changes and close the Field Template property sheet.

Note: If you've created a chart that contains a value column associated with the deleted field template, you must associate the value column with another field template before deleting it. For more information about modifying value columns, see "Modifying Chart and Value Columns" on page 8 – 14.
Updating Charts

When the data in the Query property sheet changes, or you’ve modified a chart’s elements or function in some way, you need to update the chart to reflect those changes. There are three ways to update a chart:

- automatically
- manually
- using PL/SQL

Updating Charts

To update your charts automatically:

- On the Data property page of the Chart property sheet, activate Update chart on query execution by selecting its check box. Your chart will be updated whenever its query is executed.

Updating Charts

To update your charts manually:

1. Select the chart you want to update.
2. Choose Update from the pop-up menu or Chart—>Update Chart.

Graphics re-draws the chart based on the latest information.

Updating Charts Using

PL/SQL

You can also update a chart using PL/SQL. You could assign the following procedure to a timer trigger to be executed at a specified interval. This example updates the specified chart to reflect new query results or new properties that have been applied to chart elements.

```
PROCEDURE my_timer IS
  my_query   og_query;
  my_chart   og_object;
BEGIN
  my_query:=og_get_query('emp_query');
  my_chart:=og_get_object('emp_chart');
  og_execute_query(my_query);
  og_update_chart(my_chart, OG_ALL_CHUPDA);
END;
```

For more information about timers, see “Using Timer Triggers” on page 11–9 in chapter “Using Triggers”.

Updating Charts

Manually

Using PL/SQL

When the data in the Query property sheet changes, or you've modified a chart's elements or function in some way, you need to update the chart to reflect those changes. There are three ways to update a chart:

- **automatically**
- **manually**
- **using PL/SQL**

To update your charts automatically:

1. On the Data property page of the Chart property sheet, activate Update chart on query execution by selecting its check box. Your chart will be updated whenever its query is executed.

To update your charts manually:

1. Select the chart you want to update.
2. Choose Update from the pop-up menu or Chart —> Update Chart. Graphics re-draws the chart based on the latest information.

You can also update a chart using PL/SQL. You could assign the following procedure to a timer trigger to be executed at a specified interval. This example updates the specified chart to reflect new query results or new properties that have been applied to chart elements.

```plsql
PROCEDURE my_timer IS
  my_query   og_query;
  my_chart   og_object;
BEGIN
  my_query:=og_get_query('emp_query');
  my_chart:=og_get_object('emp_chart');
  og_execute_query(my_query);
  og_update_chart(my_chart, OG_ALL_CHUPDA);
END;
```

For more information about timers, see “Using Timer Triggers” on page 11 – 9 in chapter “Using Triggers”.

Working with Drill-down Charts

Using Graphics, you can establish drill-down relationships between “master” and “detail” charts in a display. At runtime, users can click on an element within the master chart to “drill down” to detailed information about that element. For example, in a pie chart showing company expenses for each department, you could click on a pie slice to show detailed information about that department.

Drill-down relationships are based on parameters. The master chart sets the parameter value, and the detail chart uses this value in its query. When the user clicks on a chart element in the master chart, Graphics does the following:

- determines the data value represented by the chart element
- assigns the value to a parameter
- executes the detail query that references the parameter
- updates the detail chart

Creating a Drill-down Relationship

Follow the steps below to create a drill-down relationship between two charts.

1. Select the master chart. This is the chart that will “drive” the drill-down action.
2. Select the chart element that you want to use to determine the value of the parameter. This can be a tick label, a pie slice, etc.
   Remember, the parameter represented by the chart element will be used in the query for the detail chart. The parameter establishes the master–detail relationship.
3. Open the property sheet of the chart element you have selected. (double-click on a selected chart element or choose Properties from the pop-up menu, or choose Tools—>Properties).
4. Choose the Drill-down tab to show the Drill-down property page.
5. Choose the New button to create a new parameter.
   The Parameters dialog appears.
6. Accept the default parameter name, PARAMn, or type a new name in the Name field.
7. From the Type drop-down list, select a data type (CHAR, NUMBER, or DATE).
   The datatype you select for the parameter should match the datatype of the column the parameter will represent.
Using Graphics, you can establish drill-down relationships between "master" and "detail" charts in a display. At runtime, users can click on an element within the master chart to "drill down" to detailed information about that element. For example, in a pie chart showing company expenses for each department, you could click on a pie slice to show detailed information about that department.

Drill-down relationships are based on parameters. The master chart sets the parameter value, and the detail chart uses this value in its query. When the user clicks on a chart element in the master chart, Graphics does the following:

• determines the data value represented by the chart element
• assigns the value to a parameter
• executes the detail query that references the parameter
• updates the detail chart

Follow the steps below to create a drill-down relationship between two charts.

1. Select the master chart. This is the chart that will "drive" the drill-down action.
2. Select the chart element that you want to use to determine the value of the parameter. This can be a tick label, a pie slice, etc. Remember, the parameter represented by the chart element will be used in the query for the detail chart. The parameter establishes the master–detail relationship.
3. Open the property sheet of the chart element you have selected. (double-click on a selected chart element or choose Properties from the pop-up menu, or choose Tools —> Properties).
4. Choose the Drill-down tab to show the Drill-down property page.
5. Choose the New button to create a new parameter. The Parameters dialog appears.
6. Accept the default parameter name, PARAM_n, or type a new name in the Name field.
7. From the Type drop-down list, select a data type (CHAR, NUMBER, or DATE). The datatype you select for the parameter should match the datatype of the column the parameter will represent.
8. (Optional) Type an initial value for the parameter in the Initial Value field.

9. Select OK to accept the new parameter.
   The Parameters dialog closes and you return to the Object properties sheet.

10. From the To Value Of drop-down list, select the column name that represents the chart element you selected to be the value source for the parameter.

11. From the Execute Query drop-down list, select the query that will update the drill-down detail chart.

12. Choose the Edit button so you can add the parameter to the query.
   The Query property sheet appears.

13. Edit the query to include the parameter you have created. For example, you might add a WHERE clause to the existing query such as:

   \[ \text{WHERE column1 := :PARAM0} \]

14. Select OK to accept the edits to the query and return to the Object property sheet.

15. Select OK to accept the drill-down procedure.

16. Save and run the display to test the procedure. Try clicking on the chart element in your master chart and watch the result as the detail chart is updated with the resulting query data.

Notice that the above procedure did not involve any PL/SQL code. You can use the Object property sheet to create a drill-down event for any object in your display.

For more information about creating charts, see “Creating Charts” on page 8 – 8.

For a lesson on how to create a drill-down chart, see the chapter “Tutorial” on page 2 – 1.

**Modifying a Drill-down Relationship**

Just as there is no direct programming involved in creating a drill-down relationship between two charts, there is no programming involved in modifying a drill-down relationship either.

You may wish to specify that a different query is executed, or set the parameter value to a different column.

1. Open the Object property sheet for the chart element that drives the drill-down procedure.
Modifying a Drill-down Relationship

8. (Optional) Type an initial value for the parameter in the Initial Value field.
9. Select OK to accept the new parameter. The Parameters dialog closes and you return to the Object properties sheet.
10. From the To Value Of drop-down list, select the column name that represents the chart element you selected to be the value source for the parameter.
11. From the Execute Query drop-down list, select the query that will update the drill-down detail chart.
12. Choose the Edit button so you can add the parameter to the query. The Query property sheet appears.
13. Edit the query to include the parameter you have created. For example, you might add a WHERE clause to the existing query such as:
   WHERE column1 := :PARAM0
14. Select OK to accept the edits to the query and return to the Object property sheet.
15. Select OK to accept the drill-down procedure.
16. Save and run the display to test the procedure. Try clicking on the chart element in your master chart and watch the result as the detail chart is updated with the resulting query data.

Notice that the above procedure did not involve any PL/SQL code. You can use the Object property sheet to create a drill-down event for any object in your display.

For more information about creating charts, see "Creating Charts" on page 8-8.

For a lesson on how to create a drill-down chart, see the chapter "Tutorial" on page 2-1.

Just as there is no direct programming involved in creating a drill-down relationship between two charts, there is no programming involved in modifying a drill-down relationship either.

You may wish to specify that a different query is executed, or set the parameter value to a different column.

1. Open the Object property sheet for the chart element that drives the drill-down procedure.
2. Select the Drill-down tab to show the Drill-down property page.
3. Select your new options.
4. Choose OK to accept the new drill-down relationship and close the Object property sheet.

If you wish to modify the parameter itself, you can either access it through the Object property page by choosing the Edit button to show the Parameters dialog, or through the Object Navigator.

For more information about parameters, see “Using Parameters” on page 10 – 1.

**Converting Charts to Artwork**

Converting a chart to artwork separates the chart from its data source. You may wish to re-execute the chart’s query, or update the chart before converting it to artwork. This ensures that the chart reflects the latest data.

To convert a dynamic chart into a static artwork object:
1. Select the chart you want to convert to artwork.
2. Choose **Chart—>Convert to Artwork**.

Graphics converts the chart to a group object composed of shapes which you can manipulate individually. As an artwork object, the chart can no longer be updated.

**Renaming Charts**

You can change the name of a chart either in the Object Navigator, or in the Chart property sheet.

**Note:** If you rename a chart that is referenced in a PL/SQL program unit, you must change those references to reflect the new chart name.

**Renaming Charts in the Object Navigator**

You can change the name of your chart directly in the Object Navigator.

To rename a chart in the Object Navigator:
1. Select the chart object you wish to rename.
2. Move the mouse pointer over the chart name.
   The cursor changes from a pointer to a text editor.
Renaming Charts in the Object Navigator

1. Select the Drill-down tab to show the Drill-down property page.
2. Select your new options.
3. Choose OK to accept the new drill-down relationship and close the Object property sheet.
4. If you wish to modify the parameter itself, you can either access it through the Object property page by choosing the Edit button to show the Parameters dialog, or through the Object Navigator.

For more information about parameters, see "Using Parameters" on page 10–1.

Converting Charts to Artwork

Converting a chart to artwork separates the chart from its data source. You may wish to re-execute the chart’s query, or update the chart before converting it to artwork. This ensures that the chart reflects the latest data.

To convert a dynamic chart into a static artwork object:
1. Select the chart you want to convert to artwork.
2. Choose Chart —> Convert to Artwork.

Graphics converts the chart to a group object composed of shapes which you can manipulate individually. As an artwork object, the chart can no longer be updated.

Renaming Charts

You can change the name of a chart either in the Object Navigator, or in the Chart property sheet.

Note: If you rename a chart that is referenced in a PL/SQL program unit, you must change those references to reflect the new chart name.

You can change the name of your chart directly in the Object Navigator.

To rename a chart in the Object Navigator:
1. Select the chart object you wish to rename.
2. Move the mouse pointer over the chart name. The cursor changes from a pointer to a text editor.
Renaming Charts in the Chart Property Sheet

To rename your chart in the Chart property page:

1. Open the property sheet for the chart.
2. Edit the chart name in the name field.
3. Choose the OK button to apply the new name and close the chart property sheet.

Deleting Charts

If you wish to completely remove a chart you have created, as opposed to cutting it to the system clipboard, there are two methods of doing so:

- use the Object Navigator
- use the menu

Note: If you delete a chart that you have referenced in any PL/SQL program units, you must change those references to reflect the deletion.

Note also that deleting a chart does not delete the query used to create that chart, nor does it delete any chart templates used for that chart. You can delete these separately, or use them again.

Deleting Charts in the Object Navigator

To delete a chart from a display using the Object Navigator:

1. Select the chart object in the Navigator.
2. Choose the Delete button, or choose Navigator—>Delete from the menu.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

Deleting Charts Using the Menu

To delete a chart from a display using the menu:

1. In the Layout Editor, select the chart you wish to delete.
2. Choose Edit—>Clear, or choose the Delete key.
Renaming Charts in the Chart Property Sheet

Deleting Charts in the Object Navigator

Deleting Charts Using the Menu

8 – 29 Using Charts

3. Click on the name once to position the cursor and enter edit mode. Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To rename your chart in the Chart property page:
1. Open the property sheet for the chart.
2. Edit the chart name in the name field.
3. Choose the OK button to apply the new name and close the chart property sheet.

Deleting Charts

If you wish to completely remove a chart you have created, as opposed to cutting it to the system clipboard, there are two methods of doing so:
• use the Object Navigator
• use the menu

Note: If you delete a chart that you have referenced in any PL/SQL program units, you must change those references to reflect the deletion. Note also that deleting a chart does not delete the query used to create that chart, nor does it delete any chart templates used for that chart. You can delete these separately, or use them again.

To delete a chart from a display using the Object Navigator:
1. Select the chart object in the Navigator.
2. Choose the Delete button, or choose Navigator —> Delete from the menu.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

To delete a chart from a display using the menu:
1. In the Layout Editor, select the chart you wish to delete.
2. Choose Edit —> Clear, or choose the Delete key.
CHAPTER 9

Using Chart Templates

This chapter discusses the use of chart templates in Graphics.
The following topics are covered in this chapter:

- overview of chart templates – 9 – 1
- the chart template editor – 9 – 3
- working with chart templates – 9 – 5
- working with field templates – 9 – 7
- selecting chart templates – 9 – 9
- administering exported chart templates – 9 – 10

Overview of Chart Templates

Using a single chart template, you can create multiple charts with the same set of basic characteristics. For example, instead of specifying the properties for each chart individually, you can create a single chart template and associate it with multiple queries.

Types of Templates

There are two basic types of chart templates:

- default
- customized
This chapter discusses the use of chart templates in Graphics. The following topics are covered in this chapter:

- overview of chart templates
- the chart template editor
- working with chart templates
- working with field templates
- selecting chart templates
- administering exported chart templates

Overview of Chart Templates

Using a single chart template, you can create multiple charts with the same set of basic characteristics. For example, instead of specifying the properties for each chart individually, you can create a single chart template and associate it with multiple queries.

There are two basic types of chart templates:

- default
- customized
<table>
<thead>
<tr>
<th>Default Chart Templates</th>
<th>Graphics provides a set of 56 default chart templates divided into 10 categories, including bar, line, and pie. Each time you select a chart type and subtype during the chart creation process, you are selecting one of the default chart templates styles.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Because you cannot modify the default chart templates, Graphics copies the default chart template and assigns it the name $templatex$, where $x$ is a number that is incremented each time you create a new chart template for the current display.</td>
</tr>
<tr>
<td>Customized Chart Templates</td>
<td>While you cannot modify the default chart templates themselves, you can customize the chart template copies that Graphics creates during the chart creation process.</td>
</tr>
<tr>
<td></td>
<td>Once you have customized a chart template, you can assign it to multiple charts by selecting it from the Templates drop-down list in the Chart property sheet during chart creation. To customize chart templates, use the Chart Template editor discussed in the next section.</td>
</tr>
<tr>
<td>Components of Templates</td>
<td>Chart template components are divided into two main categories:</td>
</tr>
<tr>
<td></td>
<td>Chart frame Includes components that make up the basic structure of the chart, such as its axis label, fonts, and legend position. A chart template can have only one chart frame.</td>
</tr>
<tr>
<td></td>
<td>When you first select a chart template, the chart frame is what appears in the Chart Template editor by default.</td>
</tr>
<tr>
<td></td>
<td>Field templates Includes components that determine how value (dependent) data are represented as chart fields, such as bars or lines. A chart template has at least one default field template, but you can create additional ones to associate with different chart values (query columns).</td>
</tr>
<tr>
<td></td>
<td>You can view a field template when the chart frame is showing in the Chart Template editor by choosing the Field Template radio button at the top of the editor.</td>
</tr>
<tr>
<td>Properties of Templates</td>
<td>Chart template properties define the appearance and behavior of chart frame and field template components. The properties that apply and how you define them vary depending on the type of chart template. In general, you define chart template properties in the Chart Template editor by selecting a component, opening its property sheet or Format menu, and choosing options for it.</td>
</tr>
</tbody>
</table>
Graphics provides a set of 56 default chart templates divided into 10 categories, including bar, line, and pie. Each time you select a chart type and subtype during the chart creation process, you are selecting one of the default chart templates styles.

Because you cannot modify the default chart templates, Graphics copies the default chart template and assigns it the name $templatex$, where $x$ is a number that is incremented each time you create a new chart template for the current display.

While you cannot modify the default chart templates themselves, you can customize the chart template copies that Graphics creates during the chart creation process. Once you have customized a chart template, you can assign it to multiple charts by selecting it from the Templates drop-down list in the Chart property sheet during chart creation. To customize chart templates, use the Chart Template editor discussed in the next section.

Chart template components are divided into two main categories:

- Includes components that make up the basic structure of the chart, such as its axis label, fonts, and legend position. A chart template can have only one chart frame. When you first select a chart template, the chart frame is what appears in the Chart Template editor by default.

- Includes components that determine how value (dependent) data are represented as chart fields, such as bars or lines. A chart template has at least one default field template, but you can create additional ones to associate with different chart values (query columns). You can view a field template when the chart frame is showing in the Chart Template editor by choosing the Field Template radio button at the top of the editor.

Chart template properties define the appearance and behavior of chart frame and field template components. The properties that apply and how you define them vary depending on the type of chart template. In general, you define chart template properties in the Chart Template editor by selecting a component, opening its property sheet or Format menu, and choosing options for it.
The Chart Template Editor

Each display contains only one Chart Template editor. Although you can define many chart templates in a single display, you can view or modify only one at a time in the Chart Template editor.

The figure below shows a sample Chart Template editor.

![Chart Template Editor](image)

Note that the Chart Template editor, like the Layout editor, allows you to set multiple views by dragging the split bars in the lower right corner. This allows you to show a chart template in one view and its field templates in another.

The Chart Template editor has the following components:

- frame editor
- field template editor
- tool palette

Frame Editor

The frame editor is where you select and modify elements of the chart’s frame. The frame determines the basic structure of the chart, and has little to do with the data that will be plotted. The appearance of the frame in the editor represents how the chart will appear when it is created. You can change the font, color, and other graphical properties of frame elements, and see the results immediately. You can also move the legend (if any) by dragging the legend’s box (not the text) to a new location.
The Chart Template Editor

Each display contains only one Chart Template editor. Although you can define many chart templates in a single display, you can view or modify only one at a time in the Chart Template editor.

The figure below shows a sample Chart Template editor. Note that the Chart Template editor, like the Layout editor, allows you to set multiple views by dragging the split bars in the lower right corner. This allows you to show a chart template in one view and its field templates in another.

The Chart Template editor has the following components:

- frame editor
- field template editor
- tool palette

The frame editor is where you select and modify elements of the chart's frame. The frame determines the basic structure of the chart, and has little to do with the data that will be plotted. The appearance of the frame in the editor represents how the chart will appear when it is created. You can change the font, color, and other graphical properties of frame elements, and see the results immediately. You can also move the legend (if any) by dragging the legend's box (not the text) to a new location.
The figure below shows a sample frame in the Chart Template editor.

Field Template Editor  The field template editor is where you select and modify elements of the chart’s field templates. A field template determines how a set of data will appear in the chart. When you create a chart, you must associate columns from the query with specific field templates. The graphical properties of the field templates determine how the data will be plotted in the chart.

The figure below shows a sample field template in the Chart Template editor.
The figure below shows a sample frame in the Chart Template editor. The field template editor is where you select and modify elements of the chart's field templates. A field template determines how a set of data will appear in the chart. When you create a chart, you must associate columns from the query with specific field templates. The graphical properties of the field templates determine how the data will be plotted in the chart.
Tool Palette

The Tool palette in the Chart Template editor contains a Select tool, as well as the Line/Fill/Text Display indicator, Line Color, Fill Color, and Pattern palettes, Text Color palette, and Symbol palettes.

For more information about applying patterns and colors to chart elements in the layout, see “Setting Patterns and Color for Objects” on page 4 – 25.

Working with Chart Templates

Creating Chart Templates

You can create a customized chart template prior to creating any charts. Once you have created a customized chart template, you can assign it to charts as you create them by selecting the customized chart template from the Templates drop-down list in the Chart property sheet.

In addition, you can export a chart template to the file system or the database so you can use it in other displays.

To create a new chart template to assign to charts at a later time:

1. Choose Tools—>Templates to show the Chart Template editor.

   Alternatively, you can select the Templates node in the Object Navigator and choose the Create button or Navigator—>Create.

   **Suggestion:** If no chart templates currently exist, double-click on the Templates node in the Object Navigator to show a new default column chart template in the Chart Template editor.

2. Accept the default chart type (column) or choose an alternate chart type and subtype from the Type menu.

3. Accept the default chart template name (templatex) or edit the name by choosing Template—>Rename to show the Rename Template dialog.

4. Customize the chart’s frame and axis properties as desired.

5. Choose File—>Close to close the Chart Template editor when you are done.

Customizing the Chart Template

Before you edit the chart frame of the current chart template, make sure it’s showing in the Chart Template editor so that you can verify the changes you make:

- In the Chart Template editor, select the Frame radio button to show the chart frame.
The Tool palette in the Chart Template editor contains a Select tool, as well as the Line/Fill/Text Display indicator, Line Color, Fill Color, and Pattern palettes, Text Color palette, and Symbol palettes.

For more information about applying patterns and colors to chart elements in the layout, see "Setting Patterns and Color for Objects" on page 4-25.

Working with Chart Templates
You can create a customized chart template prior to creating any charts. Once you have created a customized chart template, you can assign it to charts as you create them by selecting the customized chart template from the Templates drop-down list in the Chart property sheet.

In addition, you can export a chart template to the file system or the database so you can use it in other displays.

To create a new chart template to assign to charts at a later time:

1. Choose Tools—>Templates to show the Chart Template editor.

   Alternatively, you can select the Templates node in the Object Navigator and choose the Create button or Navigator—>Create.

   Suggestion: If no chart templates currently exist, double-click on the Templates node in the Object Navigator to show a new default column chart template in the Chart Template editor.

2. Accept the default chart type (column) or choose an alternate chart type and subtype from the Type menu.

3. Accept the default chart template name (template x) or edit the name by choosing Template—>Rename to show the Rename Template dialog.

4. Customize the chart's frame and axis properties as desired.

5. Choose File—>Close to close the Chart Template editor when you are done.

Before you edit the chart frame of the current chart template, make sure it’s showing in the Chart Template editor so that you can verify the changes you make:

In the Chart Template editor, select the Frame radio button to show the chart frame.
You customize a chart template’s frame and axis properties the same way you customize a chart’s frame and axis properties.

**Note:** Where the sections below reference the Chart menu, use the Template menu within the Chart Template editor.

See the following sections for more information on customizing chart templates:

- “Specifying Axis Frame Properties” – 8 – 15
- “Specifying Axis Properties” – 8 – 17
- “Moving the Legend” – 8 – 19
- “Adding, Editing and Deleting Reference Lines” – 8 – 19
- “Specifying Number and Date Formats” – 8 – 20
- “Setting Properties for Text Objects” – 4 – 11
- “Setting Patterns and Colors for Objects” – 4 – 25

**Note:** Use the color palettes in the Chart Template editor instead of the color palettes in the Layout editor.

- “Editing Line and Edge Properties” – 4 – 29

In addition to customizing the template’s frame and axis properties, you can customize one or more field templates for use with the chart template. See the following section “Working with Field Templates” for more information.

**Renaming Chart Templates**

You can change the name of your chart template directly in the Object Navigator, or from the Chart Template editor.

**Note:** If you rename a chart template that you have referenced in any PL/SQL program units, you must change those references to reflect the new name.

To rename a chart template in the Object Navigator:

1. Select the chart template object you wish to rename.
2. Move the mouse pointer over the chart template name.
   
   The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.

**Suggestion:** If you wish to replace the entire name, click on the name to highlight it, then type the new name.
You customize a chart template's frame and axis properties the same way you customize a chart's frame and axis properties. Note: Where the sections below reference the Chart menu, use the Template menu within the Chart Template editor. See the following sections for more information on customizing chart templates:

- "Specifying Axis Frame Properties" – 8 – 15
- "Specifying Axis Properties" – 8 – 17
- "Moving the Legend" – 8 – 19
- "Adding, Editing and Deleting Reference Lines" – 8 – 19
- "Specifying Number and Date Formats" – 8 – 20
- "Setting Properties for Text Objects" – 4 – 11
- "Setting Patterns and Colors for Objects" – 4 – 25

Note: Use the color palettes in the Chart Template editor instead of the color palettes in the Layout editor.

- "Editing Line and Edge Properties" – 4 – 29

In addition to customizing the template's frame and axis properties, you can customize one or more field templates for use with the chart template. See the following section "Working with Field Templates" for more information.

You can change the name of your chart template directly in the Object Navigator, or from the Chart Template editor. Note: If you rename a chart template that you have referenced in any PL/SQL program units, you must change those references to reflect the new name.

To rename a chart template in the Object Navigator:

1. Select the chart template object you wish to rename.
2. Move the mouse pointer over the chart template name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.

Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.
Deleting Chart Templates

To rename a chart template in the Chart Template editor:
1. Select the chart template you wish to rename from the Template drop-down list.
   The chart template appears in the editor layout.
2. Choose Template—>Rename to show the Rename Template dialog.
3. Type the new template name in the Template Name field.
4. Choose the OK button to apply the new name and close the dialog.

You can delete a chart template directly from the Object Navigator, or from the Chart Template editor.

Note: If you delete a chart template that you’ve referenced in any PL/SQL program units, you must change those references to reflect the deletion.

To delete a chart template from the Object Navigator:
1. Select the chart template object in the Navigator.
2. Choose the Delete button, or choose Navigator—>Delete from the menu.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

To delete a chart template from the Chart Template editor:
1. Select the chart template you wish to delete from the Template drop-down list.
2. Choose Template—>Delete.
   An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

Working with Field Templates

You can create multiple field templates for a chart template if you want to assign different properties to value column fields. Each field template can be assigned to a different value column field, assigned to multiple value column fields, or remain unused until you need it.
To rename a chart template in the Chart Template editor:
1. Select the chart template you wish to rename from the Template drop-down list. The chart template appears in the editor layout.
2. Choose Template —> Rename to show the Rename Template dialog.
3. Type the new template name in the Template Name field.
4. Choose the OK button to apply the new name and close the dialog.

You can delete a chart template directly from the Object Navigator, or from the Chart Template editor.

Note: If you delete a chart template that you've referenced in any PL/SQL program units, you must change those references to reflect the deletion.

To delete a chart template from the Object Navigator:
1. Select the chart template object in the Navigator.
2. Choose the Delete button, or choose Navigator —> Delete from the menu. An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

To delete a chart template from the Chart Template editor:
1. Select the chart template you wish to delete from the Template drop-down list.
2. Choose Template —> Delete. An alert appears, prompting you to confirm the deletion.
3. Choose the Yes button to confirm.

Working with Field Templates
You can create multiple field templates for a chart template if you want to assign different properties to value column fields. Each field template can be assigned to a different value column field, assigned to multiple value column fields, or remain unused until you need it.
Adding Field Templates

To add a field template to a chart template:

1. Choose Field Templates from the pop-up menu, or choose Templates—>Field Templates to show the Field Template property sheet.

2. Choose the New button to create a new field template, initially named $tempx$. You can rename the new field template or keep the default.

3. Choose the Axis Field tab to set properties for axis chart field templates.

4. Choose the OK button to apply the new field template and close the Field Templates dialog.

Note: To see a field template appear in your chart, you must associate it with a value column. For more information, see “Modifying Category and Value Columns” on page 8 – 14.

Editing Field Templates

Before you edit a field template for the current chart template, make sure it’s showing in the Chart Template editor so that you can verify the changes you make:

1. From the Chart Template editor, select the Field template radio button to show the current field template.

2. Select the field template you wish to edit, from the drop-down list.

Specifying Axis Field Properties

Field properties include options such as plot type, line style, and label rotation, depending on the chart type. To specify field properties for the field template:

1. Double-click on the field template, or choose Template—>Field Templates to show the Field Template property sheet.

2. Choose the Axis Field tab to show the Axis Field property page.

3. Specify the desired options.

4. Choose the OK button to apply the changes and close the property sheet.

Applying Colors and Patterns

The procedures for applying colors and patterns to field templates are the same as those for objects, however, you use the color palettes in the Chart Template editor instead of the color palettes in the Layout editor. The color palettes behave the same way.

For more information about applying patterns and colors to chart elements in the layout, see “Setting Patterns and Color for Objects” on page 4 – 25.
To add a field template to a chart template:

1. Choose Field Templates from the pop-up menu, or choose Templates —> Field Templates to show the Field Template property sheet.

2. Choose the New button to create a new field template, initially named ftempx. You can rename the new field template or keep the default.

3. Choose the Axis Field tab to set properties for axis chart field templates.

4. Choose the OK button to apply the new field template and close the Field Templates dialog.

Note: To see a field template appear in your chart, you must associate it with a value column. For more information, see "Modifying Category and Value Columns" on page 8 – 14.

Before you edit a field template for the current chart template, make sure it's showing in the Chart Template editor so that you can verify the changes you make:

1. From the Chart Template editor, select the Field template radio button to show the current field template.

2. Select the field template you wish to edit, from the drop-down list.

Field properties include options such as plot type, line style, and label rotation, depending on the chart type. To specify field properties for the field template:

1. Double-click on the field template, or choose Template —> Field Templates to show the Field Template property sheet.

2. Choose the Axis Field tab to show the Axis Field property page.

3. Specify the desired options.

4. Choose the OK button to apply the changes and close the property sheet.

The procedures for applying colors and patterns to field templates are the same as those for objects, however, you use the color palettes in the Chart Template editor instead of the color palettes in the Layout editor. The color palettes behave the same way.

For more information about applying patterns and colors to chart elements in the layout, see "Setting Patterns and Color for Objects" on page 4 – 25.
Assigning Field Templates

Once you have created a field template and customized it, you can assign it to a value column in the chart. To assign a field template to a value column field:

1. Select the Chart in the Layout editor or Object Navigator.
2. Choose Properties from the pop-up menu, or choose Tools—>Properties to show the Chart’s property sheet.
3. Choose the Value tab to show the chart’s Value property page.
4. From the Chart Values list, select the value column you wish to assign a field template to.
5. Select the field template from the Field Templates drop-down list.
6. Choose the OK button to apply your changes and close the Chart property sheet.

Deleting Field Templates

To delete a field template from a chart template:

1. Choose Template—>Field Templates to show the Field Template property sheet.
2. Select a field template from the from the Field Template drop-down list.
3. Select the Delete button to remove the template from the list.
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.
5. Choose the OK button to accept the changes and close the Field Template property sheet.

Note: If you’ve created a chart that contains a value column associated with the deleted field template, you must associate the value column with another field template. For more information, see “Modifying Category and Value Columns” on page 8 – 14.

Selecting a Chart Template

If you select Tools—>Templates from the menu without first selecting a chart, the Chart Template editor appears with the current chart template. The current chart template is the one that was last selected in the Chart Template editor. You can select a different default or customized chart template to be the “current” chart template.
Once you have created a field template and customized it, you can assign it to a value column in the chart. To assign a field template to a value column field:

1. Select the Chart in the Layout editor or Object Navigator.
2. Choose Properties from the pop-up menu, or choose Tools —> Properties to show the Chart's property sheet.
3. Choose the Value tab to show the chart's Value property page.
4. From the Chart Values list, select the value column you wish to assign a field template to.
5. Select the field template from the Field Templates drop-down list.
6. Choose the OK button to apply your changes and close the Chart property sheet.

To delete a field template from a chart template:

1. Choose Template —> Field Templates to show the Field Template property sheet.
2. Select a field template from the from the Field Template drop-down list.
3. Select the Delete button to remove the template from the list. An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.
5. Choose the OK button to accept the changes and close the Field Template property sheet.

Note: If you've created a chart that contains a value column associated with the deleted field template, you must associate the value column with another field template. For more information, see "Modifying Category and Value Columns" on page 8 – 14.

Selecting a Chart Template

If you select Tools —> Templates from the menu without first selecting a chart, the Chart Template editor appears with the current chart template. The current chart template is the one that was last selected in the Chart Template editor. You can select a different default or customized chart template to be the "current" chart template.
### Selecting a Default Chart Template

To select a different chart template in the Chart Template editor:

1. Choose **Tools**—>**Templates** to show the Chart Template editor.
2. Select the type of chart you want (such as Pie), from the Type menu.
3. Choose one of the chart templates shown on the submenu for that type of chart.

   The chart frame for the default chart template you selected appears in the Chart Template editor. This remains the current chart template until you create or edit another one.
4. Close the Chart Template editor by choosing **File**—>**Close**.

   This chart type is now the current chart template. When you select **Tools**—>**Templates** from the menu, this chart template will be shown in the Chart Template editor.

### Selecting a Customized Chart Template

To change the default chart type to a chart template that you have customized:

1. Choose **Tools**—>**Templates** to show the Chart Template editor.
2. In the Chart Template editor, select the Template pop-list to show a list of customized templates.
3. Choose one of the chart templates listed to show it in the Chart Template editor.

   This remains the current chart template until you create or edit another one.
4. Close the Chart Template editor by choosing **File**—>**Close**.

   This chart type is now the current chart template. When you select **Tools**—>**Templates** from the menu, this chart template will be shown in the Chart Template editor.

### AdministeringExported Chart Templates

Once you have created and customized a chart template, you can export it to the file system or to the database for use in other displays.

### Exporting Chart Templates

You can export chart templates from the Object Navigator or from the Chart Template editor. To do so:

1. Select the template object in the Navigator, or select the chart template you wish to export from the Templates drop-down list in the Chart Template editor.
Selecting a Default Chart Template

Selecting a Customized Chart Template

Exporting Chart Templates

To select a different chart template in the Chart Template editor:

1. Choose **Tools** —> **Templates** to show the Chart Template editor.
2. Select the type of chart you want (such as Pie), from the Type menu.
3. Choose one of the chart templates shown on the submenu for that type of chart.
   
   The chart frame for the default chart template you selected appears in the Chart Template editor. This remains the current chart template until you create or edit another one.

4. Close the Chart Template editor by choosing **File** —> **Close**.
   
   This chart type is now the current chart template. When you select **Tools** —> **Templates** from the menu, this chart template will be shown in the Chart Template editor.

To change the default chart type to a chart template that you have customized:

1. Choose **Tools** —> **Templates** to show the Chart Template editor.
2. In the Chart Template editor, select the Template pop-list to show a list of customized templates.
3. Choose one of the chart templates listed to show it in the Chart Template editor.
   
   This remains the current chart template until you create or edit another one.

4. Close the Chart Template editor by choosing **File** —> **Close**.
   
   This chart type is now the current chart template. When you select **Tools** —> **Templates** from the menu, this chart template will be shown in the Chart Template editor.

Administering Exported Chart Templates

Once you have created and customized a chart template, you can export it to the file system or to the database for use in other displays.

You can export chart templates from the Object Navigator or from the Chart Template editor. To do so:

1. Select the template object in the Navigator, or select the chart template you wish to export from the Templates drop-down list in the Chart Template editor.
2. Choose **Edit** → **Export** → **Template** to show the Export dialog.
3. Select the export options such a location and file/module name.
4. Choose the OK button to export the template and close the export dialog.

**Importing Chart Templates**

You can import a previously exported chart template from the Object Navigator, or the Chart Template editor.

To import a chart template:

1. Select the Templates node in the Object Navigator, or choose **Tools** → **Templates** to show the Chart Template editor.
2. Choose **Edit** → **Import** → **Template** to show the Import dialog.
3. Select the file/module name and location.
4. Choose the OK button to import the specified chart template into the current display.

**Renaming, Deleting, and Granting Access to Exported Chart Templates**

The procedures for renaming, deleting, and granting access to exported chart templates are the same as those for displays. For more information about administering displays, see the following sections of the “Managing Displays” chapter:

- “Granting and Revoking Access to Displays” – 6 – 19
- “Renaming Displays” – 6 – 20
- “Deleting Displays” – 6 – 20
Importing Chart Templates

Renaming, Deleting, and Granting Access to Exported Chart Templates

Using Chart Templates

2. Choose Edit —> Export —> Template to show the Export dialog.

3. Select the export options such as location and file/module name.

4. Choose the OK button to export the template and close the export dialog.

You can import a previously exported chart template from the Object Navigator, or the Chart Template editor.

To import a chart template:

1. Select the Templates node in the Object Navigator, or choose Tools —> Templates to show the Chart Template editor.

2. Choose Edit —> Import —> Template to show the Import dialog.

3. Select the file/module name and location.

4. Choose the OK button to import the specified chart template into the current display.

The procedures for renaming, deleting, and granting access to exported chart templates are the same as those for displays. For more information about administering displays, see the following sections of the "Managing Displays" chapter:

• "Granting and Revoking Access to Displays" – 6 – 19
• "Renaming Displays" – 6 – 20
• "Deleting Displays" – 6 – 20
Using Parameters

This chapter describes how to use parameters in Graphics. Parameters are global variables you can define within a display. You can reference parameters in either SQL SELECT statements or PL/SQL program units.

The following topics are covered in this chapter:

- working with parameters – 10 – 1
- creating parameters – 10 – 2
- editing parameters – 10 – 5
- referencing parameters – 10 – 5
- assigning values to parameters – 10 – 7
- renaming parameters – 10 – 9
- deleting parameters – 10 – 10

Working with Parameters

Parameters are most commonly used in drill-down relationships, where they represent the value of a chart element or the name of a graphic object in your display. The parameter is then used in a query or PL/SQL program unit to drive a specified action.
CHAPTER 10

10 – 1 Using Parameters

This chapter describes how to use parameters in Graphics. Parameters are global variables you can define within a display. You can reference parameters in either SQL SELECT statements or PL/SQL program units.

The following topics are covered in this chapter:

• working with parameters – 10 – 1
• creating parameters – 10 – 2
• editing parameters – 10 – 5
• referencing parameters – 10 – 5
• assigning values to parameters – 10 – 7
• renaming parameters – 10 – 9
• deleting parameters – 10 – 10

Working with Parameters

Parameters are most commonly used in drill-down relationships, where they represent the value of a chart element or the name of a graphic object in your display. The parameter is then used in a query or PL/SQL program unit to drive a specified action.
Parameters can also be used in multiple PL/SQL program units as global variables.

Possible uses for parameters are to:

- represent the value of a chart element in a drill-down relationship
- represent the name of a graphic object in a drill-down relationship or button procedure
- store a value in a PL/SQL program unit
- store a string value such as a where clause for use in a query

**Parameter Components**

You create parameters using the Parameters dialog box. Define the following settings when creating a parameter:

- **Name**: Is the name that you will use to refer to the parameter in your SQL or PL/SQL statements.
- **Type**: Is the data type of the value the parameter represents. The valid entries in the pop-list are Number, Char, and Date.
- **Initial Value**: (Optional) is the value assigned to the parameter when you first run the display. The value entered here can be superseded by a value that is assigned manually or programmatically.

To create a global variable of a different datatype than NUMBER, CHAR, or DATE (e.g., one of the Graphics PL/SQL built-in datatypes), create a package to define those variables. For more information on PL/SQL packages, see the PL/SQL User’s Guide and Reference.

**Creating Parameters**

You can create parameters before you use them, or create them within other tasks such as establishing drill-down relationships. You can create parameters in the following ways:

- create standalone parameters
  - from the menu
  - from the Object Navigator
- create parameters on-the-fly
  - within a query or program unit
  - within a drill-down procedure
Parameters can also be used in multiple PL/SQL program units as global variables. Possible uses for parameters are to:

- represent the value of a chart element in a drill-down relationship
- represent the name of a graphic object in a drill-down relationship or button procedure
- store a value in a PL/SQL program unit
- store a string value such as a where clause for use in a query

You create parameters using the Parameters dialog box. Define the following settings when creating a parameter:

- Name: the name that you will use to refer to the parameter in your SQL or PL/SQL statements.
- Type: the data type of the value the parameter represents. The valid entries in the pop-list are Number, Char, and Date.
- Initial Value: (Optional) the value assigned to the parameter when you first run the display. The value entered here can be superseded by a value that is assigned manually or programmatically.

To create a global variable of a different datatype than NUMBER, CHAR, or DATE (e.g., one of the Graphics PL/SQL built-in datatypes), create a package to define those variables. For more information on PL/SQL packages, see the PL/SQL User's Guide and Reference.

Creating Parameters

You can create parameters before you use them, or create them within other tasks such as establishing drill-down relationships. You can create parameters in the following ways:

- create standalone parameters
  - from the menu
  - from the Object Navigator
- create parameters on-the-fly
  - within a query or program unit
  - within a drill-down procedure

**Name**

**Type**

**Initial Value**
Once you have created a parameter, you can reference it within program units, SQL queries, or drill-down procedures.

Creating Parameters with the Object Navigator

To create a standalone parameter using the Object Navigator:
1. Select the Parameters node in the Object Navigator
2. Choose the Create button, or choose Navigator—>Create from the menu to show the Parameters dialog box.

   **Suggestion:** If no parameters currently exist, double-click on the Parameters node to show the Parameters dialog box.

   The new parameter is initially named $\text{param}_x$ and has a type of Char.
3. Define the new parameter’s name, type, and initial value.
4. Choose the OK button to accept the new parameter settings and close the dialog box.

Creating Parameters with the Menu

To create a standalone parameter from the menu:
1. Choose Tools—>Parameters to show the Parameters dialog box.
2. Choose the New button to create a new parameter.

   The new parameter is initially named $\text{param}_x$ and has a type of Char.
3. Define the parameter’s name, type, and initial value.
4. Choose the OK button to accept the new parameter settings and close the dialog box.

Creating Parameters on-the-fly

You can create new parameters as you program, without having to interrupt the current task. You can create parameters on-the-fly by simply referencing a new parameter name in a query or program unit, or you can create a new parameter as part of creating a drill-down relationship between two charts.

Reference a New Parameter Name

If you are creating a SQL SELECT statement or PL/SQL program unit and decide you need to use a parameter, simply reference a new and unique parameter name as you would if the parameter existed. When you attempt to execute the query, or compile the program unit, Graphics will prompt you to create the new parameter and will show the Parameter dialog.
Once you have created a parameter, you can reference it within program units, SQL queries, or drill-down procedures.

To create a standalone parameter using the Object Navigator:
1. Select the Parameters node in the Object Navigator.
2. Choose the Create button, or choose Navigator —> Create from the menu to show the Parameters dialog box.
   * Suggestion: If no parameters currently exist, double-click on the Parameters node to show the Parameters dialog box.
3. Define the new parameter’s name, type, and initial value.
4. Choose the OK button to accept the new parameter settings and close the dialog box.

To create a standalone parameter from the menu:
1. Choose Tools —> Parameters to show the Parameters dialog box.
2. Choose the New button to create a new parameter.
3. Define the parameter’s name, type, and initial value.
4. Choose the OK button to accept the new parameter settings and close the dialog box.

You can create new parameters as you program, without having to interrupt the current task. You can create parameters on-the-fly by simply referencing a new parameter name in a query or program unit, or you can create a new parameter as part of creating a drill-down relationship between two charts.

If you are creating a SQL SELECT statement or PL/SQL program unit and decide you need to use a parameter, simply reference a new and unique parameter name as you would if the parameter existed. When you attempt to execute the query, or compile the program unit, Graphics will prompt you to create the new parameter and will show the Parameter dialog.
To create a new parameter by referencing it:

1. Within a SQL query or PL/SQL program unit, enter a reference to a unique parameter name. For example:
   
   ```sql
   :object := OG_GET_OBJECT (circle1);
   ```

2. Choose the OK button to execute your query, or choose the Compile button to compile your program unit.

3. An alert appears prompting you that the referenced parameter does not exist and asking if you would like to create it.

4. Choose the Yes button to confirm and show the Parameters dialog.

5. Define the parameter’s type, and initial value.

6. Choose the OK button to accept the new parameter settings and close the dialog box.

7. Choose the OK button to re-execute your query, or choose the Compile button to re-compile your program unit.

You can create a parameter to drive your drill-down relationships while you define the drill-down relation. It is not necessary to create a parameter prior to creating a drill-down relationship.

1. Double-click on an object to which you wish to assign parameter values (chart element, etc.) to show the Object property sheet.

   Alternatively, you can select the object and choose the Properties command from the pop-up menu, or choose **Tools**—>**Properties** to show the Object property sheet.

2. Choose the Drill-down tab to show the Drill-down property page.

3. Choose the New button to the right of the Set Parameter field to show the parameters dialog box.

4. Define the parameter’s type, and initial value.

5. Choose OK to accept the new parameter settings and close the dialog box.

6. Set the To Value Of and Execute Query fields in the Drill-down property page.

7. Choose the OK button to accept the drill-down relationship.

For more information about defining drill-down relationships, see “Using Drill-down Charts” on page 8 – 26.
To create a new parameter by referencing it:

1. Within a SQL query or PL/SQL program unit, enter a reference to a unique parameter name. For example:
   
   ```
   :object := OG_GET_OBJECT (circle1);
   ```

2. Choose the OK button to execute your query, or choose the Compile button to compile your program unit.

3. An alert appears prompting you that the referenced parameter does not exist and asking if you would like to create it.

4. Choose the Yes button to confirm and show the Parameters dialog.

5. Define the parameter's type, and initial value.

6. Choose the OK button to accept the new parameter settings and close the dialog box.

7. Choose the OK button to re-execute your query, or choose the Compile button to re-compile your program unit.

You can create a parameter to drive your drill-down relationships while you define the drill-down relation. It is not necessary to create a parameter prior to creating a drill-down relationship.

1. Double-click on an object to which you wish to assign parameter values (chart element, etc.) to show the Object property sheet. Alternatively, you can select the object and choose the Properties command from the pop-up menu, or choose Tools —> Properties to show the Object property sheet.

2. Choose the Drill-down tab to show the Drill-down property page.

3. Choose the New button to the right of the Set Parameter field to show the parameters dialog box.

4. Define the parameter's type, and initial value.

5. Choose OK to accept the new parameter settings and close the dialog box.

6. Set the To Value Of and Execute Query fields in the Drill-down property page.

7. Choose the OK button to accept the drill-down relationship.

For more information about defining drill-down relationships, see "Using Drill-down Charts" on page 8–26.
Editing Parameters

You can modify a parameter either through the Object Navigator or the menu.

Note: If you only want to change the name of your parameter, simply click on the parameter name in the Object Navigator and edit it directly. If you change a parameter’s name, you must also change any references to that parameter in your queries or program units.

Editing Parameters with the Object Navigator

To modify a parameter with the Object Navigator:
1. Double-click on the parameter object to show the Parameters dialog.
   Alternatively, you can select the parameter object and choose the Properties command from the pop-up menu, or choose Tools—>Properties.
2. Make the desired setting changes.
3. Choose the OK button to accept the changes and close the dialog box.

Editing Parameters Using the Menu

To modify a parameter with the menu:
1. Choose Tools—>Parameters to show the Parameters dialog box.
2. Choose the name of the parameter you wish to modify from the Parameters drop-down list.
3. Make the desired setting changes.
4. Choose the OK button to accept the changes and close the dialog box.

Referencing Parameters

To reference parameters in Graphics, use either bind or lexical references.

Note that although parameter names are listed in all capital letters in the Parameters dialog box, they are not case-sensitive when used in your SQL or PL/SQL statements.

If you reference a parameter that does not exist, an alert appears asking if you wish to create it at that time. If you answer yes, the Parameters dialog is presented.
You can modify a parameter either through the Object Navigator or the menu.

**Note:** If you only want to change the name of your parameter, simply click on the parameter name in the Object Navigator and edit it directly. If you change a parameter's name, you must also change any references to that parameter in your queries or program units.

**To modify a parameter with the Object Navigator:**
1. Double-click on the parameter object to show the Parameters dialog.
   Alternatively, you can select the parameter object and choose the Properties command from the pop-up menu, or choose Tools —> Properties.
2. Make the desired setting changes.
3. Choose the OK button to accept the changes and close the dialog box.

**To modify a parameter with the menu:**
1. Choose Tools —> Parameters to show the Parameters dialog box.
2. Choose the name of the parameter you wish to modify from the Parameters drop-down list.
3. Make the desired setting changes.
4. Choose the OK button to accept the changes and close the dialog box.

**Referencing Parameters**
To reference parameters in Graphics, use either bind or lexical references. Note that although parameter names are listed in all capital letters in the Parameters dialog box, they are not case-sensitive when used in your SQL or PL/SQL statements.

If you reference a parameter that does not exist, an alert appears asking if you wish to create it at that time. If you answer yes, the Parameters dialog is presented.
Using Bind References

Use a bind reference to substitute a single literal value, such as a one-word character string, number, or date in either SQL SELECT statements or PL/SQL program units.

You create a bind reference to a parameter by typing a colon(:) immediately before the parameter name. Because you can create parameters on-the-fly, it is not necessary to define the parameter prior to referencing it.

For example, in the following SQL SELECT statement, *my_param* represents a single value for the ENAME column in the EMP table:

```sql
select ename, sal
from emp
where ename = :my_param
```

Using Lexical References

Use a lexical reference to substitute a compound text string in a SQL SELECT statement.

**Note:** Only a parameter of type Char may be used as a lexical parameter.

You create a lexical reference to a parameter by typing an ampersand (&) immediately before the parameter name. Because you can create parameters on-the-fly, it is not necessary to define the parameter prior to referencing it.

For example, in the following SQL SELECT statement, *my_param* represents a WHERE clause:

```sql
select ename, sal
from emp
&my_param
```

When you define the parameter, you simply type the WHERE clause in the initial value field. Alternatively, you could assign the value programatically, then execute the query based on the new value, and update the associated chart:

```sql
PROCEDURE set_param (name IN VARCHAR2, qry IN OG_QUERY) IS
  the_chart  OG_OBJECT;
BEGIN
  :my_param:='WHERE ENAME = '||name;
  OG_EXECUTE_QUERY(qry);
  the_chart:=OG_GET_CHART('chart0');
  OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;
```
Use a bind reference to substitute a single literal value, such as a one-word character string, number, or date in either SQL SELECT statements or PL/SQL program units. You create a bind reference to a parameter by typing a colon (:) immediately before the parameter name. Because you can create parameters on-the-fly, it is not necessary to define the parameter prior to referencing it.

For example, in the following SQL SELECT statement, `my_param` represents a single value for the ENAME column in the EMP table:

```sql
SELECT ename, sal
FROM emp
WHERE ename = :my_param
```

Use a lexical reference to substitute a compound text string in a SQL SELECT statement. Note: Only a parameter of type Char may be used as a lexical parameter. You create a lexical reference to a parameter by typing an ampersand (&) immediately before the parameter name. Because you can create parameters on-the-fly, it is not necessary to define the parameter prior to referencing it.

For example, in the following SQL SELECT statement, `my_param` represents a WHERE clause:

```sql
SELECT ename, sal
FROM emp
&my_param
```

When you define the parameter, you simply type the WHERE clause in the initial value field. Alternatively, you could assign the value programmatically, then execute the query based on the new value, and update the associated chart:

```sql
PROCEDURE set_param (name IN VARCHAR2, qry IN OG_QUERY) IS
  the_chart  OG_OBJECT;
BEGIN
  :my_param:='WHERE ENAME = '||name;
  OG_EXECUTE_QUERY(qry);
  the_chart:=OG_GET_CHART('chart0');
  OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END;
```
Assigning Values to Parameters

You can assign values to parameters in three ways:

- initially, in the Parameters dialog box
- within PL/SQL program units
- as arguments to the Runtime or Batch executables

At runtime, the values of the parameters are substituted for their references in the SQL SELECT statements or PL/SQL program units.

Assigning Initial Values to Parameters

When you create the parameter, you can assign it an initial value by typing a value in the Initial Value field. For information about assigning initial values to parameters, see “Creating Parameters” on page 10 – 2.

Assigning Values Using PL/SQL

To assign a value to a parameter in a PL/SQL program unit, you can reference the parameter and assign it a value as you would any other variable.

For example, to assign the parameter DNO a value of 10, you could include the following statement in a procedure:

```sql
:DNO := 10;
```

You can assign parameters an absolute value, such as a number or text string, or you can assign them relative values using other variables, parameters, and operators.

For example, suppose you defined a parameter named `my_num` of type `Number`. The following is valid in a PL/SQL statement:

```sql
PROCEDURE double IS
BEGIN
  :my_num := :my_num*2;
END;
```

This would set the value of `my_num` to twice its previous value.

Assigning Values Based on User Action

You can tie a parameter value to any object in your display, for instance, a chart element, and have the user set the parameter value by selecting a display object in the Runtime display.

Drill-down Charts

A common application of this type of value assignment is a drill-down relationship between two charts. In this case, the parameter serves as the connecting link between a master chart and a detail chart.

To assign a parameter value to a chart element:

1. Select the master chart.
You can assign values to parameters in three ways:

- Initially, in the Parameters dialog box
- Within PL/SQL program units
- As arguments to the Runtime or Batch executables

At runtime, the values of the parameters are substituted for their references in the SQL SELECT statements or PL/SQL program units.

When you create the parameter, you can assign it an initial value by typing a value in the Initial Value field. For information about assigning initial values to parameters, see "Creating Parameters" on page 10–2.

To assign a value to a parameter in a PL/SQL program unit, you can reference the parameter and assign it a value as you would any other variable.

For example, to assign the parameter DNO a value of 10, you could include the following statement in a procedure:

```
: DNO := 10;
```

You can assign parameters an absolute value, such as a number or text string, or you can assign them relative values using other variables, parameters, and operators.

For example, suppose you defined a parameter named `my_num` of type `Number`. The following is valid in a PL/SQL statement:

```
PROCEDURE double IS
BEGIN
: my_num := :my_num * 2;
END;
```

This would set the value of `my_num` to twice its previous value.

You can tie a parameter value to any object in your display, for instance, a chart element, and have the user set the parameter value by selecting a display object in the Runtime display.

A common application of this type of value assignment is a drill-down relationship between two charts. In this case, the parameter serves as the connecting link between a master chart and a detail chart.

To assign a parameter value to a chart element:
1. Select the master chart.
2. Select the chart element you wish to assign the parameter value to (query column).

3. Choose the Properties command from the pop-up menu, or choose **Tools**—>**Properties** to show the object’s property sheet.

4. Choose the Drill-down tab to show the drill-down page.

5. Select a parameter from the Set Parameter pop-list, or choose the New button to create a new parameter.

6. Select a query column from the To Value of pop-list.

7. If you wish the user action to execute a query based on the parameter value assignment, select a query to execute from the Execute Query drop-down list, otherwise leave the default value of **None**.

8. Choose the OK button to apply the drill-down relationship and close the Object property sheet.

For more information about defining drill-down relationships, see “Using Drill-down Charts” on page 8–26.

**Object Button Procedures**

Similar to a drill-down chart, a button procedure is activated by the user in the Runtime display. However, rather than select a chart element to drive the action, the user selects a graphic object.

1. Double-click on the graphic object you wish to set the parameter value to.

   Alternatively, you can select the object then choose Properties from the pop-up menu, or choose **Tools**—>**Properties**.

2. Choose the Drill-down tab to show the drill-down page.

3. Select a parameter from the Set Parameter pop-list, or choose the New button to create a new parameter.

4. Type the parameter name in the Name field.

   **Note:** The Object Name value is defined as the datatype CHAR. If you have defined the parameter as being a datatype other than CHAR, you must use the TO_CHAR conversion function when you reference the parameter in a query or PL/SQL program unit. For more information on TO_CHAR, see the *PL/SQL User’s Guide and Reference*.

   You can now use the parameter in any program unit within the display.

**Assigning Values on the Command Line**

You can assign a value to a parameter on the Graphics command line. The value specified for the parameter supersedes the parameter’s initial value as set in the Parameters dialog box.
Assigning Values on the Command Line

1. Select the chart element you wish to assign the parameter value to (query column).
2. Choose the Properties command from the pop-up menu, or choose Tools —> Properties to show the object's property sheet.
3. Choose the Drill-down tab to show the drill-down page.
4. Select a parameter from the Set Parameter pop-list, or choose the New button to create a new parameter.
5. Select a query column from the To Value of pop-list.
6. If you wish the user action to execute a query based on the parameter value assignment, select a query to execute from the Execute Query drop-down list, otherwise leave the default value of None.
7. Choose the OK button to apply the drill-down relationship and close the Object property sheet.

For more information about defining drill-down relationships, see "Using Drill-down Charts" on page 8 – 26.

Similar to a drill-down chart, a button procedure is activated by the user in the Runtime display. However, rather than select a chart element to drive the action, the user selects a graphic object.

1. Double-click on the graphic object you wish to set the parameter value to.
2. Alternatively, you can select the object then choose Properties from the pop-up menu, or choose Tools —> Properties.
3. Choose the Drill-down tab to show the drill-down page.
4. Select a parameter from the Set Parameter pop-list, or choose the New button to create a new parameter.
5. Type the parameter name in the Name field.

Note: The Object Name value is defined as the datatype CHAR. If you have defined the parameter as being a datatype other than CHAR, you must use the TO_CHAR conversion function when you reference the parameter in a query or PL/SQL program unit. For more information on TO_CHAR, see the PL/SQL User's Guide and Reference.

You can now use the parameter in any program unit within the display.

You can assign a value to a parameter on the Graphics command line. The value specified for the parameter supersedes the parameter's initial value as set in the Parameters dialog box.
The example below shows how you would assign values to parameters *hisparam* (NUMBER datatype) and *herparam* (DATE datatype):

```
g25des openfile=chart1 hisparam=29 herparam='02-OCT-94'
```

For more information about assigning parameter values on the command line, see the following sections of the *Graphics Reference Manual*:

- “G25DES”
- “G25RUN”

Parameters are global, whereas variables can be used only within the program unit where they are defined. If you wish to use the variable outside the program unit, use a parameter instead.

**Renaming Parameters**

You can change the name of a parameter either in the Object Navigator, or in the Parameters dialog.

**Note:** If you change the name of a parameter you have referenced in any SQL SELECT statements, PL/SQL program units, or drill-down relationships, you must change those references to reflect the new parameter name.

### Renaming Parameters in the Object Navigator

To change the name of a parameter directly in the Object Navigator:

1. Select the parameter object you wish to rename.
2. Move the mouse pointer over the parameter name.
   
   The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.
   
   **Suggestion:** If you wish to replace the entire name, click on the name to highlight it, then type the new name.

### Renaming Parameters in the Parameter Dialog

To change the name of a parameter in the Parameters dialog:

1. Open the parameters dialog box.
2. Select the parameter you wish to rename from the Name drop-down list.
3. Edit the name.
4. Choose the OK button to accept the new name and close the Parameters dialog box.
The example below shows how you would assign values to parameters `hisparam` (NUMBER datatype) and `herparam` (DATE datatype):

```
g25des openfile=chart1 hisparam=29 herparam='02–OCT–94'
```

For more information about assigning parameter values on the command line, see the following sections of the Graphics Reference Manual:

- “G25DES”
- “G25RUN”

Parameters are global, whereas variables can be used only within the program unit where they are defined. If you wish to use the variable outside the program unit, use a parameter instead.

Renaming Parameters

You can change the name of a parameter either in the Object Navigator, or in the Parameters dialog.

Note: If you change the name of a parameter you have referenced in any SQL SELECT statements, PL/SQL program units, or drill-down relationships, you must change those references to reflect the new parameter name.

To change the name of a parameter directly in the Object Navigator:

1. Select the parameter object you wish to rename.
2. Move the mouse pointer over the parameter name. The cursor changes from a pointer to a text editor.
3. Click on the name once to position the cursor and enter edit mode.
   
   Suggestion: If you wish to replace the entire name, click on the name to highlight it, then type the new name.

To change the name of a parameter in the Parameters dialog:

1. Open the parameters dialog box.
2. Select the parameter you wish to rename from the Name drop-down list.
3. Edit the name.
4. Choose the OK button to accept the new name and close the Parameters dialog box.
Deleting Parameters

You can delete a parameter either through the Object Navigator, or through the menu.

Note: If you delete a parameter you have referenced in any SQL SELECT statements, PL/SQL program units, or drill-down relationships, you must change those references to reflect the deletion.

Deleting Parameters in the Object Navigator

To delete a parameter using the Object Navigator:
1. Select the parameter object in the Navigator.
2. Choose the Delete button, or choose **Navigator —> Delete** from the menu.
   
   An alert appears, prompting you to confirm the deletion.
3. Answer **Yes** to confirm.

Deleting Parameters with the Parameters Dialog

To delete a parameter using the menu:
1. Choose **Tools —> Parameters** to show the Parameters dialog box.
2. Select the name of the parameter you wish to delete from the Parameters drop-down list.
3. Choose the Delete button.
   
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.
5. Choose the OK button to close the Parameters dialog.
Deleting Parameters in the Object Navigator

Deleting Parameters with the Parameters Dialog

You can delete a parameter either through the Object Navigator, or through the menu.

Note: If you delete a parameter you have referenced in any SQL SELECT statements, PL/SQL program units, or drill-down relationships, you must change those references to reflect the deletion.

To delete a parameter using the Object Navigator:

1. Select the parameter object in the Navigator.
2. Choose the Delete button, or choose Navigator —> Delete from the menu.
   An alert appears, prompting you to confirm the deletion.
3. Answer Yes to confirm.

To delete a parameter using the menu:

1. Choose Tools —> Parameters to show the Parameters dialog box.
2. Select the name of the parameter you wish to delete from the Parameters drop-down list.
3. Choose the Delete button.
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.
5. Choose the OK button to close the Parameters dialog.
Using Triggers

This chapter describes the different types of triggers and how they work. Triggers are PL/SQL procedures that fire at specific times during runtime. Using triggers, you can enhance your displays with such features as buttons, dynamic “on-the-fly” chart formatting, and automatic updates and commits.

The following topics are covered in this chapter:

- overview – 11 – 1
- referencing a trigger – 11 – 5
- using open display triggers – 11 – 5
- using post-execution triggers – 11 – 6
- using format triggers – 11 – 8
- using timer triggers – 11 – 9
- using button procedures – 11 – 11
- using close display triggers – 11 – 14

Overview

Graphics executes all of the trigger procedures associated with a display in a particular order. Understanding the order of execution can prove useful when developing your displays.
CHAPTER 11

Using Triggers

This chapter describes the different types of triggers and how they work. Triggers are PL/SQL procedures that fire at specific times during runtime. Using triggers, you can enhance your displays with such features as buttons, dynamic "on-the-fly" chart formatting, and automatic updates and commits.

The following topics are covered in this chapter:

• overview – 11–1
• referencing a trigger – 11–5
• using open display triggers – 11–5
• using post-execution triggers – 11–6
• using format triggers – 11–8
• using timer triggers – 11–9
• using button procedures – 11–11
• using close display triggers – 11–14

Overview

Graphics executes all of the trigger procedures associated with a display in a particular order. Understanding the order of execution can prove useful when developing your displays.
This section discusses the execution order of the events possible when you run a Graphics display:

- open display triggers
- query execution
- chart update
- timer execution
- button execution
- close display triggers

**Open Display Triggers**  
Open Display triggers fire at the time implied by their name: immediately after a display is run (opened in runtime). Open Display triggers are the first triggers to fire.

**Query Execution**  
Query execution includes the following related events, occurring in the order shown below:

- executing the query
- firing the post-execution trigger (if it exists)
- running the query filter function (if it exists)

**Executing the Query**  
Query execution occurs automatically upon opening a display if the Execute on Opening Display option is set for the query. By default, this option is activated, meaning that the query is executed immediately after the Open Display trigger is fired.

If Execute on Opening Display is not activated, query execution does not occur until the OG_EXECUTE_QUERY built-in procedure is called.

For more information about the various ways to execute queries, see “Executing Queries” on page 7–13.

**Firing the Post-Execution Trigger**  
If it exists, the Post-Execution trigger fires immediately after the query is executed.

**Executing the Query Filter Function**  
If it exists, the query filter function is executed immediately after the Post-Execution trigger is fired. The query filter does not restrict fetched data from appearing in the Query property sheet, but determines what subset of that data will be charted.

For more information about query filters, see “Filtering Chart Data” on page 8–11.
This section discusses the execution order of the events possible when you run a Graphics display:

1. Open Display triggers
2. Query execution
3. Chart update
4. Timer execution
5. Button execution
6. Close Display triggers

Open Display triggers fire at the time implied by their name: immediately after a display is run (opened in runtime). Open Display triggers are the first triggers to fire.

Query execution includes the following related events, occurring in the order shown below:

1. Executing the query
2. Firing the post-execution trigger (if it exists)
3. Running the query filter function (if it exists)

Query execution occurs automatically upon opening a display if the Execute on Opening Display option is set for the query. By default, this option is activated, meaning that the query is executed immediately after the Open Display trigger is fired.

If Execute on Opening Display is not activated, query execution does not occur until the OG_EXECUTE_QUERY built-in procedure is called. For more information about the various ways to execute queries, see "Executing Queries" on page 7-13.

If it exists, the Post-Execution trigger fires immediately after the query is executed.

If it exists, the query filter function is executed immediately after the Post-Execution trigger is fired. The query filter does not restrict fetched data from appearing in the Query property sheet, but determines what subset of that data will be charted.

For more information about query filters, see "Filtering Chart Data" on page 8-11.
Chart Updates

Chart updates include the following related events, occurring in the order shown below:

- updating the chart
- firing the format trigger (if it exists)

Updating the Chart

Chart updates occur automatically following query execution if the Update chart on query execution option is set for the chart. By default, this option is activated, meaning that the chart is updated immediately after the query execution procedures are run.

If Update chart on query execution is not activated, chart update does not occur until the OG_UPDATE_CHART built-in procedure is called.

For more information about the various ways to update charts, see “Updating Charts” on page 8 – 25.

Firing the Format Trigger

If it exists, the Format trigger fires in the final stage of the chart update, immediately before the chart is drawn.

You can use format triggers to set conditional formatting properties such as color for a chart element.

Timer Execution

Timer execution includes the following related events, occurring in the order shown below:

- executing the query
- executing the timer procedure

Executing the Query

If a query is associated with a timer (via the Execute on Timer option on the Options page of the Query property sheet), the query is executed automatically when the timer expires. By default, this option is not activated.

However, if Execute on Timer is activated, query execution could in turn spawn a chart update. For more information about the order of events related to query execution and chart updates, see the following sections:

- “Query Execution” – 11 – 2
- “Chart Updates” – 11 – 3

Executing the Timer Procedure

The timer procedure is executed simultaneously with the optional query execution (described above) when the timer expires. If no query is to be executed at timer expiration, the timer procedure is the only event to occur.
Chart updates include the following related events, occurring in the order shown below:

- updating the chart
- firing the format trigger (if it exists)

Chart updates occur automatically following query execution if the Update chart on query execution option is set for the chart. By default, this option is activated, meaning that the chart is updated immediately after the query execution procedures are run.

If Update chart on query execution is not activated, chart update does not occur until the OG_UPDATE_CHART built-in procedure is called.

For more information about the various ways to update charts, see “Updating Charts” on page 8 – 25.

If it exists, the Format trigger fires in the final stage of the chart update, immediately before the chart is drawn.

You can use format triggers to set conditional formatting properties such as color for a chart element.

Timer execution includes the following related events, occurring in the order shown below:

- executing the query
- executing the timer procedure

If a query is associated with a timer (via the Execute on Timer option on the Options page of the Query property sheet), the query is executed automatically when the timer expires. By default, this option is not activated.

However, if Execute on Timer is activated, query execution could in turn spawn a chart update. For more information about the order of events related to query execution and chart updates, see the following sections:

- “Query Execution” – 11 – 2
- “Chart Updates” – 11 – 3

The timer procedure is executed simultaneously with the optional query execution (described above) when the timer expires. If no query is to be executed at timer expiration, the timer procedure is the only event to occur.
Timers can be automatically activated when the display is run, or activated programmatically from a PL/SQL program unit using the Graphics PL/SQL built-in procedure OG_ACTIVATE_TIMER.

**Button Execution**

Button execution includes the following related events, occurring in the order shown below:

- setting the parameter (if drill-down)
- executing the detail chart query (if drill-down)
- executing the button procedure (if it exists)

**Setting the Parameter**

When a mouse event is detected on a chart element acting as a button (e.g., a pie slice) in a drill-down relationship, Graphics assigns a column value to a parameter using the column and parameter specified on the Drill-down page of the chart element’s property sheet.

For more information about drill-down relationships, see “Using Drill-down Charts” on page 8–26.

**Executing the Detail Chart Query**

When a mouse event is detected on a chart element acting as a button (e.g., a pie slice) in a drill-down relationship, Graphics executes the detail chart query specified on the Drill-down page of the chart element’s property sheet.

For more information about drill-down relationships, see “Using Drill-down Charts” on page 8–26.

Query execution could in turn spawn a chart update. For more information about the order of events related to query execution and chart updates, see the following sections:

- “Query Execution” – 11–2
- “Chart Updates” – 11–3

**Executing the Button Procedure**

When a mouse event is detected on any button object, the button procedure (if it exists) for that object is executed.

**Close Display Triggers**

Close Display triggers are fired immediately before the display is closed. Close Display triggers are the last triggers to fire.
Timers can be automatically activated when the display is run, or activated programmatically from a PL/SQL program unit using the Graphics PL/SQL built-in procedure OG_ACTIVATE_TIMER.

Button execution includes the following related events, occurring in the order shown below:

- setting the parameter (if drill-down)
- executing the detail chart query (if drill-down)
- executing the button procedure (if it exists)

When a mouse event is detected on a chart element acting as a button (e.g., a pie slice) in a drill-down relationship, Graphics assigns a column value to a parameter using the column and parameter specified on the Drill-down page of the chart element's property sheet.

For more information about drill-down relationships, see "Using Drill-down Charts" on page 8–26.

When a mouse event is detected on a chart element acting as a button (e.g., a pie slice) in a drill-down relationship, Graphics executes the detail chart query specified on the Drill-down page of the chart element's property sheet.

For more information about drill-down relationships, see "Using Drill-down Charts" on page 8–26.

Query execution could in turn spawn a chart update. For more information about the order of events related to query execution and chart updates, see the following sections:

- "Query Execution" – 11–2
- "Chart Updates" – 11–3

When a mouse event is detected on any button object, the button procedure (if it exists) for that object is executed.

Close Display triggers are fired immediately before the display is closed. Close Display triggers are the last triggers to fire.
Referencing a Trigger

For each type of trigger, a corresponding assignment/reference node appears in the Object Navigator. These nodes represent which procedures, if any, are assigned to a particular trigger.

You can assign an existing program unit to a trigger by selecting the program unit object in the Navigator, and dragging it directly beneath one of the trigger assignment nodes. When you release the mouse, the program unit name appears with the trigger node.

The following table shows which trigger procedure assignment nodes are associated with what objects in the Navigator:

<table>
<thead>
<tr>
<th>Trigger Type</th>
<th>Object Node in Navigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Display</td>
<td>Display</td>
</tr>
<tr>
<td>Post Execution</td>
<td>Query</td>
</tr>
<tr>
<td>Format</td>
<td>Chart Element</td>
</tr>
<tr>
<td>Timer</td>
<td>Query, Timer</td>
</tr>
<tr>
<td>Button</td>
<td>Object, Chart Element, Layer, etc.</td>
</tr>
<tr>
<td>Close Display</td>
<td>Display</td>
</tr>
</tbody>
</table>

At any time, you can double-click one of the trigger nodes to create a new trigger procedure with the Program Unit editor, or edit the currently assigned trigger procedure.

Using Open Display Triggers

An Open Display trigger executes a PL/SQL procedure when the display is first run. For example, you could define an Open Display trigger that sets the initial state of the display by executing queries and creating charts, activating timers, etc.

Creating Open Display Triggers

To create an open display trigger:

1. Double-click on the Open Display object in the Navigator to show the Program Unit editor.
   
   The Program Unit editor appears containing a default trigger procedure template (initially named OGRETRIGGERPROC).x

2. Rename the trigger procedure or keep the default, then enter the body of the procedure.

3. Choose the Compile button to compile the program unit.
Creating Open Display Triggers

Referencing a Trigger

For each type of trigger, a corresponding assignment/reference node appears in the Object Navigator. These nodes represent which procedures, if any, are assigned to a particular trigger.

You can assign an existing program unit to a trigger by selecting the program unit object in the Navigator, and dragging it directly beneath one of the trigger assignment nodes. When you release the mouse, the program unit name appears with the trigger node.

The following table shows which trigger procedure assignment nodes are associated with what objects in the Navigator:

<table>
<thead>
<tr>
<th>Trigger Type</th>
<th>Object Node in Navigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Display</td>
<td>Display</td>
</tr>
<tr>
<td>Post Execution Query</td>
<td></td>
</tr>
<tr>
<td>Format Chart Element</td>
<td></td>
</tr>
<tr>
<td>Timer Query, Timer</td>
<td>Button Object, Chart Element, Layer, etc.</td>
</tr>
<tr>
<td>Close Display</td>
<td>Display</td>
</tr>
</tbody>
</table>

At any time, you can double-click one of the trigger nodes to create a new trigger procedure with the Program Unit editor, or edit the currently assigned trigger procedure.

Using Open Display Triggers

An Open Display trigger executes a PL/SQL procedure when the display is first run. For example, you could define an Open Display trigger that sets the initial state of the display by executing queries and creating charts, activating timers, etc.

To create an open display trigger:

1. Double-click on the Open Display object in the Navigator to show the Program Unit editor. The Program Unit editor appears containing a default trigger procedure template (initially named OGTRIGGERPROC x).
2. Rename the trigger procedure or keep the default, then enter the body of the procedure.
3. Choose the Compile button to compile the program unit.
If any errors are encountered, correct them and recompile.

4. Choose the Close button to close the Program Unit editor.

**Example**

The following is an example of an Open Display trigger that, when it fires, hides the active layer, activates “layer_buttn”, and executes a query:

```plsql
PROCEDURE open_disp IS
    old_layr  OG_LAYER;
    nu_layr   OG_LAYER;
    the_qry   OG_QUERY;
BEGIN
    old_layer:=OG_GET_LAYER('layer_home');
    nu_layer:=OG_GET_LAYER('layer_buttn');
    the_qry:=OG_GET_QUERY('query0');
    OG_ACTIVATE_LAYER(nu_layr);
    OG_HIDE_LAYER(old_layr);
    OG_EXECUTE_QUERY(the_qry);
END:
```


**Modifying References to Trigger Procedures**

Graphics allows you to create multiple trigger procedures and assign any one to an Open Display trigger while developing your display.

To modify an Open Display trigger’s reference to a procedure:

1. Choose **Tools—>Display** to open the Display Property sheet.
2. Select an available trigger procedure from the Open Trigger drop-down list, or select **None** if no behavior is desired when the display is opened.
3. Choose the OK button to apply the change and close the Display property sheet.

**Using Post-Execution Triggers**

Post-execution triggers are PL/SQL procedures associated with queries. They fire immediately after their associated query is executed.
If any errors are encountered, correct them and recompile.

4. Choose the Close button to close the Program Unit editor.

The following is an example of an Open Display trigger that, when it fires, hides the active layer, activates "layer_buttn", and executes a query:

```plsql
PROCEDURE open_disp IS
    old_layr  OG_LAYER;
    nu_layr   OG_LAYER;
    the_qry   OG_QUERY;
BEGIN
    old_layer:=OG_GET_LAYER('layer_home');
    nu_layer:=OG_GET_LAYER('layer_buttn');
    the_qry:=OG_GET_QUERY('query0');
    OG_ACTIVATE_LAYER(nu_layr);
    OG_HIDE_LAYER(old_layr);
    OG_EXECUTE_QUERY(the_qry);
END:
```


Graphics allows you to create multiple trigger procedures and assign any one to an Open Display trigger while developing your display.

To modify an Open Display trigger’s reference to a procedure:

2. Select an available trigger procedure from the Open Trigger drop-down list, or select None if no behavior is desired when the display is opened.
3. Choose the OK button to apply the change and close the Display property sheet.

Using Post-Execution Triggers

Post-execution triggers are PL/SQL procedures associated with queries. They fire immediately after their associated query is executed.
Creating Post-Execution Triggers

To create a post-execution trigger:

1. Double-click on the Post Execution object under the query object in the Navigator.
   
   The Program Unit editor appears, containing a default trigger procedure template (initially named `OGQUERYPROCx`).

2. Rename the trigger procedure or keep the default, then enter the body of the procedure.

3. Choose the Compile button to compile the program unit.
   
   If any errors are encountered, correct them and recompile.

4. Choose the Close button to close the Program Unit editor.

Example

The following is an example of a post-execution trigger that activates a timer to update a stock report at set intervals:

```pascal
PROCEDURE post_exec IS
    the_timer  OG_TIMER;
BEGIN
    the_timer:=OG_GET_TIMER('check_stock');
    OG_ACTIVATE_TIMER(the_timer);
END;
```

Modifying References to Post-Execution Triggers

Graphics allows you to create multiple post-execution trigger procedures and easily assign them to any query in your display.

To modify a query’s reference to a post-execution trigger:

1. Double-click on a query object in the Navigator, or select the query and choose the Properties command from the pop-up menu to show the Query property sheet.
   
   Alternatively, you can choose Tools—>Queries to show the Query property sheet for the current query.

2. Select the Options tab to show the Options property page.

3. Select any available post-execution trigger procedure from the Post-Execution Trigger drop-down list, or select None if no post-execution behavior is desired.

4. Choose the OK button to apply the change and close the Object property sheet.
**Creating Post-Execution Triggers**

**Example**

**Modifying References to Post-Execution Triggers**

To create a post-execution trigger:

1. Double-click on the Post Execution object under the query object in the Navigator.
   - The Program Unit editor appears, containing a default trigger procedure template (initially named `OGQUERYPROC`).
2. Rename the trigger procedure or keep the default, then enter the body of the procedure.
3. Choose the Compile button to compile the program unit.
   - If any errors are encountered, correct them and recompile.
4. Choose the Close button to close the Program Unit editor.

The following is an example of a post-execution trigger that activates a timer to update a stock report at set intervals:

```plsql
PROCEDURE post_exec IS
    the_timer  OG TIMER;
BEGIN
    the_timer:=OG_GET_TIMER('check_stock');
    OG_ACTIVATE_TIMER(the_timer);
END;
```

Graphics allows you to create multiple post-execution trigger procedures and easily assign them to any query in your display.

To modify a query's reference to a post-execution trigger:

1. Double-click on a query object in the Navigator, or select the query and choose the Properties command from the pop-up menu to show the Query property sheet.
2. Select the Options tab to show the Options property page.
3. Select any available post-execution trigger procedure from the Post-Execution Trigger drop-down list, or select None if no post-execution behavior is desired.
4. Choose the OK button to apply the change and close the Object property sheet.
Using Format Triggers

A format trigger is a PL/SQL procedure associated with a chart element. Format triggers are executed just before the chart element is drawn, allowing you to dynamically change the formatting attributes of the element. For example, you could define a format trigger to color a chart’s bars green if they exceed a certain value.

Creating Format Triggers

To create a format trigger:

1. Double-click on the chart element you want to format, or select the chart element and choose the Properties command from the pop-up menu, to show the chart element’s Object property sheet.

2. Choose the New button that appears to the right of the Format Trigger drop-down list.

   The Program Unit editor appears, containing a default format trigger procedure template (initially named OGFORMATTRIGx).

3. Rename the format trigger procedure or keep the default, then enter the body of the procedure.

4. Choose the Compile button to compile the program unit.

   If any errors are encountered, correct them and recompile.

5. Choose the Close button to close the Program Unit editor.

Example

The following is an example of a format trigger that changes the color of the chosen chart element (e.g., pie slice or bar) to red if its value is greater than 200:

```
PROCEDURE format_point (elem IN og_object, query IN og_query) IS
  st_price  NUMBER;
BEGIN
  st_price:=OG_GET_NUMCELL(query, OG_NEWDATA, ’sell_pr’);
  IF st_price > 200 THEN
    OG_SET_BFCOLOR (elem, ’red’);
  END IF;
END:
```


Modifying References to Format Triggers

Graphics allows you to create multiple format trigger procedures and access them easily from any valid chart element’s property sheet.
A format trigger is a PL/SQL procedure associated with a chart element. Format triggers are executed just before the chart element is drawn, allowing you to dynamically change the formatting attributes of the element. For example, you could define a format trigger to color a chart's bars green if they exceed a certain value.

To create a format trigger:

1. Double-click on the chart element you want to format, or select the chart element and choose the Properties command from the pop-up menu, to show the chart element's Object property sheet.
2. Choose the New button that appears to the right of the Format Trigger drop-down list. The Program Unit editor appears, containing a default format trigger procedure template (initially named OGFORMATTRIG x).
3. Rename the format trigger procedure or keep the default, then enter the body of the procedure.
4. Choose the Compile button to compile the program unit. If any errors are encountered, correct them and recompile.
5. Choose the Close button to close the Program Unit editor.

The following is an example of a format trigger that changes the color of the chosen chart element (e.g., pie slice or bar) to red if its value is greater than 200:

```plsql
PROCEDURE format_point (elem IN og_object, query IN og_query) IS
    st_price  NUMBER;
BEGIN
    st_price:=OG_GET_NUMCELL(query, OG_NEWDATA, 'sell_pr');
    IF st_price > 200 THEN
        OG_SET_BFCOLOR (elem, 'red');
    END IF;
END:
```


Graphics allows you to create multiple format trigger procedures and access them easily from any valid chart element's property sheet.
To modify an element’s reference to a format trigger procedure:

1. Double-click on the chart element, or select the element and choose the Properties command from the pop-up menu to show the chart element’s Object property sheet.
2. Select an available format trigger procedure from the Format Trigger drop-down list, or select None if no special formatting is desired.
3. Choose the OK button to apply the change and close the Object property sheet.

Using Timer Triggers

A timer executes a PL/SQL trigger procedure at each occurrence of a specified time interval. For example, you could define a timer that fires every minute and executes a PL/SQL procedure that updates a chart.

Creating Timer Triggers

To create a timer trigger:

1. Select the Timers node in the Object Navigator.
2. Choose the Create button or **Navigator—>Create** to show the Timer Triggers dialog box.
   
   **Suggestion:** If no timers currently exist, double-click on the Timers node in the Navigator to show a new default timer in the Timer Triggers dialog box.

   Alternatively, you can select **Tools—>Timers** and choose the New button to create a new timer.

   This dialog shows the default settings for a new timer (initially named **timer x**).
3. If you wish to create a new procedure, choose the New button that appears to the right of the Procedure drop-down list.
   
   The Program Unit editor appears, containing a default trigger procedure template (initially named **OGTRIGGERPROC x**).
4. Rename the trigger procedure or keep the default, then enter the body of the procedure.
5. Choose the Compile button to compile the new procedure.
6. Choose the Close button to close the Program Unit editor and return the the Timer Triggers dialog.
Creating Timer Triggers

Using Triggers

To modify an element's reference to a format trigger procedure:

1. Double-click on the chart element, or select the element and choose the Properties command from the pop-up menu to show the chart element's Object property sheet.

2. Select an available format trigger procedure from the Format Trigger drop-down list, or select None if no special formatting is desired.

3. Choose the OK button to apply the change and close the Object property sheet.

Using Timer Triggers

A timer executes a PL/SQL trigger procedure at each occurrence of a specified time interval. For example, you could define a timer that fires every minute and executes a PL/SQL procedure that updates a chart.

To create a timer trigger:

1. Select the Timers node in the Object Navigator.

2. Choose the Create button or Navigator —> Create to show the Timer Triggers dialog box.

   Suggestion: If no timers currently exist, double-click on the Timers node in the Navigator to show a new default timer in the Timer Triggers dialog box. Alternatively, you can select Tools —> Timers and choose the New button to create a new timer. This dialog shows the default settings for a new timer (initially named timer x).

3. If you wish to create a new procedure, choose the New button that appears to the right of the Procedure drop-down list.

   The Program Unit editor appears, containing a default trigger procedure template (initially named OGTRIGGERPROC x).

4. Rename the trigger procedure or keep the default, then enter the body of the procedure.

5. Choose the Compile button to compile the new procedure.

6. Choose the Close button to close the Program Unit editor and return to the Timer Triggers dialog.
7. If you have an existing program unit you want to assign, select it from the Procedure drop-down list.

8. Modify the timer’s name and interval, if desired.

9. Use the Active check-box to determine if the timer will be activated automatically when the display is run (checked), or activated from another PL/SQL program unit (unchecked).

You can use the Graphics PL/SQL built-in procedure OG_ACTIVATE_TIMER to activate the timer from another program unit. For example, you may want to allow the user to activate the timer by pressing a button in the runtime display.


Example

The following is an example of a timer trigger:

```
PROCEDURE check_stock IS
  the_query  OG_QUERY;
  the_chart  OG_OBJECT;
BEGIN
  the_query:=OG_GET_QUERY('stock_val');
  the_chart:=OG_GET_CHART('stock');
  OG_EXECUTE_QUERY(the_query);
  OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END:
```

Modifying References to Procedures

You can easily assign any PL/SQL procedure stored in the display to an existing timer. To modify a timer’s reference to a trigger procedure:

1. Double-click on the timer you wish to modify, or choose Tools—>Timers to show the Timer Triggers dialog box.

2. Select an available procedure from the Procedure drop-down list, or select None if no behavior is desired when the timer expires.

3. Choose the OK button to apply the change and close the Timer Triggers dialog.
7. If you have an existing program unit you want to assign, select it from the Procedure drop-down list.
8. Modify the timer's name and interval, if desired.
9. Use the Active check-box to determine if the timer will be activated automatically when the display is run (checked), or activated from another PL/SQL program unit (unchecked).

You can use the Graphics PL/SQL built-in procedure OG_ACTIV ATE_TIMER to activate the timer from another program unit. For example, you may want to allow the user to activate the timer by pressing a button in the runtime display.


The following is an example of a timer trigger:

```plsql
PROCEDURE check_stock IS
    the_query  OG_QUERY;
    the_chart  OG_OBJECT;
BEGIN
    the_query:=OG_GET_QUERY('stock_val');
    the_chart:=OG_GET_CHART('stock');
    OG_EXECUTE_QUERY(the_query);
    OG_UPDATE_CHART(the_chart, OG_ALL_CHUPDA);
END:
```

You can easily assign any PL/SQL procedure stored in the display to an existing timer. To modify a timer's reference to a trigger procedure:

1. Double-click on the timer you wish to modify, or choose Tools —> Timers to show the Timer Triggers dialog box.
2. Select an available procedure from the Procedure drop-down list, or select None if no behavior is desired when the timer expires.
3. Choose the OK button to apply the change and close the Timer Triggers dialog.
Using Button Procedures

Any object you create or import into Graphics can be assigned a button procedure, including layers, graphic objects, chart elements, symbols, and text objects. Button procedures are executed when a specified “mouse event” (e.g., mouse down) involving the object is detected.

Since buttons are invoked automatically and do not return values, button functions are not supported. (For simplicity, all references to an “object” in this section should be taken to mean an “object or layer.”)

Graphics recognizes the following mouse events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouse down</td>
<td>detected when the mouse button is pressed</td>
</tr>
<tr>
<td>mouse up</td>
<td>detected when the mouse button is released</td>
</tr>
<tr>
<td>mouse move down</td>
<td>detected when the mouse is moved while the button is held down</td>
</tr>
<tr>
<td>mouse move up</td>
<td>detected when the mouse is moved while the button is not held down (may be detected only by a layer)</td>
</tr>
</tbody>
</table>

Event–Active Object

The mouse down event is the most important, since it determines which object becomes “event–active.” The event–active object “receives” the mouse down event and every subsequent event until the next mouse down (which then determines a new event–active object).

Only an object that has a procedure associated with it and is set to receive at least one mouse event is eligible to become the event–active object. (You can set an object’s procedure and desired events either in its Properties dialog or via PL/SQL.) Note that an object does not have to be set to receive a mouse down event in order to become event–active.

When Graphics detects a mouse down event, it uses the following search strategy to find the first object that meets the above criteria:

- the topmost object (on the active layer) to which the mouse is pointing
- the parent group of the topmost object (note that you can assign a procedure to a group object that is invoked when any of its children are clicked on — as long as the children have not been assigned their own procedures, which would take priority)
- the parent’s parent, and so on up the group tree of the topmost object
- the active layer
Any object you create or import into Graphics can be assigned a button procedure, including layers, graphic objects, chart elements, symbols, and text objects. Button procedures are executed when a specified "mouse event" (e.g., mouse down) involving the object is detected. Since buttons are invoked automatically and do not return values, button functions are not supported. (For simplicity, all references to an "object" in this section should be taken to mean an "object or layer.") Graphics recognizes the following mouse events:

<table>
<thead>
<tr>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouse down</td>
</tr>
<tr>
<td>detected when the mouse button is pressed</td>
</tr>
<tr>
<td>mouse up</td>
</tr>
<tr>
<td>detected when the mouse button is released</td>
</tr>
<tr>
<td>mouse down move</td>
</tr>
<tr>
<td>detected when the mouse is moved while the button is held down</td>
</tr>
<tr>
<td>mouse up move</td>
</tr>
<tr>
<td>detected when the mouse is moved while the button is not held down (may be detected only by a layer)</td>
</tr>
</tbody>
</table>

The mouse down event is the most important, since it determines which object becomes "event–active." The event–active object "receives" the mouse down event and every subsequent event until the next mouse down (which then determines a new event–active object).

Only an object that has a procedure associated with it and is set to receive at least one mouse event is eligible to become the event–active object. (You can set an object's procedure and desired events either in its Properties dialog or via PL/SQL.) Note that an object does not have to be set to receive a mouse down event in order to become event–active. When Graphics detects a mouse down event, it uses the following search strategy to find the first object that meets the above criteria:

- the topmost object (on the active layer) to which the mouse is pointing
- the parent group of the topmost object (note that you can assign a procedure to a group object that is invoked when any of its children are clicked on — as long as the children have not been assigned their own procedures, which would take priority)
- the parent's parent, and so on up the group tree of the topmost object
- the active layer
The first object that Graphics finds that meets the stated criteria becomes the event–active object. If no qualified objects are found, the mouse down event is ignored. Note that the active layer can be made event–active by clicking on a spot on the layout that is not occupied by an object that resides on that layer.

Once the event–active object has been chosen, Graphics checks if that object is set to receive mouse down events. If so, it receives the mouse down event that made it event–active, and the procedure is executed. Graphics then does the same for all subsequently detected events, invoking the procedure only for the events the object is set to receive.

When Graphics detects a mouse up event, the event–active object is “deactivated.” It no longer receives any mouse events, unless it is again made event–active by a mouse down.

**A Special Note About Layers**

A layer is able to receive one event that an object cannot: mouse move up. If the active layer has a procedure associated with it and is set to receive this event, it will do so while no other object is event–active.

**Button Procedure Specification**

A button procedure must have exactly four specific arguments. Graphics supplies the following default procedure specification when you create a new button procedure:

```plaintext
PROCEDURE proc_name
(buttonobj   IN  og_object,
 hitobj      IN  og_object,
 win         IN  og_window,
 eventinfo   IN  og_event) IS

where:

proc_name Is the name of the procedure. Graphics supplies a default unique name for the procedure; however, you may change it.

buttonobj Is the handle to the event–active object (i.e., the object that has this button procedure associated with it).

hitobj Is the handle to the topmost object on the active layer that the mouse is pointing to when the event is detected. This may differ from buttonobj if the object that was clicked on is a child of a group that has a button procedure; in that case, buttonobj is the group object and hitobj is the child.

win Is the handle to the window in which the event–active object became event–active.
```
The first object that Graphics finds that meets the stated criteria becomes the event–active object. If no qualified objects are found, the mouse down event is ignored. Note that the active layer can be made event–active by clicking on a spot on the layout that is not occupied by an object that resides on that layer.

Once the event–active object has been chosen, Graphics checks if that object is set to receive mouse down events. If so, it receives the mouse down event that made it event–active, and the procedure is executed. Graphics then does the same for all subsequently detected events, invoking the procedure only for the events the object is set to receive.

When Graphics detects a mouse up event, the event–active object is “deactivated.” It no longer receives any mouse events, unless it is again made event–active by a mouse down.

A layer is able to receive one event that an object cannot: mouse move up. If the active layer has a procedure associated with it and is set to receive this event, it will do so while no other object is event–active.

A button procedure must have exactly four specific arguments. Graphics supplies the following default procedure specification when you create a new button procedure:

\[
\text{PROCEDURE proc_name} \\
\text{( buttonobj IN og_object,)} \\
\text{hitobj IN og_object,} \\
\text{win IN og_window,} \\
\text{eventinfo IN og_event) IS} \\
\text{where:} \\
\text{Is the name of the procedure. Graphics supplies a default unique name for the procedure; however, you may change it.} \\
\text{Is the handle to the event–active object (i.e., the object that has this button procedure associated with it).} \\
\text{Is the handle to the topmost object on the active layer that the mouse is pointing to when the event is detected. This may differ from buttonobj if the object that was clicked on is a child of a group that has a button procedure; in that case, buttonobj is the group object and hitobj is the child.} \\
\text{Is the handle to the window in which the event–active object became event–active.} \\
\text{proc_name buttonobj hitobj win}
\]
**eventinfo** Is a record containing information about the mouse event that invoked this procedure. This field is of the Graphics built-in datatype OG_EVENT.

You can use the button procedure’s *eventinfo* argument to have the procedure perform different actions on different events. Note that a button procedure is invoked by any event it is set to receive. If there are no conditions placed on the procedure’s actions, all statements not limited to a specific condition are executed.

**Example**

The following button procedure moves the button object to the right and down two inches when it receives a mouse down event, and then rotates the object 45 degrees (using the mouse position as the center of rotation) when it receives a mouse up event. Note that even though the button object may also be set to receive mouse move down events, and therefore those events do invoke the procedure, this procedure simply ignores them.

```plsql
PROCEDURE move_and_rotate(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
    move_offset OG_POINT;
BEGIN
    IF eventinfo.event_type=OG_MOUSE_DOWN THEN
        move_offset.x := 2 * OG_INCH;
        move_offset.y := 2 * OG_INCH;
        OG_MOVE(buttonobj, move_offset);
    ELSIF eventinfo.event_type=OG_MOUSE_UP THEN
        OG_ROTATE(buttonobj, eventinfo.mouse_position, 45);
    END IF;
END;
```


**Creating Button Procedures**

To create a button procedure for an object:

1. Double-click on the object, or select the object and choose Tools—>Properties to show the Object property sheet.
2. If the object is unnamed, type a name in the Name field.
3. Choose the New button that appears to the right of the Procedure drop-down list.

The Program Unit editor appears, containing a default button procedure template (initially named `OGBUTTONPROC`).
Creating Button Procedures

Is a record containing information about the mouse event that invoked this procedure. This field is of the Graphics built-in datatype OG_EVENT.

You can use the button procedure's eventinfo argument to have the procedure perform different actions on different events. Note that a button procedure is invoked by any event it is set to receive. If there are no conditions placed on the procedure's actions, all statements not limited to a specific condition are executed.

The following button procedure moves the button object to the right and down two inches when it receives a mouse down event, and then rotates the object 45 degrees (using the mouse position as the center of rotation) when it receives a mouse up event. Note that even though the button object may also be set to receive mouse move down events, and therefore those events do invoke the procedure, this procedure simply ignores them.

```plsql
PROCEDURE move_and_rotate(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event) IS
    move_offset  OG_POINT;
BEGIN
    IF eventinfo.event_type=OG_MOUSE_DOWN THEN
        move_offset.x := 2 * OG_INCH;
        move_offset.y := 2 * OG_INCH;
        OG_MOVE(buttonobj, move_offset);
    ELSIF eventinfo.event_type=OG_MOUSE_UP THEN
        OG_ROTATE(buttonobj, eventinfo.mouse_position, 45);
    END IF;
END;
```


To create a button procedure for an object:

1. Double-click on the object, or select the object and choose Tools —> Properties to show the Object property sheet.
2. If the object is unnamed, type a name in the Name field.
3. Choose the New button that appears to the right of the Procedure drop-down list.

The Program Unit editor appears, containing a default button procedure template (initially named OGBUTTONPROC x).
4. Rename the button procedure or accept the default, then enter the body of the procedure.

5. Choose the Compile button to compile the program unit.
   
   If any errors are encountered, correct them and recompile.

6. Choose the Close button to close the Program Unit editor.

**Example**

The following is an example of a button procedure that deactivates a timer:

```plsql
PROCEDURE stop_plot (buttonobj IN og_object, hitobj IN og_object, win IN og_object) IS
  the_timer  OG_TIMER;
BEGIN
  the_timer:=og_get_timer('stockchk');
  OG_DEACTIVATE_TIMER(the_timer);
END:
```

**Modifying References to Button Procedures**

Graphics allows you to create multiple button procedures and access them easily from any object’s property sheet while developing your displays.

To modify an object’s reference to a button procedure:

1. Double-click on the object, or select the object and choose Tools—>Properties to show the Object property sheet.

2. Select an available button procedure from the Procedure drop-down list, or select None if no button behavior is desired.

3. Choose the OK button to apply the change and close the Object property sheet.

**Using Close Display Triggers**

A Close Display trigger executes a PL/SQL procedure when the display is closed at runtime. For example, you could define a Close Display trigger to print the display or commit changes to the database.

Note that the display will not actually close until the Close Display trigger procedure has finished executing.

**Creating Close Display Triggers**

To create a Close Display trigger:

1. In the Object Navigator, double-click on the Close Trigger icon, or choose Tools—>Display to show the Display property sheet.
Example

Modifying References to Button Procedures

Creating Close Display Triggers

1. Double-click on the object, or select the object and choose Tools —> Properties to show the Object property sheet.

2. Select an available button procedure from the Procedure drop-down list, or select None if no button behavior is desired.

3. Choose the OK button to apply the change and close the Object property sheet.

Using Close Display Triggers

A Close Display trigger executes a PL/SQL procedure when the display is closed at runtime. For example, you could define a Close Display trigger to print the display or commit changes to the database.

Note that the display will not actually close until the Close Display trigger procedure has finished executing.

To create a Close Display trigger:

1. In the Object Navigator, double-click on the Close Trigger icon, or choose Tools —> Display to show the Display property sheet.
2. Choose the New button that appears to the right of the Close Trigger drop-down list.

   The Program Unit editor appears, containing a default trigger procedure template (initially named OGTRIGGERPROCx).

3. Rename the procedure, or keep the default, then enter the body of the procedure.

4. Choose the Compile button to compile the program unit.

   If any errors are encountered, correct them and recompile.

5. Choose the Close button to close the Program Unit editor.

Example

   The following is an example of a Close Display trigger that plays the specified sound when the display is closed:

   ```plsql
   PROCEDURE close_disp IS
      the_sound  OG_SOUND;
   BEGIN
      the_sound:=OG_GET_SOUND('playme');
      OG_PLAY_SOUND(the_sound);
   END;
   ```


Modifying References to Trigger Procedures

   Graphics allows you to create multiple trigger procedures and easily assign one as a close display trigger while developing your display.

   To modify a display’s reference to a close display trigger:


2. Select an available trigger procedure from the Close Trigger drop-down list, or select None if no behavior is desired when the display is closed.

3. Choose the OK button to apply the change and close the Display property sheet.
Modifying References to Trigger Procedures

2. Choose the New button that appears to the right of the Close Trigger drop-down list. The Program Unit editor appears, containing a default trigger procedure template (initially named OGTTRIGGERPROC x).

3. Rename the procedure, or keep the default, then enter the body of the procedure.

4. Choose the Compile button to compile the program unit. If any errors are encountered, correct them and recompile.

5. Choose the Close button to close the Program Unit editor.

The following is an example of a Close Display trigger that plays the specified sound when the display is closed:

```sql
PROCEDURE close_disp IS
    the_sound  OF_SOUND;
BEGIN
    the_sound:=OG_GET_SOUND('playme');
    OG_PLAY_SOUND(the_sound);
END;
```


Graphics allows you to create multiple trigger procedures and easily assign one as a close display trigger while developing your display.

To modify a display's reference to a close display trigger:

1. Choose Tools ➔ Display to open the Display property sheet.

2. Select an available trigger procedure from the Close Trigger drop-down list, or select None if no behavior is desired when the display is closed.

3. Choose the OK button to apply the change and close the Display property sheet.
Working with PL/SQL Program Units

Using PL/SQL in Graphics allows you greater control over the way your displays operate at runtime, making them more responsive to current data conditions and end-user actions. This chapter discusses how to integrate PL/SQL into your Graphics displays using program units and libraries.

The following topics are covered in this chapter:

- working with PL/SQL program units – 12 – 1
- working with PL/SQL libraries – 12 – 6

Working with PL/SQL Program Units

PL/SQL program units are code structures you can create using PL/SQL. You can write PL/SQL program units in Graphics, or you can use your own editor and import or export PL/SQL program units.

This section does not discuss the details of the PL/SQL programming language, or how to write PL/SQL program units. For more information about writing PL/SQL program units, see the following sections:

- “Using PL/SQL in Graphics” – 13 – 1
Working with PL/SQL Program Units

PL/SQL program units are code structures you can create using PL/SQL. You can write PL/SQL program units in Graphics, or you can use your own editor and import or export PL/SQL program units. This section does not discuss the details of the PL/SQL programming language, or how to write PL/SQL program units. For more information about writing PL/SQL program units, see the following sections:

• “Using PL/SQL in Graphics” – 13 – 1
This section describes the following tasks involved in using PL/SQL program units with Graphics:

- creating PL/SQL program units
- editing PL/SQL program units
- compiling PL/SQL program units
- renaming PL/SQL program units
- deleting PL/SQL program units
- checking consistency

To create a PL/SQL program unit:

1. Select the Program Units node in the Object Navigator.
2. Choose the Create button or **Navigator**—>**Create** to show the New Program Unit dialog.

   **Suggestion:** If no program units currently exist, double-click on the Program Units node to create a new program unit.

3. Select the type of program unit you wish to create, enter a name for the program unit, and choose the OK button.

   The PL/SQL Program Unit editor appears with your program unit name in the Name field.

4. Define your program unit.

   For more information about creating PL/SQL program units, see “Defining Program Units” in the chapter “Working with PL/SQL Constructs” in the *Procedure Builder Developer’s Guide*.

5. Choose the Compile button to compile the program unit.

6. Resolve any errors that may have been encountered.

   Your program unit must be error-free before it can be successfully compiled and used in your display.

7. Choose the Close button to close the Program Unit editor when you are finished.

**Using PL/SQL Built-ins**

Graphics provides many built-in extensions to PL/SQL, including datatypes, procedures, functions, exceptions, and constants. These built-ins are designed to account for many common operations PL/SQL can perform in your display.
This section describes the following tasks involved in using PL/SQL program units with Graphics:

- creating PL/SQL program units
- editing PL/SQL program units
- compiling PL/SQL program units
- renaming PL/SQL program units
- deleting PL/SQL program units
- checking consistency

To create a PL/SQL program unit:

1. Select the Program Units node in the Object Navigator.
2. Choose the Create button or
   Navigator —> Create to show the New Program Unit dialog.
   Suggestion: If no program units currently exist, double-click on the Program Units node to create a new program unit.
3. Select the type of program unit you wish to create, enter a name for the program unit, and choose the OK button. The PL/SQL Program Unit editor appears with your program unit name in the Name field.
4. Define your program unit.
   For more information about creating PL/SQL program units, see "Defining Program Units" in the chapter "Working with PL/SQL Constructs" in the Procedure Builder Developer's Guide.
5. Choose the Compile button to compile the program unit.
6. Resolve any errors that may have been encountered. Your program unit must be error-free before it can be successfully compiled and used in your display.
7. Choose the Close button to close the Program Unit editor when you are finished.

Graphics provides many built-in extensions to PL/SQL, including datatypes, procedures, functions, exceptions, and constants. These built-ins are designed to account for many common operations PL/SQL can perform in your display.
For more information about using Graphics PL/SQL built-ins, see chapter “Using PL/SQL in Graphics” on page 13 – 1.

**Using PL/SQL Exceptions**

To trap or isolate the success or failure of PL/SQL program units, you can use exceptions. You can declare, raise, and handle your own exceptions, or you can use exceptions that have be pre-defined in Graphics. For more information about using exceptions, see the PL/SQL User’s Guide and Reference.

**Editing PL/SQL Program Units**

To edit a PL/SQL program unit:

- Double-click on the program unit object you want to edit, or select it and choose Properties from the pop-up menu. The Program Unit editor appears, containing the source of the program unit.

**Compiling PL/SQL Program Units**

You can compile PL/SQL program units individually, or you can compile all of the program units in a display at once using the PL/SQL Batch Compiler.

Uncompiled program units will appear in the Object Navigator with an asterisk(‘) after the program unit name.

**Compiling Individual Program Units**

You can compile individual PL/SQL program units using the Program Unit editor.

1. Double-click on the program unit in the Object Navigator to open the Program Unit editor.

   Alternatively, you can choose Tools—>Program Units to show the Program Unit editor and then choose the program unit from the Name drop-down list.

   The Program Unit status bar reports whether the program unit has been successfully compiled.

2. Choose the Compile button to compile the program unit.

   If any errors are encountered, they are reported in the Program Unit editor beneath the program unit body. You can double-click on an error to navigate to its source in the program unit body.

   For more information about compiling program units, see “Compiling Program Units” in the chapter “Working with PL/SQL Constructs” in the Procedure Builder Developer’s Guide.

**Batch Compiling Program Units**

If you have multiple program units in a display, use this method to compile all of them at once.
Using PL/SQL Exceptions

Editing PL/SQL Program Units

Compiling PL/SQL Program Units

Compiling Individual Program Units

Batch Compiling Program Units

To trap or isolate the success or failure of PL/SQL program units, you can use exceptions. You can declare, raise, and handle your own exceptions, or you can use exceptions that have been predefined in Graphics. For more information about using exceptions, see the PL/SQL User’s Guide and Reference.

To edit a PL/SQL program unit:

- Double-click on the program unit object you want to edit, or select it and choose Properties from the pop-up menu. The Program Unit editor appears, containing the source of the program unit.

You can compile PL/SQL program units individually, or you can compile all of the program units in a display at once using the PL/SQL Batch Compiler.

Uncompiled program units will appear in the Object Navigator with an asterisk(*) after the program unit name.

You can compile individual PL/SQL program units using the Program Unit editor.

1. Double-click on the program unit in the Object Navigator to open the Program Unit editor.

Alternatively, you can choose Tools —> Program Units to show the Program Unit editor and then choose the program unit from the Name drop-down list.

The Program Unit status bar reports whether the program unit has been successfully compiled.

2. Choose the Compile button to compile the program unit.

If any errors are encountered, they are reported in the Program Unit editor beneath the program unit body. You can double-click on an error to navigate to its source in the program unit body.

For more information about compiling program units, see “Compiling Program Units” in the chapter “Working with PL/SQL Constructs” in the Procedure Builder Developer’s Guide.

If you have multiple program units in a display, use this method to compile all of them at once.
1. Choose File→Compile to batch compile any uncompiled program units in the display, or choose File→Compile All to (re)compile every program unit in the display. The Compile All command forces the recompilation of already compiled program units. The Compile dialog appears, providing a progress indicator and a scrollable field where any compilation errors encountered are recorded.

2. You can choose from the following buttons to interact with the compilation:
   - **Interrupt**: Temporarily stops the compilation.
   - **Resume**: Resumes batch compilation after an Interrupt operation.
   - **OK**: Accepts and closes the Compile dialog.
   - **Cancel**: Cancels the batch compilation.
   - **Go To Error**: Locates the selected compilation error and opens the associated program unit at the location of the error.

3. When you’re finished, choose the OK button to close the Compile dialog.

**Recompiling Program Units**

Compiled PL/SQL code is not portable among different systems. Therefore, to prepare a display for use on a different system, you need to recompile your source code on that system.

**Changing Object References in PL/SQL Program Units**

If you change the name of an object that is referenced within one or more program units, you can change the name of the object in the program unit using the Program Unit editor, as follows:

1. Double-click on the program unit that references the object to show the program unit in the Program Unit editor.
2. Choose **Edit→Search/Replace** to show the Search/Replace dialog.
3. Type the old object name in the Search field. You can specify a case-sensitive search, or use a regular expression to search by.
4. Type the new object name in the Replace With field.
5. Choose the Replace All button to replace all occurrences of the old object name with the new object name in the current program unit.
Recompiling Program Units

Changing Object References in PL/SQL Program Units

1. Choose File —> Compile to batch compile any uncompiled program units in the display, or choose File —> Compile All to (re)compile every program unit in the display. The Compile All command forces the recompilation of already compiled program units. The Compile dialog appears, providing a progress indicator and a scrollable field where any compilation errors encountered are recorded.

2. You can choose from the following buttons to interact with the compilation:
   - Temporarily stops the compilation.
   - Resumes batch compilation after an Interrupt operation.
   - Accepts and closes the Compile dialog.
   - Cancels the batch compilation.
   - Locates the selected compilation error and opens the associated program unit at the location of the error.

3. When you're finished, choose the OK button to close the Compile dialog.

Compiled PL/SQL code is not portable among different systems. Therefore, to prepare a display for use on a different system, you need to recompile your source code on that system.

If you change the name of an object that is referenced within one or more program units, you can change the name of the object in the program unit using the Program Unit editor, as follows:

1. Double-click on the program unit that references the object to show the program unit in the Program Unit editor.
2. Choose Edit —> Search/Replace to show the Search/Replace dialog.
3. Type the old object name in the Search field. You can specify a case–sensitive search, or use a regular expression to search by.
4. Type the new object name in the Replace With field.
5. Choose the Replace All button to replace all occurrences of the old object name with the new object name in the current program unit.
To search for and replace a name in all program units in the current scope (i.e. the Program Units node, or an open library node), choose Edit—>Search/Replace All.

Renaming PL/SQL Program Units

You can change the name of any program unit from the Program Unit editor.

Note: If you change the name of a program unit that is referenced in other program units, you need to edit those references to reflect the new name. Once you recompile the renamed program unit, other program units that reference it will lose their compiled status and will appear in the Object Navigator with an asterisk (*).

1. Double-click on the program unit object to show the Program Unit editor.
2. Edit the program unit to change the name.
3. Recompile the program unit.

Deleting PL/SQL Program Units

You can delete any existing program units from the Object Navigator, or from the Program Unit editor.

Note: If you delete a program unit that is referenced by other program units, you must remove those references and recompile the program units to reflect the change.

To delete a program unit from the Object Navigator:

- Select the program unit object and choose the Delete button or Navigator—>Delete. Choose Yes to acknowledge the alert.

To delete a program unit from the Program Unit editor:

1. Double-click on the program unit object, or choose Program Units from the pop-up menu, or choose Tools—>Program Unit Editor to show the Program Unit editor.
2. If necessary, select the program unit you wish to delete from the Name drop-down list.
3. Choose the Delete button to delete the program unit. An alert appears, prompting you to confirm the deletion.
4. Choose Yes to confirm.
5. Choose the Close button to close the Program Unit editor.
Renaming PL/SQL Program Units

Deleting PL/SQL Program Units

To search for and replace a name in all program units in the current scope (i.e. the Program Units node, or an open library node), choose Edit —> Search/Replace All.

You can change the name of any program unit from the Program Unit editor.

Note: If you change the name of a program unit that is referenced in other program units, you need to edit those references to reflect the new name. Once you recompile the renamed program unit, other program units that reference it will lose their compiled status and will appear in the Object Navigator with an asterisk (*).

1. Double-click on the program unit object to show the Program Unit editor.
2. Edit the program unit to change the name.
3. Recompile the program unit.

You can delete any existing program units from the Object Navigator, or from the Program Unit editor.

Note: If you delete a program unit that is referenced by other program units, you must remove those references and recompile the program units to reflect the change.

To delete a program unit from the Object Navigator:

- Select the program unit object and choose the Delete button or Navigator —> Delete. Choose Yes to acknowledge the alert.

To delete a program unit from the Program Unit editor:

1. Double-click on the program unit object, or choose Program Units from the pop-up menu, or choose Tools —> Program Unit Editor to show the Program Unit editor.
2. If necessary, select the program unit you wish to delete from the Name drop-down list.
3. Choose the Delete button to delete the program unit.
4. An alert appears, prompting you to confirm the deletion.
5. Choose Yes to confirm.
6. Choose the Close button to close the Program Unit editor.
If you've used PL/SQL in your display, you may want to verify that there are no inconsistencies between program units and references to those program units. An example of an inconsistency would be if you referenced a trigger procedure but failed to create it.

To check consistency:

1. Choose **File**—>**Check Consistency** to show the Consistency Checker dialog.

   Graphics looks for inconsistencies and reports them in the message display field.

2. When you’re finished, choose the OK button to close the Consistency Checker dialog.

**Working with PL/SQL Libraries**

PL/SQL libraries are collections of program units that you can create within Graphics. Once it’s created, you can attach a PL/SQL library to any display you create. In addition, you can grant access to libraries you have created so others can use them.

This section describes the tasks involved in using PL/SQL libraries with Graphics, including:

- creating libraries
- opening libraries
- modifying libraries
- saving libraries
- copying program units into a display from an open library
- attaching libraries
- detaching libraries
- granting and revoking access to libraries
- renaming libraries
- deleting libraries

For more information about working with PL/SQL libraries, see “Using PL/SQL Libraries” in the “Working with PL/SQL Constructs” chapter of the *Procedure Builder Developer's Guide*.

**Creating PL/SQL Libraries**

To create a PL/SQL library:
If you've used PL/SQL in your display, you may want to verify that there are no inconsistencies between program units and references to those program units. An example of an inconsistency would be if you referenced a trigger procedure but failed to create it.

To check consistency:

1. Choose File –> Check Consistency to show the Consistency Checker dialog. Graphics looks for inconsistencies and reports them in the message display field.
2. When you're finished, choose the OK button to close the Consistency Checker dialog.

Working with PL/SQL Libraries

PL/SQL libraries are collections of program units that you can create within Graphics. Once it's created, you can attach a PL/SQL library to any display you create. In addition, you can grant access to libraries you have created so others can use them.

This section describes the tasks involved in using PL/SQL libraries with Graphics, including:

- creating libraries
- opening libraries
- modifying libraries
- saving libraries
- copying program units into a display from an open library
- attaching libraries
- detaching libraries
- granting and revoking access to libraries
- renaming libraries
- deleting libraries

For more information about working with PL/SQL libraries, see "Using PL/SQL Libraries" in the "Working with PL/SQL Constructs" chapter of the Procedure Builder Developer's Guide.
Opening PL/SQL Libraries

To open a PL/SQL library stored in the file system or the database:

1. Select File—>Open to show the Filter dialog.
2. Select Libraries or All Files from the Display field.
3. Select either the Database or File System radio button in the Object Access field according to where the library is stored.
4. Choose the OK button to confirm your selections.
   - If your library is stored on the file system, your system file dialog appears. If your library is stored in the database, the Open from Database dialog appears.
5. Specify the name of the library you wish to open and accept the dialog to open the library.

   **Suggestion:** Libraries use the .pll file extension to distinguish them from other files.

Modifying the Content of a PL/SQL Library

Using the Object Navigator, you can easily modify the contents of an open PL/SQL library by adding and deleting program units.

Adding Existing Program Units to Open Libraries

To add program units to an open library using the Object Navigator:

1. Expand the subnodes of the open library you wish to modify.
2. Select the program unit you wish to add from the Displays node.
3. Click-and-drag the program unit’s icon below the Program Units subnode of the open library.

   **Suggestion:** If no libraries are currently open, you can double-click on the Libraries node in the Navigator to create a new library.

Now that you’ve created a library, you can add existing or create new program units to store in it.
Opening PL/SQL Libraries

Modifying the Content of a PL/SQL Library

Adding Existing Program Units to Open Libraries

12 – 7 Working with PL/SQL Program Units

1. Select the Libraries node in the Object Navigator.
2. Choose the Create button or Navigator —> Create.
A new open library, initially named LIB_xxx, is created.

Suggestion: If no libraries are currently open, you can double-click on the Libraries node in the Navigator to create a new library.

Now that you've created a library, you can add existing or create new program units to store in it.

To open a PL/SQL library stored in the file system or the database:

1. Select File —> Open to show the Filter dialog.
2. Select Libraries or All Files from the Display field.
3. Select either the Database or File System radio button in the Object Access field according to where the library is stored.
4. Choose the OK button to confirm your selections.

If your library is stored on the file system, your system file dialog appears. If your library is stored in the database, the Open from Database dialog appears.

5. Specify the name of the library you wish to open and accept the dialog to open the library.

Suggestion: Libraries use the .pll file extension to distinguish them from other files.

Using the Object Navigator, you can easily modify the contents of an open PL/SQL library by adding and deleting program units.

To add program units to an open library using the Object Navigator:

1. Expand the subnodes of the open library you wish to modify.
2. Select the program unit you wish to add from the Displays node.
3. Click-and-drag the program unit's icon below the Program Units subnode of the open library.
When you release the mouse button, the program unit is automatically copied into the open library. If a program unit of the same name and type already exists in the library, an alert prompts you to confirm that the original program unit be overwritten.

To create a new program unit in an open library:
1. Select the Program Units subnode beneath the open library node in the Object Navigator.
2. Choose the Create button, or Navigator—>Create to show the New Program Unit dialog.
   
   **Suggestion:** If no program units currently exist in the library, double-click on the Program Units subnode in the Navigator to show the New Program Unit dialog.
3. Type a name for the new program unit.
4. Select a type for the new program unit.
5. Choose the OK button to accept the New Program Unit dialog and show the Program Unit editor.
6. Define the new program unit.
7. Choose the Compile button to compile the new program unit.
8. Choose the Close button to close the Program Unit editor.
   
   Your new program unit appears below the Program Units subnode of the library.

To delete program units from an open library:
1. Expand the subnodes of the open library you wish to modify.
2. Select the program unit you wish to delete from the library.
3. Choose the Delete button or Navigator—>Delete.
   
   An alert appears, prompting you to confirm the deletion.
4. Choose the Yes button to confirm.

To save an open library:
1. Select the open library you wish to save.
2. Choose the Save button, File—>Save, or File—>Save As to show the Save Library dialog.
   
   Use the Save As command only if this is the first time you are saving the library.
Creating New Program Units in an Open Library

When you release the mouse button, the program unit is automatically copied into the open library. If a program unit of the same name and type already exists in the library, an alert prompts you to confirm that the original program unit be overwritten.

To create a new program unit in an open library:
1. Select the Program Units subnode beneath the open library node in the Object Navigator.
2. Choose the Create button, or Navigator —> Create to show the New Program Unit dialog.
3. Type a name for the new program unit.
4. Select a type for the new program unit.
5. Choose the OK button to accept the New Program Unit dialog and show the Program Unit editor.
6. Define the new program unit.
7. Choose the Compile button to compile the new program unit.
8. Choose the Close button to close the Program Unit editor.

Your new program unit appears below the Program Units subnode of the library.

To delete program units from an open library:
1. Expand the subnodes of the open library you wish to modify.
2. Select the program unit you wish to delete from the library.
3. Choose the Delete button or Navigator —> Delete.
4. An alert appears, prompting you to confirm the deletion.
5. Choose the Yes button to confirm.

To save an open library:
1. Select the open library you wish to save.
2. Choose the Save button, File —> Save, or File —> Save As to show the Save Library dialog.
3. Use the Save As command only if this is the first time you are saving the library.
3. Specify a name for the library and whether to save the library in the file system or the database.

4. Choose OK to save the library.

Libraries are automatically assigned the file extension `.pll` if saved in the file system.

**Loading Program Units into Displays**

You may wish to load individual program units into a display rather than attaching the PL/SQL library a program unit is stored in. There are no performance benefits to loading individual program units over attaching libraries.

Use the following criteria to determine whether you should load individual program units instead of attaching libraries:

- your libraries are very large and you may run into size limitations
- moving your display would require you to change all pathnames that point to the library
- your display only requires a few program units and it’s easier to maintain without the attached library

To load program units into a display from an open library:

- Click-and-drag the library program unit below the Program Units node of the display.

When you release the mouse button, the program unit is automatically copied into the display. If a program unit of the same name and type already exists in the display, an alert prompts you to confirm that the program unit be overwritten.

**Attaching PL/SQL Libraries to Displays**

You can attach a specific library to a display to use the contents of that library in the display. By attaching the library to the display, you ensure that the library and its contents will accompany the (runtime) display.

**Note:** You cannot attach open libraries. If the library you wish to attach is currently open, close it and then attach it.

The attached library is not static, that is, it is not a “snapshot” of the library at the time you attached it. The attached library is a link to the library and receives all updates you make to the library.

You can attach multiple libraries to a display.

To attach a PL/SQL library stored in the filesystem or the database:

1. In the Object Navigator, select the Attached Libraries node (within the display node).
3. Specify a name for the library and whether to save the library in the file system or the database.

4. Choose OK to save the library. Libraries are automatically assigned the file extension .pll if saved in the file system.

You may wish to load individual program units into a display rather than attaching the PL/SQL library a program unit is stored in. There are no performance benefits to loading individual program units over attaching libraries.

Use the following criteria to determine whether you should load individual program units instead of attaching libraries:

- your libraries are very large and you may run into size limitations
- moving your display would require you to change all pathnames that point to the library
- your display only requires a few program units and it's easier to maintain without the attached library

To load program units into a display from an open library:

- Click-and-drag the library program unit below the Program Units node of the display. When you release the mouse button, the program unit is automatically copied into the display. If a program unit of the same name and type already exists in the display, an alert prompts you to confirm that the program unit be overwritten.

You can attach a specific library to a display to use the contents of that library in the display. By attaching the library to the display, you ensure that the library and its contents will accompany the (runtime) display.

Note: You cannot attach open libraries. If the library you wish to attach is currently open, close it and then attach it.

The attached library is not static, that is, it is not a "snapshot" of the library at the time you attached it. The attached library is a link to the library and receives all updates you make to the library.

You can attach multiple libraries to a display.

To attach a PL/SQL library stored in the filesystem or the database:

1. In the Object Navigator, select the Attached Libraries node (within the display node).
2. Choose the Create button or **Navigator—>Create** to show the Attach Library dialog.

**Suggestion:** If no libraries are currently attached, double-click on the Attached Libraries node in the Navigator to show the Attach Library dialog.

3. Enter the name of the library you wish to attach.

4. Choose the Attach button to attach the library.

If you supply a pathname with the library (i.e., the library is not stored in your current or working directory), you will be prompted to strip the pathname.

If you decide not to strip the pathname, Graphics will look for the library only in the specified pathname. If you decide to distribute your display, Graphics may not be able to find the library.

If you decide to strip the pathname, Graphics will search for the library in the following search path:

- the user’s current or working directory
- the directory specified by the Graphics path environment variable
- the directory specified by the Oracle path environment variable

See the *Developer/2000 Installation Guide* for information on setting these environment variables on your platform.

**Detaching PL/SQL Libraries**

To detach a library that is attached to a display:

1. Select the library you wish to detach under the Attached Libraries node in the Object Navigator.

2. Choose the Delete button or **Navigator—>Delete**.

   An alert appears, prompting you to confirm the detachment.

3. Choose the OK button to detach the library.

**Granting and Revoking Access to PL/SQL Libraries**

You can grant and revoke access to PL/SQL libraries if you created the library or were granted access to it. Granting access to other users allows them to grant access to other users, and rename or modify the PL/SQL library.
Detaching PL/SQL Libraries

Granting and Revoking Access to PL/SQL Libraries

1. Choose the Create button or Navigator —> Create to show the Attach Library dialog.

Suggestion: If no libraries are currently attached, double-click on the Attached Libraries node in the Navigator to show the Attach Library dialog.

3. Enter the name of the library you wish to attach.

4. Choose the Attach button to attach the library.

If you supply a pathname with the library (i.e., the library is not stored in your current or working directory), you will be prompted to strip the pathname.

If you decide not to strip the pathname, Graphics will look for the library only in the specified pathname. If you decide to distribute your display, Graphics may not be able to find the library.

If you decide to strip the pathname, Graphics will search for the library in the following search path:
- the user’s current or working directory
- the directory specified by the Graphics path environment variable
- the directory specified by the Oracle path environment variable

See the Developer/2000 Installation Guide for information on setting these environment variables on your platform.

To detach an library that is attached to a display:

1. Select the library you wish to detach under the Attached Libraries node in the Object Navigator.

2. Choose the Delete button or Navigator —> Delete.

An alert appears, prompting you to confirm the detachment.

3. Choose the OK button to detach the library.

You can grant and revoke access to PL/SQL libraries if you created the library or were granted access to it. Granting access to other users allows them to grant access to other users, and rename or modify the PL/SQL library.
Granting and Revoking Access to Libraries in the File System

Some operating systems permit users to grant read, write, and execute permissions on individual files. For more information about how to set file permissions on your operating system, see your operating system documentation or check with your system administrator.

Granting and Revoking Access to Libraries in the Database

You can grant another user access (read/write/execute privileges) to a library you’ve created and stored in the database. However, the user must also have access to the Graphics database tables. For more information, consult your database administrator.

To grant or revoke access to a library:

1. Choose File—>Administration—>Access—>Library to show the Module Access in Database dialog box.
   This dialog contains a list of all libraries stored in the database that you have access to.
2. Select a library from the list.
3. Choose the OK button to show the Module Access dialog.
   This dialog contains a list of users who have access to the library.
4. To grant access, type the database username of the person to whom you want to grant access in the User field.
5. Choose the Add button to add the username to the Access list.
6. To revoke access from a user in the Access list, select the username in the list.
7. Choose the Remove button to remove the username from the Access list.
8. Choose the OK button to accept your changes and close the Module Access dialog.

Access privileges are updated for your library.

Renaming PL/SQL Libraries

You can rename libraries stored in the file system or in the database using the RENAME command from the PL/SQL Interpreter. You cannot rename PL/SQL libraries in the Object Navigator. If your library is stored in the file system, you can use your operating system’s mechanism for renaming files.

Renaming Libraries in the PL/SQL Interpreter

If you did not create the library yourself, and the library is stored in the database, you must have GRANT privilege to rename the library. If the library is stored in the file system, you must have the correct file permissions to rename the library.
Renaming Libraries in the PL/SQL Interpreter

Some operating systems permit users to grant read, write, and execute permissions on individual files. For more information about how to set file permissions on your operating system, see your operating system documentation or check with your system administrator.

You can grant another user access (read/write/execute privileges) to a library you've created and stored in the database. However, the user must also have access to the Graphics database tables. For more information, consult your database administrator.

To grant or revoke access to a library:

1. Choose File —> Administration —> Access —> Library to show the Module Access in Database dialog box. This dialog contains a list of all libraries stored in the database that you have access to.

2. Select a library from the list.

3. Choose the OK button to show the Module Access dialog. This dialog contains a list of users who have access to the library.

4. To grant access, type the database username of the person to whom you want to grant access in the User field.

5. Choose the Add button to add the username to the Access list.

6. To revoke access from a user in the Access list, select the username in the list.

7. Choose the Remove button to remove the username from the Access list.

8. Choose the OK button to accept your changes and close the Module Access dialog. Access privileges are updated for your library.

You can rename libraries stored in the file system or in the database using the RENAME command from the PL/SQL Interpreter. You cannot rename PL/SQL libraries in the Object Navigator. If your library is stored in the file system, you can use your operating system's mechanism for renaming files.

If you did not create the library yourself, and the library is stored in the database, you must have GRANT privilege to rename the library. If the library is stored in the file system, you must have the correct file permissions to rename the library.
To rename a PL/SQL library using the RENAME command:

1. Choose Tools—>PL/SQL Interpreter to show the PL/SQL Interpreter.
2. Type the RENAME command on the Interpreter command line as follows:
   
   ```pl/sql
   PL/SQL> .RENAME LIB library_name1 TO library_name2
   ```

   For more information about the RENAME command, see the “RENAME (Libraries)” section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

**Renaming Libraries in the File System**

You can use your operating system to rename a library that is stored on the file system. You must have the appropriate file permissions to be able to rename the file.

See your operating system documentation for the correct method of renaming files.

**Deleting PL/SQL Libraries**

You can delete a PL/SQL library using the DELETE command from the PL/SQL Interpreter command line if they are stored in the database, or using an operating system command if they are stored in the file system. You must have the necessary permissions on the library to delete it.

**Deleting Libraries from the File System**

You can use your operating system to delete a library that is stored as a file. You must have the appropriate file permissions to delete a file.

See your operating system documentation for the correct method of deleting files.

**Deleting Libraries from the Database**

To delete a library stored in the database:

1. Choose Tools—>PL/SQL Interpreter to show the PL/SQL Interpreter.
2. Enter the following command on the Interpreter command line:
   
   ```pl/sql
   PL/SQL> .DELETE LIBRARY library_name;
   ```

   where `library_name` is the name of the library you want to delete.
3. Close the PL/SQL Interpreter.
Renaming Libraries in the File System

Deleting PL/SQL Libraries

Deleting Libraries from the File System

Deleting Libraries from the Database

To rename a PL/SQL library using the RENAME command:

1. Choose Tools —> PL/SQL Interpreter to show the PL/SQL Interpreter.
2. Type the RENAME command on the Interpreter command line as follows:
   
   PL/SQL> .RENAME LIB library_name1 TO library_name2

   For more information about the RENAME command, see the "RENAME (Libraries)" section in the "Command Reference" chapter of the Procedure Builder Developer's Guide.

You can use your operating system to rename a library that is stored on the file system. You must have the appropriate file permissions to be able to rename the file.

See your operating system documentation for the correct method of renaming files.

You can delete a PL/SQL library using the DELETE command from the PL/SQL Interpreter command line if they are stored in the database, or using an operating system command if they are stored in the file system. You must have the necessary permissions on the library to delete it.

You can use your operating system to delete a library that is stored as a file. You must have the appropriate file permissions to delete a file.

See your operating system documentation for the correct method of deleting files.

To delete a library stored in the database:

1. Choose Tools —> PL/SQL Interpreter to show the PL/SQL Interpreter.
2. Enter the following command on the Interpreter command line:
   
   PL/SQL> .DELETE LIBRARY library_name;

   where library_name is the name of the library you want to delete.
3. Close the PL/SQL Interpreter.
Using PL/SQL in Graphics

This chapter describes concepts related to using PL/SQL within Graphics. It describes the following topics:

- about PL/SQL – 13 – 1
- specifying coordinate and size information – 13 – 5
- referencing Graphics objects – 13 – 5

For more information about working with PL/SQL program units and libraries, see “Working with PL/SQL Program Units” on page 12 – 1.

About PL/SQL

PL/SQL is a procedural extension of SQL that provides programming constructs such as loops, conditionals, and procedures. Using PL/SQL in Graphics allows you greater control over the way your displays operate, making them more responsive to current data, end-user actions, and other conditions present at runtime. You might use PL/SQL in your displays to do the following:

- Create and manipulate objects. For example, animate an object so that it appears to move across the layout.
CHAPTER 13

Using PL/SQL in Graphics

This chapter describes concepts related to using PL/SQL within Graphics. It describes the following topics:

• About PL/SQL
• Specifying coordinate and size information
• Referencing Graphics objects

For more information about working with PL/SQL program units and libraries, see “Working with PL/SQL Program Units” on page 12-1.

About PL/SQL

PL/SQL is a procedural extension of SQL that provides programming constructs such as loops, conditionals, and procedures. Using PL/SQL in Graphics allows you greater control over the way your displays operate, making them more responsive to current data, end-user actions, and other conditions present at runtime. You might use PL/SQL in your displays to do the following:

• Create and manipulate objects. For example, animate an object so that it appears to move across the layout.
• Control the order in which operations flow. For example, cause a specified layer to appear if a user clicks on an object.
• Control the timing of operations. For example, cause an action to occur at timed intervals.
• Conditionalize operations. For example, calculate current data values and show different objects, depending on the results.
• Define and interpret end-user actions. For example, display an error message if a user types an invalid statement.

This section describes the following PL/SQL constructs:
• program units
• built-ins
• exceptions
• libraries
• same-name resolution

Program Units

Program units are code structures you can create using PL/SQL. You can use Graphics to write PL/SQL program units, or you can use your own editor and import or export program units.

There are two types of PL/SQL program units you can use in Graphics:
• subprograms
• packages

After using the Object Navigator to create or access a program unit, you can edit it using the PL/SQL Program Unit editor. You use the editor to write and debug your code. For more information about using the editor, see “Using the Program Unit Editor” in the “Working with PL/SQL Constructs” chapter of the Procedure Builder Developer’s Guide.

Subprograms

Subprograms are PL/SQL procedures and functions you can create in Graphics.

A procedure performs a specified sequence of actions. For example, a procedure might update a chart based on new data. There are two special types of Graphics procedures you can create:

Button Procedure Associated with an object or layer and executed when a specific mouse event occurs at runtime: mouse button up/down, or mouse move with button up/down.
• Control the order in which operations flow. For example, cause a specified layer to appear if a user clicks on an object.
• Control the timing of operations. For example, cause an action to occur at timed intervals.
• Conditionalize operations. For example, calculate current data values and show different objects, depending on the results.
• Define and interpret end-user actions. For example, display an error message if a user types an invalid statement.

This section describes the following PL/SQL constructs:
• program units
• built-ins
• exceptions
• libraries
• same-name resolution

Program units are code structures you can create using PL/SQL. You can use Graphics to write PL/SQL program units, or you can use your own editor and import or export program units.

There are two types of PL/SQL program units you can use in Graphics:
• subprograms
• packages

After using the Object Navigator to create or access a program unit, you can edit it using the PL/SQL Program Unit editor. You use the editor to write and debug your code. For more information about using the editor, see "Using the Program Unit Editor" in the "Working with PL/SQL Constructs" chapter of the Procedure Builder Developer's Guide.

Subprograms are PL/SQL procedures and functions you can create in Graphics.

A **procedure** performs a specified sequence of actions. For example, a procedure might update a chart based on new data. There are two special types of Graphics procedures you can create:
- Associated with an object or layer and executed when a specific mouse event occurs at runtime: mouse button up/down, or mouse move with button up/down.
- **Button Procedure**
Trigger Procedure  Executed when a trigger fires. Possible triggers: the display is opened, the display is closed, a timed interval passes, a query is executed.

A function performs a specified sequence of actions and return a value. For example, a function might calculate and return the sales tax for an item. One type of Graphics function you can create is a query filter, which filters data returned by a query and charts the values meeting a specified condition.

Packages  A package groups logically–related PL/SQL procedures, functions, datatypes, variables, constants, exceptions, cursors, etc. Constructs defined in a package are commonly referred to as “packaged”; e.g., packaged functions or packaged procedures. When you reference a packaged construct, precede each appearance of its name with the package name followed by a period (.)

Built-ins  Graphics provides extensions to the PL/SQL language that are used specifically for customizing the behavior of Graphics applications. These constructs, called built–ins, are implemented as extensions to the STANDARD PL/SQL package.

Graphics provides the following types of built–in constructs:
- datatypes
- global variables
- subprograms
- exceptions

You can use any PL/SQL built-ins within your own program units, but you cannot edit the built-ins themselves.

Note: The Graphics built–ins have been designed to give you programmatic control over specific features and functionality. They may be used only when defining Graphics displays.


Exceptions  To trap the success or failure of PL/SQL program units, you can use exceptions. For example, you can display an error if a user performs an invalid action at runtime. There are two types of exceptions:
- user-defined
- pre-defined
Executed when a trigger fires. Possible triggers:

- the display is opened,
- the display is closed,
- a timed interval passes,
- a query is executed.

A function performs a specified sequence of actions and returns a value. For example, a function might calculate and return the sales tax for an item. One type of Graphics function you can create is a query filter, which filters data returned by a query and charts the values meeting a specified condition.

A package groups logically-related PL/SQL procedures, functions, datatypes, variables, constants, exceptions, cursors, etc. Constructs defined in a package are commonly referred to as "packaged"; e.g., packaged functions or packaged procedures. When you reference a packaged construct, precede each appearance of its name with the package name followed by a period (.)

Graphics provides extensions to the PL/SQL language that are used specifically for customizing the behavior of Graphics applications. These constructs, called built-ins, are implemented as extensions to the STANDARD PL/SQL package.

Graphics provides the following types of built-in constructs:

- datatypes
- global variables
- subprograms
- exceptions

You can use any PL/SQL built-ins within your own program units, but you cannot edit the built-ins themselves.

Note: The Graphics built-ins have been designed to give you programmatic control over specific features and functionality. They may be used only when defining Graphics displays.


To trap the success or failure of PL/SQL program units, you can use exceptions. For example, you can display an error if a user performs an invalid action at runtime. There are two types of exceptions:

- user-defined
- pre-defined
User-defined Exceptions
In Graphics, you can declare, raise, and handle your own exceptions. For more information about exceptions, see the PL/SQL User’s Guide and Reference.

Pre-defined Exceptions
Many exceptions have been pre-defined for use within Graphics. To take action on a pre-defined exception, you need to write an exception handler for it. Exception handlers define statements that are executed when certain conditions or exceptions are raised in a PL/SQL program unit. When an exception is raised, the normal execution is stopped and control is passed to the appropriate exception handler.

For more information about writing exception handlers, see the PL/SQL User’s Guide and Reference.


Libraries
PL/SQL libraries are collections of PL/SQL program units that you can create in Graphics. Unlike individual program units that you can store with displays, PL/SQL libraries are stored in the database or file system and can be used by multiple users in multiple displays. To reference PL/SQL libraries in Graphics, you need to attach them to your displays, using the Object Navigator.

You can reference PL/SQL libraries multiple times within a single display or share them with other modules in an application. For more information about working with PL/SQL libraries, see “Using PL/SQL Libraries” in the “Working with PL/SQL Constructs” chapter of the Procedure Builder Developer’s Guide.

Same Name Resolution
If you make references to PL/SQL constructs with duplicate names, Graphics will search for them in the following order:

1. Among the following, there is no pre-defined order:
   - local program units saved with a display
   - PL/SQL STANDARD built-ins
   - Graphics built-ins
2. Attached libraries
3. Stored subprograms
In Graphics, you can declare, raise, and handle your own exceptions. For more information about exceptions, see the PL/SQL User’s Guide and Reference.

Many exceptions have been pre-defined for use within Graphics. To take action on a pre-defined exception, you need to write an exception handler for it. Exception handlers define statements that are executed when certain conditions or exceptions are raised in a PL/SQL program unit. When an exception is raised, the normal execution is stopped and control is passed to the appropriate exception handler. For more information about writing exception handlers, see the PL/SQL User’s Guide and Reference.


PL/SQL libraries are collections of PL/SQL program units that you can create in Graphics. Unlike individual program units that you can store with displays, PL/SQL libraries are stored in the database or file system and can be used by multiple users in multiple displays. To reference PL/SQL libraries in Graphics, you need to attach them to your displays, using the Object Navigator.

You can reference PL/SQL libraries multiple times within a single display or share them with other modules in an application. For more information about working with PL/SQL libraries, see “Using PL/SQL Libraries” in the “Working with PL/SQL Constructs” chapter of the Procedure Builder Developer’s Guide.

If you make references to PL/SQL constructs with duplicate names, Graphics will search for them in the following order:

1. Among the following, there is no pre-defined order:
   - local program units saved with a display
   - PL/SQL STANDARD built-ins
   - Graphics built-ins
2. Attached libraries
3. Stored subprograms
Specifying Coordinate and Size Information

Many of the built-in constructs provided refer to, or require you to provide, numeric values that specify a position on the layout, or dimensions of an object. The unit of measurement that you must use for these values is called a layout unit.

Layout units are used to provide a consistent measurement system for Graphics applications that may be run on several display devices, each with a different display resolution. For example, 1 inch on a device with a screen resolution of 72 lines/inch is different than on a device with a screen resolution of 90 lines/inch. In contrast, the Graphics layout resolution is the same on all systems. By specifying all of your measurements in layout units, you can be sure that they remain consistent across all devices.

OG_INCH Built-in Global

Graphics provides a global variable called OG_INCH that is equal to the number of layout units in one inch.

For example, to define a rectangular region that is three inches high and two inches wide, you could declare a local variable:

```plsql
my_rect   OG_RECTANGLE;
```

Then you could use the OG_INCH global variable, as in the following statements:

```plsql
my_rect.height := 3 * OG_INCH;
my_rect.width := 2 * OG_INCH;
```

We recommend that you do not determine and use the literal value this global variable in your applications. To ensure that your display works properly in the current and future version of Graphics, it is best for you to use the supported built-in global variable.

Referencing Graphics Objects

Many built-in subprograms allow you to perform operations on Graphics objects. For example, the OG_EXECUTE_QUERY procedure allows you to execute a query. These subprograms require that you specify which object you want to manipulate.

Using Object Handles

Instead of referring to an object by its name, many built-in procedures, functions, and datatypes require that a handle (i.e., pointer) to the object be used. For example, to execute a query, you must pass the query’s handle to OG_EXECUTE_QUERY:
Many of the built-in constructs provided refer to, or require you to provide, numeric values that specify a position on the layout, or dimensions of an object. The unit of measurement that you must use for these values is called a layout unit.

Layout units are used to provide a consistent measurement system for Graphics applications that may be run on several display devices, each with a different display resolution. For example, 1 inch on a device with a screen resolution of 72 lines/inch is different than on a device with a screen resolution of 90 lines/inch. In contrast, the Graphics layout resolution is the same on all systems. By specifying all of your measurements in layout units, you can be sure that they remain consistent across all devices.

Graphics provides a global variable called OG_INCH that is equal to the number of layout units in one inch. For example, to define a rectangular region that is three inches high and two inches wide, you could declare a local variable:

```sql
my_rect   OG_RECTANGLE;
```

Then you could use the OG_INCH global variable, as in the following statements:

```sql
my_rect.height := 3 * OG_INCH;
my_rect.width := 2 * OG_INCH;
```

We recommend that you do not determine and use the literal value this global variable in your applications. To ensure that your display works properly in the current and future version of Graphics, it is best for you to use the supported built-in global variable.

Referencing Graphics Objects

Many built-in subprograms allow you to perform operations on Graphics objects. For example, the OG_EXECUTE_QUERY procedure allows you to execute a query. These subprograms require that you specify which object you want to manipulate.

Instead of referring to an object by its name, many built-in procedures, functions, and datatypes require that a handle (i.e., pointer) to the object be used. For example, to execute a query, you must pass the query's handle to OG_EXECUTE_QUERY:
PROCEDURE getdata IS
    query_handle  OG_QUERY;
BEGIN
    query_handle := OG_GET_QUERY('query0');
    OG_EXECUTE_QUERY(query_handle);
END;

Graphics provides a set of built-in functions that you use to obtain an object's handle. Each of these functions returns a value of a specific datatype, depending on the type of object whose handle it retrieves.

The following table lists the Graphics objects whose handles you can get, the functions used to get them, and the datatype of the return value:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Function</th>
<th>Return Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>OG_GET_AXIS</td>
<td>OG_AXIS</td>
</tr>
<tr>
<td>Button Procedure</td>
<td>OG_GET_BUTTONPROC</td>
<td>OG_BUTTONPROC</td>
</tr>
<tr>
<td>Chart Template</td>
<td>OG_GET_TEMPLATE</td>
<td>OG_TEMPLATE</td>
</tr>
<tr>
<td>Display</td>
<td>OG_GET_DISPLAY</td>
<td>OG_DISPLAY</td>
</tr>
<tr>
<td>Field Template</td>
<td>OG_GET_FTEMP</td>
<td>OG_FTEMP</td>
</tr>
<tr>
<td>Graphic Object</td>
<td>OG_GET_OBJECT</td>
<td>OG_OBJECT</td>
</tr>
<tr>
<td>Layer</td>
<td>OG_GET_LAYER</td>
<td>OG_LAYER</td>
</tr>
<tr>
<td>Query</td>
<td>OG_GET_QUERY</td>
<td>OG_QUERY</td>
</tr>
<tr>
<td>Reference Line</td>
<td>OG_GET_REFLINE</td>
<td>OG_REFLINE</td>
</tr>
<tr>
<td>Sound</td>
<td>OG_GET_SOUND</td>
<td>OG_SOUND</td>
</tr>
<tr>
<td>Timer</td>
<td>OG_GET_TIMER</td>
<td>OG_TIMER</td>
</tr>
<tr>
<td>Window</td>
<td>OG_GET_WINDOW</td>
<td>OG_WINDOW</td>
</tr>
</tbody>
</table>

Handles are used instead of names to improve efficiency. Each time you refer to an item by name, Graphics must search through all of the existing items of that type to find the one whose name matches the one you specified. This approach requires a lot of time, and is used by the functions listed above to find the named item. However, once the item is found, a handle is created that points directly to that item. By using the handle, Graphics avoids having to search for the item’s name each time you reference it.

The best way to “store” handles is through the use of variables. You declare any variables you will used at the beginning of your program unit by giving them a name and a datatype.
PROCEDURE getdata IS
  query_handle  OG_QUERY;
BEGIN
  query_handle := OG_GET_QUERY('query0');
  OG_EXECUTE_QUERY(query_handle);
END;

Graphics provides a set of built-in functions that you use to obtain an object's handle. Each of these functions returns a value of a specific datatype, depending on the type of object whose handle it retrieves.

The following table lists the Graphics objects whose handles you can get, the functions used to get them, and the datatype of the return value:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Function</th>
<th>Return Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>OG_GET_AXIS</td>
<td>OG_AXIS</td>
</tr>
<tr>
<td>Button Proc</td>
<td>OG_GET_BUTTONPROC</td>
<td>OG_BUTTONPROC</td>
</tr>
<tr>
<td>Chart Template</td>
<td>OG_GET_TEMPLATE</td>
<td>OG_TEMPLATE</td>
</tr>
<tr>
<td>Display</td>
<td>OG_GET_DISPLAY</td>
<td>OG_DISPLAY</td>
</tr>
<tr>
<td>Field Template</td>
<td>OG_GET_FTEMP</td>
<td>OG_FTEMP</td>
</tr>
<tr>
<td>Graphic Object</td>
<td>OG_GET_OBJECT</td>
<td>OG_OBJECT</td>
</tr>
<tr>
<td>Layer</td>
<td>OG_GET_LAYER</td>
<td>OG_LAYER</td>
</tr>
<tr>
<td>Query</td>
<td>OG_GET_QUERY</td>
<td>OG_QUERY</td>
</tr>
<tr>
<td>Reference Line</td>
<td>OG_GET_REFLINE</td>
<td>OG_REFLINE</td>
</tr>
<tr>
<td>Sound</td>
<td>OG_GET_SOUND</td>
<td>OG_SOUND</td>
</tr>
<tr>
<td>Timer</td>
<td>OG_GET_TIMER</td>
<td>OG_TIMER</td>
</tr>
<tr>
<td>Window</td>
<td>OG_GET_WINDOW</td>
<td>OG_WINDOW</td>
</tr>
</tbody>
</table>

Handles are used instead of names to improve efficiency. Each time you refer to an item by name, Graphics must search through all of the existing items of that type to find the one whose name matches the one you specified. This approach requires a lot of time, and is used by the functions listed above to find the named item. However, once the item is found, a handle is created that points directly to that item. By using the handle, Graphics avoids having to search for the item's name each time you reference it.

The best way to "store" handles is through the use of variables. You declare any variables you will use at the beginning of your program unit by giving them a name and a datatype.
You can then assign the variable to the value that the function returns. Using the example above, the variable “query_handle” is declared as having the OG_QUERY built-in datatype. This is the datatype returned by the function OG_GET_QUERY.

The first line of the procedure assigns the query_handle variable to the value returned by the OG_GET_QUERY function. You can now use the variable query_handle anywhere within the program unit that requires the object handle for the query that is currently stored in the variable.

Building a Program Unit

Use the following steps to determine the content of your Graphics program units.

1. Determine what Graphics built-in procedures you need to use to perform an action.
2. Examine the procedures to determine what arguments they require. You can supply arguments to procedures in the form of handles returned by functions, arguments you supply to the program unit, or absolute values, depending on the procedure definition.
3. Determine what Graphics built-in functions you need (if any) to retrieve the required arguments.
4. Determine what variables you will need to define to store values returned by the functions.

Example

You want to create a program unit to activate a layer in the display. You’ll use the Graphics PL/SQL built-in procedure OG_ACTIVATE_LAYER, which requires an argument of the built-in datatype OG_LAYER. To get this datatype value, you’ll use the function OG_GET_LAYER, and declare a variable of the datatype OG_LAYER to store the value returned by OG_GET_LAYER.

Working backwards from the above steps, you would create the following procedure:

```
PROCEDURE act_layer IS
  /*Declare the variable to store the function return */
  the_layer  OG_LAYER;
BEGIN
  /*Use the function to assign the variable’s value*/
  the_layer:=OG_GET_LAYER('layer1');
  /*Use the variable value as an argument to the procedure */
  OG_ACTIVATE_LAYER(the_layer);
END;
```
You can then assign the variable to the value that the function returns. Using the example above, the variable "query_handle" is declared as having the OG_QUERY built-in datatype. This is the datatype returned by the function OG_GET_QUERY.

The first line of the procedure assigns the query_handle variable to the value returned by the OG_GET_QUERY function. You can now use the variable query_handle anywhere within the program unit that requires the object handle for the query that is currently stored in the variable.

Use the following steps to determine the content of your Graphics program units.

1. Determine what Graphics built-in procedures you need to use to perform an action.
2. Examine the procedures to determine what arguments they require. You can supply arguments to procedures in the form of handles returned by functions, arguments you supply to the program unit, or absolute values, depending on the procedure definition.
3. Determine what Graphics built-in functions you need (if any) to retrieve the required arguments.
4. Determine what variables you will need to define to store values returned by the functions.

You want to create a program unit to activate a layer in the display. You'll use the Graphics PL/SQL built-in procedure OG_ACTIVATE_LAYER, which requires an argument of the built-in datatype OG_LAYER. To get this datatype value, you'll use the function OG_GET_LAYER, and declare a variable of the datatype OG_LAYER to store the value returned by OG_GET_LAYER.

Working backwards from the above steps, you would create the following procedure:

```
PROCEDURE act_layer IS
  /*Declare the variable to store the function return */
  the_layer  OG_LAYER;
BEGIN
  /*Use the function to assign the variable's value*/
  the_layer:=OG_GET_LAYER('layer1');
  /*Use the variable value as an argument to the procedure */
  OG_ACTIVATE_LAYER(the_layer);
END;
```
Example

A second example involves exploding a pie slice. You want to create a program unit to explode the pie slice representing the highest salary. You’ll use a FOR loop to determine the highest salary value and use the built-in procedure OG_SET_EXPLOSION to explode the corresponding pie slice.

```plaintext
PROCEDURE explode IS
  pie   OG_OBJECT;
pieq  OG_QUERY;
mrows NUMBER;
msal  NUMBER;
csal  NUMBER;
rowsq NUMBER;
BEGIN
  -- Assign initial variable values
  msal := 0;
mrows := 0
  pie := OG_GET_OBJECT ('sal_chart');
pieq := OG_GET_QUERY('query0');
  -- where 'sal_chart' is the name of the chart and
  -- 'query0' is the name of the query.

  -- Execute the the chart query to get new data
  OG_EXECUTE_QUERY(pieq);

  -- Count the number of rows returned by the query
  rowsq := OG_NUMROWS(pieq, OG_NEWDATA);

  -- Determine maximum value by looping through query rows
  OG_START_FROM(pieq, OG_NEWDATA, 0);
  FOR i IN 0 .. rowsq - 1 LOOP
    csal := OG_GET_NUMCELL (pieq, OG_NEWDATA, 'sal');
    IF csal > msal THEN
      msal := csal;
mrows := i;
    END IF;
  OG_NEXT_ROW(pieq, OG_NEWDATA);
END LOOP;

  OG_SET_EXPLOSION(pie, OG_ALL_CHUPDA);-- explode the slice

  -- Update the chart
  OG_UPDATE_CHART(pie, OG_ALL_CHUPDA);
END;
```
A second example involves exploding a pie slice. You want to create a program unit to explode the pie slice representing the highest salary. You'll use a FOR loop to determine the highest salary value and use the built–in procedure OG_SET_EXPLOSION to explode the corresponding pie slice.

PROCEDURE explode IS
  pie   OG_OBJECT;
  pieq  OG_QUERY;
  mrows NUMBER;
  msal  NUMBER;
  csal  NUMBER;
  rowsq NUMBER;
BEGIN
  --Assign initial variable values
  msal := 0;
  mrows := 0
  pie := OG_GET_OBJECT ('sal_chart');
  pieq := OG_GET_QUERY('query0');
  --where 'sal_chart' is the name of the chart and
  --'query0' is the name of the query.
  --Execute the chart query to get new data
  OG_EXECUTE_QUERY(pieq);
  --Count the number of rows returned by the query
  rowsq := OG_NUMROWS(pieq, OG_NEWDATA);
  --Determine maximum value by looping through query rows
  OG_START_FROM(pieq, OG_NEWDATA, 0);
  FOR i IN 0 .. rowsq – 1 LOOP
    csal := OG_GET_NUMCELL (pieq, OG_NEWDATA, 'sal');
    IF csal > msal THEN
      msal := csal;
      mrows := i;
    END IF;
    OG_NEXT_ROW(pieq, OG_NEWDATA);
  END LOOP;
  OG_SET_EXPLOSION(pie, OG_ALL_CHUPDA); -- explode the slice
  --Update the chart
  OG_UPDATE_CHART(pie, OG_ALL_CHUPDA);
END;
The Interpreter is the central debugging workspace of Procedure Builder. It is a paned window that enables you to define, display, and debug your PL/SQL program units.
Debugging Displays

Graphics uses Procedure Builder features and functionality to allow you to debug the PL/SQL program units used in your displays. This chapter introduces how to use that functionality.

The following topics are covered in this chapter:

- the Procedure Builder interface
- testing your display
- dealing with unhandled exceptions
- setting debug actions
- correcting errors
- Procedure Builder packages

For complete information about using Procedure Builder to debug your PL/SQL program units, see the Procedure Builder Developer’s Guide.
Depending on the context in which Procedure Builder is invoked, the Interpreter can be presented in two ways:

- modal
- modeless

The modal Interpreter is presented automatically by Graphics while executing an application (e.g., during runtime simulation of a display, Graphics encounters a breakpoint you have set in a program unit). When the modal Interpreter is opened, execution of both the application and its PL/SQL is suspended until you close the modal Interpreter window.
Depending on the context in which Procedure Builder is invoked, the Interpreter can be presented in two ways:

- Modal
- Modeless

The modal Interpreter is presented automatically by Graphics while executing an application (e.g., during runtime simulation of a display, Graphics encounters a breakpoint you have set in a program unit). When the modal Interpreter is opened, execution of both the application and its PL/SQL is suspended until you close the modal Interpreter window.
The modeless Interpreter is presented when you manually invoke Procedure Builder by choosing Tools—>PL/SQL Interpreter from the menu.

The Interpreter has four main components:

- **menu bar** (modeless only) Shows the menu commands to perform most tasks.
- **control bar** Contains buttons you can use as accelerators for most common tasks.
- **Source pane** Allows you to show a program unit’s source and define debug actions.
- **Navigator pane** Shows the object hierarchy for the current session (debugging nodes only in the modal Interpreter).
  
  In the modeless Interpreter, the Navigator can be hidden.
- **Interpreter pane** Provides a command line where you can enter any PL/SQL constructs and Procedure Builder commands.

Both types of Interpreters have identical functions. However, since menus are not accessible in modal windows, the modal Interpreter contains additional buttons for accessing functions that are available via menus in the modeless Interpreter.

Pop-up menus containing many common commands are available in the different panes.

For a more detailed description of the Interpreter, see the “Working with the Interpreter” chapter in the Procedure Builder Developer’s Guide.

You can invoke the modeless Interpreter at any time, either in the Designer, or during runtime simulation. To open the modeless Interpreter within Graphics:

- Choose Tools—>PL/SQL Interpreter.

The modeless Interpreter appears. By default, the modeless Interpreter does not contain the Navigator pane. Any debugging actions you take in the modeless Interpreter can be accessed through objects in the Graphics Object Navigator.

Alternatively, you can insert the Navigator pane at any time by choosing View—>Navigator Pane from the menu.
The modeless Interpreter is presented when you manually invoke Procedure Builder by choosing **Tools** —> **PL/SQL Interpreter** from the menu.

The Interpreter has four main components:

1. (modeless only) Shows the menu commands to perform most tasks.
2. Contains buttons you can use as accelerators for most common tasks.
3. Allows you to show a program unit's source and define debug actions.
4. Shows the object hierarchy for the current session (debugging nodes only in the modal Interpreter).

In the modeless Interpreter, the Navigator can be hidden.

Provides a command line where you can enter any PL/SQL constructs and Procedure Builder commands.

Both types of Interpreters have identical functions. However, since menus are not accessible in modal windows, the modal Interpreter contains additional buttons for accessing functions that are available via menus in the modeless Interpreter.

Pop-up menus containing many common commands are available in the different panes.

For a more detailed description of the Interpreter, see the "Working with the Interpreter" chapter in the **Procedure Builder Developer's Guide**.

You can invoke the modeless Interpreter at any time, either in the Designer, or during runtime simulation. To open the modeless Interpreter within Graphics, choose **Tools** —> **PL/SQL Interpreter**.

The modeless Interpreter appears. By default, the modeless Interpreter does not contain the Navigator pane. Any debugging actions you take in the modeless Interpreter can be accessed through objects in the Graphics Object Navigator.

Alternatively, you can insert the Navigator pane at any time by choosing **View** —> **Navigator Pane** from the menu.
Using the Interpreter

You can perform Procedure Builder operations in two ways:

- using menus, editors, and command dialog boxes
- typing commands on the Interpreter command line

For example, to show the source of the procedure `my_proc` in the Source pane, you could click on the program unit's entry in the Object Navigator.

Alternatively, you can enter the following command at the `PL/SQL>` prompt in the Interpreter pane.

```
PL/SQL> .list proc my_proc
```

While entering commands on the command line, you can use the New Prompt command from the pop-up menu to abandon the command and show a new `PL/SQL>` prompt in the Interpreter pane.

You can record all Interpreter input and output to a log file if you wish to retain a record of your debugging session. For more information, see the “Logging Interpreter Input and Output” section below.

Testing Your Display in Runtime Simulation

Before you generate your displays, it is wise to test them to ensure that they operate correctly.

The simplest way to test your display is to use the Graphics runtime simulation mode (also called the debugger). From the Designer, you can run your display and test it in a separate debugging window. This allows you to quickly test and debug your display, and return to the Designer to perform any necessary fixes.

Using Runtime Simulation

To test your display in runtime simulation:

1. Click on the Run button in either the Object Navigator or the Layout editor, or choose File→Run to run the active display.

   If multiple displays are open, select the display object in the Navigator for the display you wish to run, then click on the Run button.

   If the display has not recently been saved, Graphics creates a temporary file of the current state of the display and runs that. If you have not saved the display at all, you will be prompted to do so before Graphics runs it.
Using the Interpreter

Using Runtime Simulation

14 – 4 Graphics Developer's Guide

You can perform Procedure Builder operations in two ways:
• using menus, editors, and command dialog boxes
• typing commands on the Interpreter command line

For example, to show the source of the procedure my_proc in the Source pane, you could click on the program unit's entry in the Object Navigator.

Alternatively, you can enter the following command at the PL/SQL prompt in the Interpreter pane.

PL/SQL> .list proc my_proc

While entering commands on the command line, you can use the New Prompt command from the pop-up menu to abandon the command and show a new PL/SQL prompt in the Interpreter pane.

You can record all Interpreter input and output to a log file if you wish to retain a record of your debugging session. For more information, see the "Logging Interpreter Input and Output" section below.

Testing Your Display in Runtime Simulation

Before you generate your displays, it is wise to test them to ensure that they operate correctly.

The simplest way to test your display is to use the Graphics runtime simulation mode (also called the debugger). From the Designer, you can run your display and test it in a separate debugging window. This allows you to quickly test and debug your display, and return to the Designer to perform any necessary fixes.

To test your display in runtime simulation:
1. Click on the Run button in either the Object Navigator or the Layout editor, or choose File —> Run to run the active display.
2. If multiple displays are open, select the display object in the Navigator for the display you wish to run, then click on the Run button.
3. If the display has not recently been saved, Graphics creates a temporary file of the current state of the display and runs that. If you have not saved the display at all, you will be prompted to do so before Graphics runs it.
If no displays are currently open in the Designer, you will be prompted to select one.

The runtime version of the selected display is opened in a separate Graphics Debugger window.

**Note:** When you test a display in runtime simulation, Graphics minimizes all other currently open windows except for the Object Navigator.

2. Test any drill-down relationships or button procedures to make sure they behave correctly.

   If you have any active timers, make sure they are operating and producing the correct results.

If your display works in runtime simulation, you can now safely generate your display. However, if you make any other changes to your display before generating it, it is a good idea to run the display again to test your changes.

If you encountered any errors during Runtime Simulation, see the sections below for ideas on how to solve them.

**Dealing with Unhandled Exceptions**

If Graphics encounter any errors during runtime simulation, it provides an error message alert with an error message number. Make sure you note the error message number and under what conditions it occurred. Was it upon opening the display? After you attempted to drill-down on a chart or after you clicked on a button?

Graphics error messages are documented in the “Error Messages” appendix of the *Graphics Reference Manual*.

An error message OG–00637 indicates that an unhandled PL/SQL exception has been encountered. Note the exception number listed in the error message as this will determine where you can track down the exception error text that explains the possible cause of the error.

Graphics provides a number of built-in PL/SQL exceptions for errors encountered with any Graphics built-in procedures, functions, or datatypes.

For more information about Graphics built-in exceptions, see the “PL/SQL Built-in Exceptions” chapter of the *Graphics Reference Manual*. 
If no displays are currently open in the Designer, you will be prompted to select one. The runtime version of the selected display is opened in a separate Graphics Debugger window.

**Note:** When you test a display in runtime simulation, Graphics minimizes all other currently open windows except for the Object Navigator.

2. Test any drill-down relationships or button procedures to make sure they behave correctly. If you have any active timers, make sure they are operating and producing the correct results.

If your display works in runtime simulation, you can now safely generate your display. However, if you make any other changes to your display before generating it, it is a good idea to run the display again to test your changes.

If you encountered any errors during Runtime Simulation, see the sections below for ideas on how to solve them.

**Dealing with Unhandled Exceptions**

If Graphics encounter any errors during runtime simulation, it provides an error message alert with an error message number. Make sure you note the error message number and under what conditions it occurred. Was it upon opening the display? After you attempted to drill-down on a chart or after you clicked on a button?


An error message OG–00637 indicates that an unhandled PL/SQL exception has been encountered. Note the exception number listed in the error message as this will determine where you can track down the exception error text that explains the possible cause of the error.

Graphics provides a number of built-in PL/SQL exceptions for errors encountered with any Graphics built-in procedures, functions, or datatypes.

**Tracing Errors**

Based on the error and the conditions under which it occurred, you can usually track down the source of the error. If you have nested program units, use the first program unit in the sequence to trace the error down to the source.

For example, you have a button procedure A that calls program unit B that calls program unit C. An error is encountered during execution of one of these program units and a PL/SQL exception is raised. In this case, you would select program unit A to trace the source of the error.

**Viewing Errors in the Debugger**

To view a program unit’s errors in the debugger:

1. Expand the `display_name<debug>` node in the Object Navigator.
2. Double-click on the program unit to show the program unit in the Program Unit editor.
   
   If you are searching for an error in a sequence of nested program units, open the first program unit in the sequence. For example, if the error occurred as a result of clicking on a button, open the associated button procedure.

   The compilation errors associated with that program unit are shown in the pane below the program unit source. If you are dealing with nested program units, the last program unit to be called before the error was encountered appears at the top of the error stack.

   If the cause of the error is not immediately apparent, continue to the next section. If the cause of the error is apparent, proceed to the section “Correcting Errors” on page 14 – 12.

**Setting Debug Actions**

If it is not immediately apparent how to fix an error, you may wish to set a debug action to help you locate the source of the problem. For example, the offending program unit may contain calls to subprograms, loops, or other control expressions that make it difficult to locate the exact source of the problem.

Debug actions enable you to monitor and/or interrupt the execution of PL/SQL program units. There are two types of debug actions:

- **breakpoints**
- **debug triggers**
Based on the error and the conditions under which it occurred, you can usually track down the source of the error. If you have nested program units, use the first program unit in the sequence to trace the error down to the source.

For example, you have a button procedure A that calls program unit B that calls program unit C. An error is encountered during execution of one of these program units and a PL/SQL exception is raised. In this case, you would select program unit A to trace the source of the error.

To view a program unit's errors in the debugger:
1. Expand the display_name<debug> node in the Object Navigator.
2. Double-click on the program unit to show the program unit in the Program Unit editor.

If you are searching for an error in a sequence of nested program units, open the first program unit in the sequence. For example, if the error occurred as a result of clicking on a button, open the associated button procedure.

The compilation errors associated with that program unit are shown in the pane below the program unit source. If you are dealing with nested program units, the last program unit to be called before the error was encountered appears at the top of the error stack.

If the cause of the error is not immediately apparent, continue to the next section. If the cause of the error is apparent, proceed to the section "Correcting Errors" on page 14–12.

Setting Debug Actions

If it is not immediately apparent how to fix an error, you may wish to set a debug action to help you locate the source of the problem. For example, the offending program unit may contain calls to subprograms, loops, or other control expressions that make it difficult to locate the exact source of the problem.

Debug actions enable you to monitor and/or interrupt the execution of PL/SQL program units. There are two types of debug actions:

• breakpoints
• debug triggers
In addition, you can use the DEBUG package to specify debugging tasks such as setting a breakpoint, suspending execution of the program unit and returning control to the Interpreter, retrieving and setting variable and parameter values, or interpreting PL/SQL commands.

For more information about the DEBUG package, see the “Procedure Builder Packages” chapter of the Procedure Builder Developer’s Guide.

**Where to Set Debug Actions**

You can set debug actions either in the Designer or in the debugger. Where you decide to set the debug actions depends on the complexity of your program units and the display itself.

**Setting Debug Actions in the Designer**

If you set debug actions in the Designer, and then run your display, the modal Interpreter will automatically appear when a debug action such as a debug trigger is reached. When the modal Interpreter appears, execution of your display and its PL/SQL is suspended until you close the Interpreter.

Some considerations for setting debug actions in the Designer include the following:

- you want to set a debug action on an Open Display trigger that is fired as soon as the display is run
- you want to retain the debug action for future use

If you set a debug action in the Designer and then make modifications to the program unit and recompile it, you may need to modify the debug action to change its source location. Debug actions are associated with line numbers in the program unit source.

**Setting Debug Actions in the Debugger**

Setting debug actions in the debugger allows you to debug on-the-fly, in the modeless Interpreter.

Once you close the display in the debugger, all debug actions you have set in program units will be lost. If your debug actions consist only of breakpoints and simple debug triggers, this may not be a consideration to you.

If you debug your program units directly in the debugger, your changes are not reflected in the Designer.

**Using Breakpoints**

The most common type of debug action is the breakpoint. Breakpoints are handy for incrementally monitoring execution of program units.

Graphics suspends execution of the program unit just before reaching the line on which the breakpoint is specified. At this point, you can use Procedure Builder’s features to inspect and/or modify the program state (i.e. variable values).
In addition, you can use the DEBUG package to specify debugging tasks such as setting a breakpoint, suspending execution of the program unit and returning control to the Interpreter, retrieving and setting variable and parameter values, or interpreting PL/SQL commands. For more information about the DEBUG package, see the "Procedure Builder Packages" chapter of the Procedure Builder Developer's Guide.

You can set debug actions either in the Designer or in the debugger. Where you decide to set the debug actions depends on the complexity of your program units and the display itself.

If you set debug actions in the Designer, and then run your display, the modal Interpreter will automatically appear when a debug action such as a debug trigger is reached. When the modal Interpreter appears, execution of your display and its PL/SQL is suspended until you close the Interpreter.

Some considerations for setting debug actions in the Designer include the following:

- you want to set a debug action on an Open Display trigger that is fired as soon as the display is run
- you want to retain the debug action for future use

If you set a debug action in the Designer and then make modifications to the program unit and recompile it, you may need to modify the debug action to change its source location. Debug actions are associated with line numbers in the program unit source.

Setting debug actions in the debugger allows you to debug on-the-fly, in the modeless Interpreter. Once you close the display in the debugger, all debug actions you have set in program units will be lost. If your debug actions consist only of breakpoints and simple debug triggers, this may not be a consideration to you.

If you debug your program units directly in the debugger, your changes are not reflected in the Designer.

The most common type of debug action is the breakpoint. Breakpoints are handy for incrementally monitoring execution of program units. Graphics suspends execution of the program unit just before reaching the line on which the breakpoint is specified. At this point, you can use Procedure Builder's features to inspect and/or modify the program state (i.e. variable values).
Once satisfied, you can resume execution with the GO or STEP commands, or you can abort execution using the RESET command.

For example, suppose you have a procedure that contains a counting loop. The loop calls a function and increments the value of a local variable. To ensure that the variable is assigned the correct value for each cycle of the loop, you could place a breakpoint on the line directly after the variable assignment. When the breakpoint is reached and the procedure is interrupted, you can examine the call stack trace in the Navigator to check the variable value, then use the STEP command to cycle to the next breakpoint.

Setting Breakpoints

Setting a breakpoint suspends execution at a specific source line of a program unit, passing control to the Interpreter:

To set a breakpoint in a program unit:

1. Choose Tools—>PL/SQL Interpreter to show the PL/SQL Interpreter.
2. In the Navigator, select the program unit you wish to debug. The program unit appears in the Interpreter’s Source pane. Each line is numbered.
3. Double-click on the line on which you want to set the breakpoint. The breakpoint is inserted.

You can also define a “break trigger” for a breakpoint. A break trigger is a PL/SQL block that executes each time the breakpoint is reached. This block might contain an Procedure Builder command that should be performed each time the breakpoint is reached.

You can use the BREAK command in the Interpreter pane to insert a breakpoint at a specified source line. For more information, see the BREAK command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

Using Debug Triggers

Debug triggers are a more general form of debug action. A debug trigger associates a block of PL/SQL code with a specific source line within a program unit.

Graphics executes a debug trigger just before reaching the line on which the debug trigger is specified. You can assign a debug trigger to fire at any of the following locations:

- upon reaching a single line in a program unit (e.g., the current source location, line 5, line 23, etc.)
Once satisfied, you can resume execution with the GO or STEP commands, or you can abort execution using the RESET command. For example, suppose you have a procedure that contains a counting loop. The loop calls a function and increments the value of a local variable. To ensure that the variable is assigned the correct value for each cycle of the loop, you could place a breakpoint on the line directly after the variable assignment. When the breakpoint is reached and the procedure is interrupted, you can examine the call stack trace in the Navigator to check the variable value, then use the STEP command to cycle to the next breakpoint.

Setting a breakpoint suspends execution at a specific source line of a program unit, passing control to the Interpreter:

To set a breakpoint in a program unit:

1. Choose "Tools" —> "PL/SQL Interpreter" to show the PL/SQL Interpreter.
2. In the Navigator, select the program unit you wish to debug. The program unit appears in the Interpreter's Source pane. Each line is numbered.
3. Double-click on the line on which you want to set the breakpoint. The breakpoint is inserted.

You can also define a "break trigger" for a breakpoint. A break trigger is a PL/SQL block that executes each time the breakpoint is reached. This block might contain a Procedure Builder command that should be performed each time the breakpoint is reached.

You can use the BREAK command in the Interpreter pane to insert a breakpoint at a specified source line. For more information, see the BREAK command section in the "Command Reference" chapter of the Procedure Builder Developer's Guide.

Debug triggers are a more general form of debug action. A debug trigger associates a block of PL/SQL code with a specific source line within a program unit. Graphics executes a debug trigger just before reaching the line on which the debug trigger is specified. You can assign a debug trigger to fire at any of the following locations:

- upon reaching a single line in a program unit (e.g., the current source location, line 5, line 23, etc.)
Setting Debug Triggers

- every time the Interpreter takes control (i.e., whenever it suspends program execution due to a breakpoint, program stepping, etc.)
- every PL/SQL source line being executed

Debug triggers are especially handy for setting conditional breakpoints. You can raise the exception DEBUG.BREAK from within the trigger body. For example, the debug trigger shown below establishes a conditional breakpoint on line 10 of my_proc, which will be reached only if the local NUMBER variable my_sal exceeds 5000:

```plsql
PL/SQL> .trigger proc my_proc line 10 is
    > IF DEBUG.GETN('my_sal') > 5000 THEN
    >   RAISE DEBUG.BREAK;
    > END IF;
```

For more information about the DEBUG package, and other built-in packages provided with Procedure Builder, see the “Procedure Builder Packages” chapter of the Procedure Builder Developer's Guide.

A brief description of each of the packages provided with Procedure Builder appears at the end of this chapter on page 14–13.

Setting Debug Triggers

Set a debug trigger when you want to attach a condition to a debug action. For example, you may want to insert a breakpoint only if a variable is assigned a certain value.

To set a debug trigger:

1. Choose Tools—>PL/SQL Interpreter to show the PL/SQL Interpreter.
2. In the Navigator, select the program unit you wish to debug.
   The program unit appears in the Interpreter’s Source pane. Each line is numbered.
3. Select the line where you wish to place the debug trigger.
4. Choose Trigger from the pop-up menu, or choose Debug—>Trigger to show the PL/SQL Trigger dialog.
5. Type the body of the trigger in the Trigger Body field.
6. Choose the OK button to accept the trigger and close the dialog.

For more information about creating debug triggers, see the “Debugging PL/SQL Program Units” chapter in the Procedure Builder Developer's Guide.
Setting Debug Triggers

14 – 9 Debugging Displays

• every time the Interpreter takes control (i.e., whenever it suspends program execution due to a breakpoint, program stepping, etc.)
• every PL/SQL source line being executed

Debug triggers are especially handy for setting conditional breakpoints. You can raise the exception DEBUG.BREAK from within the trigger body. For example, the debug trigger shown below establishes a conditional breakpoint on line 10 of my_proc, which will be reached only if the local NUMBER variable my_sal exceeds 5000:

PL/SQL> .trigger proc my_proc line 10 is
   +> IF DEBUG.GETN('my_sal') > 5000 THEN
   +>   RAISE DEBUG.BREAK;
   +> END IF;

For more information about the DEBUG package, and other built-in packages provided with Procedure Builder, see the "Procedure Builder Packages" chapter of the Procedure Builder Developer's Guide.

A brief description of each of the packages provided with Procedure Builder appears at the end of this chapter on page 14 – 13.

Set a debug trigger when you want to attach a condition to a debug action. For example, you may want to insert a breakpoint only if a variable is assigned a certain value.

To set a debug trigger:
1. Choose Tools —> PL/SQL Interpreter to show the PL/SQL Interpreter.
2. In the Navigator, select the program unit you wish to debug. The program unit appears in the Interpreter's Source pane. Each line is numbered.
3. Select the line where you wish to place the debug trigger.
4. Choose Trigger from the pop-up menu, or choose Debug —> Trigger to show the PL/SQL Trigger dialog.
5. Type the body of the trigger in the Trigger Body field.
6. Choose the OK button to accept the trigger and close the dialog.

For more information about creating debug triggers, see the "Debugging PL/SQL Program Units" chapter in the Procedure Builder Developer's Guide.
Using Debug Levels

When a debug action interrupts program execution, the Interpreter takes control and establishes what is known as a debug level. At a debug level, you can enter commands and PL/SQL statements to inspect and modify the state of the interrupted program unit, as well as resume execution.

Since any PL/SQL code interactively entered at a debug level may itself be interrupted (for example, by encountering another breakpoint), it is possible for debug levels to nest. To facilitate distinguishing one debug level from another, the levels are numbered. The most deeply nested level is assigned the highest number. Numbering starts at zero with the outermost level.

The 0th or outermost level is commonly referred to as top level. The top level has no associated program state since it is the outermost level at which program units are originally invoked. When code invoked from the top level is interrupted, debug level 1 is established. Similarly, interrupting code invoked from debug level 1 establishes debug level 2, and so on.

The Interpreter command prompt reflects the current debug level. When the Interpreter enters levels below the top level, the prompt includes a prefix containing the current debug level number. For example, the Interpreter command prompt at debug level 1 appears as shown below:

(debug 1)PL/SQL>

You can use the RESET command to return control to an outer debug level without continuing execution in the current debug level. See the section “Cancelling Program Unit Execution” below for details.

Controlling Program Unit Execution

Once you have inspected and/or modified the program state following a debug action, you can control execution of the program unit as follows:

- stepping through the program unit
- resuming execution
- cancelling program unit execution

Stepping Through the Program Unit

You can use the STEP command to incrementally step through your interrupted program unit. This command provides four different . Control returns to the Interpreter after the specified set of statements have been executed. There are four variations of the STEP command for temporarily resuming execution of an interrupted program.

You can invoke the STEP command either by choosing one of the buttons from the control bar, or by entering it on the Interpreter command line.
When a debug action interrupts program execution, the Interpreter takes control and establishes what is known as a debug level. At a debug level, you can enter commands and PL/SQL statements to inspect and modify the state of the interrupted program unit, as well as resume execution. Since any PL/SQL code interactively entered at a debug level may itself be interrupted (for example, by encountering another breakpoint), it is possible for debug levels to nest. To facilitate distinguishing one debug level from another, the levels are numbered. The most deeply nested level is assigned the highest number. Numbering starts at zero with the outermost level.

The 0th or outermost level is commonly referred to as top level. The top level has no associated program state since it is the outermost level at which program units are originally invoked. When code invoked from the top level is interrupted, debug level 1 is established. Similarly, interrupting code invoked from debug level 1 establishes debug level 2, and so on.

The Interpreter command prompt reflects the current debug level. When the Interpreter enters levels below the top level, the prompt includes a prefix containing the current debug level number. For example, the Interpreter command prompt at debug level 1 appears as shown below:

(debug 1)PL/SQL>

You can use the RESET command to return control to an outer debug level without continuing execution in the current debug level. See the section “Cancelling Program Unit Execution” below for details.

Once you have inspected and/or modified the program state following a debug action, you can control execution of the program unit as follows:

- stepping through the program unit
- resuming execution
- cancelling program unit execution

You can use the STEP command to incrementally step through your interrupted program unit. This command provides four different variations. Control returns to the Interpreter after the specified set of statements have been executed. There are four variations of the STEP command for temporarily resuming execution of an interrupted program.

You can invoke the STEP command either by choosing one of the buttons from the control bar, or by entering it on the Interpreter command line.
Resuming Execution
You can use the GO command to resume execution of a program indefinitely—that is, until either the currently executing thread terminates or is interrupted again due to a debug action. You can invoke the GO command either by choosing the Go button from the control bar or by entering it on the Interpreter command line.

For more information, see the GO command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

Cancelling Program Unit Execution
You can use the RESET command to return control to an outer debug level without continuing execution in the current debug level. Thus, RESET effectively aborts execution at the current (and possibly higher) debug levels.

You can invoke the RESET command either by choosing the Reset button in the control bar, or by entering it on the command line in the Interpreter.

For more information, see the RESET command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

Logging Interpreter Input and Output
Procedure Builder enables you to keep a record of Interpreter input and output when you are debugging program units. After you enable logging, commands typed in the Interpreter—and any resulting Interpreter output messages—are appended or written to a log file until you disable or terminate logging.

Enabling a Log File
To enable Interpreter logging:

1. Choose Debug—>Log to show the Log Interpreter dialog.
2. Type a directory path and log file name in the File field, or choose the Find button to open your operating system file dialog where you can select an existing log file.
3. If you wish to append the current log to an existing log file, select the Append check box.
4. Choose OK to accept your changes and close the dialog.

Disabling a Log File
To disable logging to the current log file:

1. Choose Debug—>Log to show the Log Interpreter dialog.
2. Select the Disabled check box.
3. Choose OK to accept your changes and close the dialog.

For more information, see the LOG command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.
Resuming Execution

Cancelling Program Unit Execution

Logging Interpreter Input and Output

Enabling a Log File

Disabling a Log File

14 – 11 Debugging Displays

For more information, see the STEP command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

You can use the GO command to resume execution of a program indefinitely—that is, until either the currently executing thread terminates or is interrupted again due to a debug action. You can invoke the GO command either by choosing the Go button from the control bar or by entering it on the Interpreter command line. For more information, see the GO command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

You can use the RESET command to return control to an outer debug level without continuing execution in the current debug level. Thus, RESET effectively aborts execution at the current (and possibly higher) debug levels.

You can invoke the RESET command either by choosing the Reset button in the control bar, or by entering it on the command line in the Interpreter. For more information, see the RESET command section in the “Command Reference” chapter of the Procedure Builder Developer’s Guide.

Procedure Builder enables you to keep a record of Interpreter input and output when you are debugging program units. After you enable logging, commands typed in the Interpreter—and any resulting Interpreter output messages—are appended or written to a log file until you disable or terminate logging.

To enable Interpreter logging:

1. Choose Debug —> Log to show the Log Interpreter dialog.

2. Type a directory path and log file name in the File field, or choose the Find button to open your operating system file dialog where you can select an existing log file.

3. If you wish to append the current log to an existing log file, select the Append check box.

4. Choose OK to accept your changes and close the dialog.

To disable logging to the current log file:

1. Choose Debug —> Log to show the Log Interpreter dialog.

2. Select the Disabled check box.

3. Choose OK to accept your changes and close the dialog.
Viewing the Log File

To view the contents of an existing log file:

- Choose **Debug**—>**Interpreter Log** to show the Interpreter Log browser.

OR

1. Choose **Debug**—>**Log** to show the Log Interpreter dialog.
2. Select the Show Log check box.
3. Choose OK to accept your changes and close the dialog.

The Interpreter Log browser appears.

Command Line Options

You can control logging by entering one of the following commands in the Interpreter:

- LOG
- DISABLE (Logging)
- ENABLE (Logging)

For more information on these and other Procedure Builder commands, see “Command Reference” chapter in the *Procedure Builder Developer’s Guide*.

Correcting Errors

You can correct errors either in the debugger, or in the Designer. If you correct and recompile program units in the debugger, your changes will not automatically be reflected in the Designer. If you make multiple changes to program units in the debugger, you should synchronize the program units in the debugger with those in the Designer.

If you make your changes in the Designer, remember to save and run your display again to make sure your changes have fixed the error.

Synchronizing Program Units

If you corrected your PL/SQL errors in the debugger, you should synchronize the contents of the corrected program units in the debugger with the contents of the program units in the Designer. The synchronize operation replaces the program units in the Designer with those from the debugger.

- Choose **Edit**—>**Synchronize Program Units** from the menu.
Viewing the Log File

Command Line Options

Synchronizing Program Units

1. Choose **Debug** —> **Log** to show the Log Interpreter dialog.
2. Select the Show Log check box.
3. Choose OK to accept your changes and close the dialog.

The Interpreter Log browser appears.

You can control logging by entering one of the following commands in the Interpreter:

- `LOG`
- `DISABLE (Logging)`
- `ENABLE (Logging)`

For more information on these and other Procedure Builder commands, see "Command Reference" chapter in the Procedure Builder Developer's Guide.

Correcting Errors

You can correct errors either in the debugger, or in the Designer. If you correct and recompile program units in the debugger, your changes will not automatically be reflected in the Designer. If you make multiple changes to program units in the debugger, you should synchronize the program units in the debugger with those in the Designer.

If you make your changes in the Designer, remember to save and run your display again to make sure your changes have fixed the error.

If you corrected your PL/SQL errors in the debugger, you should synchronize the contents of the corrected program units in the debugger with the contents of the program units in the Designer. The synchronize operation replaces the program units in the Designer with those from the debugger.

Choose **Edit** —> **Synchronize Program Units** from the menu.
Procedure Builder Packages

In addition to the Graphics built-in PL/SQL datatypes, functions, and procedures, Procedure Builder provides the following built-in PL/SQL packages that contain many PL/SQL constructs you can reference while debugging your program units:

- **DDE**: Provides Dynamic Data Exchange support within CDE2 tools.
- **DEBUG**: Provides procedures, functions, and exceptions for debugging PL/SQL program units.
- **LIST**: Provides procedures, functions, and exceptions you can use to create and maintain lists of character strings (VARCHAR2). This provides a means of creating arrays in PL/SQL Version 1.
- **OLE2**: Provides a PL/SQL API for creating, manipulating and accessing attributes of OLE2 Automation Objects.
- **ORA_FFI**: Provides a public interface for calling out to foreign(C) functions from PL/SQL.
- **ORA_NLS**: Enables you to extract high-level information about your current language environment.
- **ORA_PROF**: Provides procedures, functions, and exceptions you can use for tuning your PL/SQL program units (i.e. examining how much time a specific piece of code takes to run).
- **TEXT_IO**: Provides constructs that allow you to read and write information from and to files.
- **TOOL_ENV**: Allows you to interact with Oracle environment variables.
- **TOOL_ERR**: Allows you to access and manipulate the error stack created by other built-in packages such as DEBUG.
- **TOOL_RES**: Provides you with a means of extracting string resources from a resource file with the goal of making PL/SQL code more portable by isolating all textual data in the resource file.

For more information about these packages and their contents, see the “Procedure Builder Packages” chapter of the *Procedure Builder Developer’s Guide*. 
In addition to the Graphics built-in PL/SQL datatypes, functions, and procedures, Procedure Builder provides the following built-in PL/SQL packages that contain many PL/SQL constructs you can reference while debugging your program units:

- Provides Dynamic Data Exchange support within CDE2 tools.
- Provides procedures, functions, and exceptions for debugging PL/SQL program units.
- Provides procedures, functions, and exceptions you can use to create and maintain lists of character strings (VARCHAR2). This provides a means of creating arrays in PL/SQL Version 1.
- Provides a PL/SQL API for creating, manipulating and accessing attributes of OLE2 Automation Objects.
- Provides a public interface for calling out to foreign (C) functions from PL/SQL.
- Enables you to extract high-level information about your current language environment.
- Provides procedures, functions, and exceptions you can use for tuning your PL/SQL program units (i.e. examining how much time a specific piece of code takes to run).
- Provides constructs that allow you to read and write information from and to files.
- Allows you to interact with Oracle environment variables.
- Allows you to access and manipulate the error stack created by other built-in packages such as DEBUG.
- Provides you with a means of extracting string resources from a resource file with the goal of making PL/SQL code more portable by isolating all textual data in the resource file.

For more information about these packages and their contents, see the "Procedure Builder Packages" chapter of the Procedure Builder Developer's Guide.
CHAPTER 15

Runtime Issues

This chapter describes how to prepare your displays to be used with Graphics Runtime.

The following topics are covered in this chapter:

• preparing a display for runtime – 15 – 1
• chaining displays – 15 – 3
• sharing information among displays – 15 – 4
• managing windows – 15 – 6

Preparing a Display for Runtime

Sizing the Layout

You can specify a size for your display in the Layout Settings dialog.

1. Choose View—>View Options—>Layout from the menu.
2. Change the default Horizontal and Vertical size in the Layout Size field.
3. Choose either the Inches or Centimeters radio buttons.
4. Choose the OK button to accept the Layout size settings and close the dialog.

Your display layout is reduced to the specified size. You can use these size settings to determine the size of the window the display appears in.
CHAPTER 15

Sizing the Layout

This chapter describes how to prepare your displays to be used with Graphics Runtime.

The following topics are covered in this chapter:

- preparing a display for runtime – 15 – 1
- chaining displays – 15 – 3
- sharing information among displays – 15 – 4
- managing windows – 15 – 6

Preparing a Display for Runtime

You can specify a size for your display in the Layout Settings dialog.

1. Choose View —> View Options —> Layout from the menu.

2. Change the default Horizontal and Vertical size in the Layout Size field.

3. Choose either the Inches or Centimeters radio buttons.

4. Choose the OK button to accept the Layout size settings and close the dialog.

Your display layout is reduced to the specified size. You can use these size settings to determine the size of the window the display appears in.
Setting Window Properties for the Default Window

By default, Graphics opens each display in a window named Main Layout. You can change the following properties of the default window:

- position in the Runtime layout
- horizontal and vertical size of the window
- window name

If you want to set properties for the default window, you should do so in an Open Display trigger such as the following:

```plaintext
PROCEDURE OGTRIGGERPROC0 IS
  the_win OG_WINDOW;
  win_pos OG_POINT;
  win_wid NUMBER;
  win_ht NUMBER;
BEGIN
  the_win:=OG_GET_WINDOW('Main Layout');

  /*Define the position for the window*/
  win_pos.x:=1*OG_INCH;
  win_pos.y:=.5*OG_INCH;

  /*Set the position property for the window*/
  OG_SET_POSITION(the_win, win_pos);

  /*Define the size of the window */
  win_wid:=4*OG_APP.HSCREEN_RES;
  win_ht:=5*OG_APP.VSCREEN_RES;

  /*Set the size property of the window*/
  OG_SET_WINDOW_SIZE(the_win, win_wid, win_ht);

  /*Assign a name to the window */
  OG_SET_WINDOW_NAME(the_win, 'Start Here');
END;
```

When the display is opened in Graphics Runtime, the display will be opened in a 4x5 window with the title ‘Start Here’ at the coordinate position (1, .5).

Layout Units

You define a specific horizontal and vertical size for the display window using the OG_APP constants HSCREEN_RES and VSCREEN_RES. These are similar to OG_INCH, but are used exclusively for window dimensions.
By default, Graphics opens each display in a window named Main Layout. You can change the following properties of the default window:

- position in the Runtime layout
- horizontal and vertical size of the window
- window name

If you want to set properties for the default window, you should do so in an Open Display trigger such as the following:

```plaintext
PROCEDURE OGTRIGGERPROC0 IS
  the_win    OG_WINDOW;
  win_pos    OG_POINT;
  win_wid    NUMBER;
  win_ht     NUMBER;
BEGIN
  the_win:=OG_GET_WINDOW('Main Layout');
  /*Define the position for the window*/
  win_pos.x:=1*OG_INCH;
  win_pos.y:=.5*OG_INCH;
  /*Set the position property for the window*/
  OG_SET_POSITION(the_win, win_pos);
  /*Define the size of the window*/
  win_wid:=4*OG_APP.HSCREEN_RES;
  win_ht:=5*OG_APP.VSCREEN_RES;
  /*Set the size property of the window*/
  OG_SET_WINDOW_SIZE(the_win, win_wid, win_ht);
  /*Assign a name to the window*/
  OG_SET_WINDOW_NAME(the_win, 'Start Here');
END;
```

When the display is opened in Graphics Runtime, the display will be opened in a 4x5 window with the title 'Start Here' at the coordinate position (1, .5).

You define a specific horizontal and vertical size for the display window using the OG_APP constants HSCREEN_RES and VSCREEN_RES. These are similar to OG_INCH, but are used exclusively for window dimensions.
Horizontal and vertical layout resolution are always equal. The OG_APP.HSCREEN_RES and OG_APP.VSCREEN_RES constants represent the number of layout units (pixels) in one horizontal inch and one vertical inch, respectively. This unit of measurement ensures that your display window will appear the same regardless of the screen resolution on which it is used.

You use the OG_POINT PL/SQL built-in datatype to represent (x,y) coordinates on the layout. The value (0,0) represents the upper left corner of the display. These coordinates are expressed in layout units. For more information on layout units and the OG_POINT datatype, see “Specifying Size and Coordinate Information” on page 12 – 5.

Chaining Displays

You can open one display from another using the PL/SQL built-in procedure OG_OPEN_DISPLAY. You can either open a second display and keep the original display open, or close the original display.

If you decide to open a new display and close the original one, you must be sure that the OG_CLOSE_DISPLAY subprogram is last in the stack. This is to ensure that all other subprograms have run before the close display subprogram is called. Any subprograms listed after the OG_CLOSE_DISPLAY subprogram will not be called. Upon the OG_CLOSE_DISPLAY subprogram, the Close Display trigger, if any, is executed.

For example:

```sql
PROCEDURE next_display IS
    next_disp  OG_DISPLAY;
    old_disp   OG_DISPLAY;
BEGIN
    next_display:=OG_OPEN_DISPLAY(‘more.ogd’, OG_FILESYSTEM);
    old_disp:=OG_GET_DISPLAY(‘plan.ogd’);
    OG_CLOSE_DISPLAY(old_disp);
END;
```
Horizontal and vertical layout resolution are always equal. The OG_APP.HSCREEN_RES and OG_APP.VSCREEN_RES constants represent the number of layout units (pixels) in one horizontal inch and one vertical inch, respectively. This unit of measurement ensures that your display window will appear the same regardless of the screen resolution on which it is used.

You use the OG_POINT PL/SQL built-in datatype to represent (x,y) coordinates on the layout. The value (0,0) represents the upper left corner of the display. These coordinates are expressed in layout units. For more information on layout units and the OG_POINT datatype, see "Specifying Size and Coordinate Information" on page 12-5.

Chaining Displays

You can open one display from another using the PL/SQL built-in procedure OG_OPEN_DISPLAY. You can either open a second display and keep the original display open, or close the original display. If you decide to open a new display and close the original one, you must be sure that the OG_CLOSE_DISPLAY subprogram is last in the stack. This is to ensure that all other subprograms have run before the close display subprogram is called. Any subprograms listed after the OG_CLOSE_DISPLAY subprogram will not be called. Upon the OG_CLOSE_DISPLAY subprogram, the Close Display trigger, if any, is executed.

For example:

PROCEDURE next_display IS
  next_disp  OG_DISPLAY;
  old_disp   OG_DISPLAY;
BEGIN
  next_display:=OG_OPEN_DISPLAY('more.ogd', OG_FILESYSTEM);
  old_disp:=OG_GET_DISPLAY('plan.ogd');
  OG_CLOSE_DISPLAY(old_disp);
END;
Sharing Information Among Displays

You can share data between displays using the following methods:

- create a package to store global variables
- use database pipes and the DBMS_PIPE package

Creating a Package

A package is a collection of related PL/SQL procedures, functions, datatypes, variables, and other constructs that is stored as a database object. Displays can share global variables stored in a package as long as all displays involved maintain the same database connection. Once a display disconnects, the package state is reset.

To ensure that your displays maintain the same database connection, use the OPEN_DISPLAY PL/SQL built-in procedure to open one display from another. See the “Chaining Displays” section above for more information.

The package you create should contain both the variable definitions, and subprograms to manipulate the variables (i.e., procedures to set the variable value, and functions to retrieve the variable value). You can reference the variable and the associated subprograms in any display that shares the database connection.

The following example shows a package specification and body that contains a global variable `bar` and two subprograms:

```sql
PACKAGE foo IS
  bar  VARCHAR2;
  PROCEDURE set_bar (txt VARCHAR2);
  FUNCTION get_bar return VARCHAR2;
END;

PACKAGE BODY foo IS
  PROCEDURE set_bar (txt VARCHAR2) IS
    BEGIN
      foo.bar:=txt;
    END;

  FUNCTION get_bar RETURN VARCHAR2 IS
    BEGIN
      return foo.bar;
    END;
END;
```
You can share data between displays using the following methods:

- create a package to store global variables
- use database pipes and the DBMS_PIPE package

A package is a collection of related PL/SQL procedures, functions, datatypes, variables, and other constructs that is stored as a database object. Displays can share global variables stored in a package as long as all displays involved maintain the same database connection. Once a display disconnects, the package state is reset.

To ensure that your displays maintain the same database connection, use the OPEN_DISPLAY PL/SQL built-in procedure to open one display from another. See the "Chaining Displays" section above for more information.

The package you create should contain both the variable definitions, and subprograms to manipulate the variables (i.e., procedures to set the variable value, and functions to retrieve the variable value). You can reference the variable and the associated subprograms in any display that shares the database connection.

The following example shows a package specification and body that contains a global variable `bar` and two subprograms:

```plsql
PACKAGE foo IS
  bar  VARCHAR2;
  PROCEDURE set_bar (txt VARCHAR2);
  FUNCTION get_bar return VARCHAR2;
END;

PACKAGE BODY foo IS
  PROCEDURE set_bar (txt VARCHAR2) IS
    BEGIN
      foo.bar:=txt;
    END;

  FUNCTION get_bar RETURN VARCHAR2 IS
    BEGIN
      return foo.bar;
    END;
END;
```
Once you have created the package, you can reference the global variable and manipulate its value between displays by using the stored subprograms. For example:

```plaintext
PROCEDURE set_param IS
    BEGIN
        /*assign the local parameter 'param' to the value of the
global variable bar*/
        :param:=foo.get_bar;

        /*change the value of bar for the next time it is
referred*/
        foo.set_bar('object1');
    END;
```

For more information about creating PL/SQL packages, see the PL/SQL User's Guide and Reference.

Using Database Pipes

You can use the Oracle7 Server package DBMS_PIPE to pass information from one display to another. Using pipes to share information involves the following steps:

1. Use the PACK_MESSAGE procedure to add information to the local message buffer.
2. Use the SEND_MESSAGE procedure to designate a pipe through which to send the information in the message buffer.
3. Use the RECIEVE_MESSAGE procedure to receive the message on the named pipe.
4. Use the UNPACK _MESSAGE procedure to access the information in the buffer.

The following procedure packs a variable value to a message buffer and then sends the buffer to the server using a pipe named 'the_pipe'.

```plaintext
PROCEDURE pass_var (my_var NUMBER) IS
    stat NUMBER;
    BEGIN
        DBMS_PIPE.PACK_MESSAGE(my_var);
        stat:=DBMS_PIPE.SEND_MESSAGE('the_pipe', 30, 8192);
    END;
```
Using Database Pipes

Once you have created the package, you can reference the global variable and manipulate its value between displays by using the stored subprograms. For example:

```plsql
PROCEDURE set_param IS
BEGIN
    /*assign the local parameter 'param' to the value of the global variable bar*/
    :param:=foo.get_bar;
    /*change the value of bar for the next time it is referenced*/
    foo.set_bar('object1');
END;
```

For more information about creating PL/SQL packages, see the PL/SQL User's Guide and Reference.

You can use the Oracle7 Server package DBMS_PIPE to pass information from one display to another. Using pipes to share information involves the following steps:

1. Use the PACK_MESSAGE procedure to add information to the local message buffer.
2. Use the SEND_MESSAGE procedure to designate a pipe through which to send the information in the message buffer.
3. Use the RECEIVE_MESSAGE procedure to receive the message on the named pipe.
4. Use the UNPACK_MESSAGE procedure to access the information in the buffer.

The following procedure packs a variable value to a message buffer and then sends the buffer to the server using a pipe named 'the_pipe'.

```plsql
PROCEDURE pass_var (my_var NUMBER) IS
    stat NUMBER;
BEGIN
    DBMS_PIPE.PACK_MESSAGE(my_var);
    stat:=DBMS_PIPE.SEND_MESSAGE('the_pipe', 30, 8192);
END;
```
Note: Graphics does not currently recognize default values for arguments in the specs of stored subprograms. Thus, while DBMS_PIPE.SEND_MESSAGE supplied default values for its TIMEOUT and MAXPIPESIZE arguments, you must provide explicit values.

The following procedure can be executed from a second display that shares the same database connection.

```sql
PROCEDURE get_var IS
  my_var  NUMBER;
  stat    NUMBER;
BEGIN
  /*Receive the pipe from the server */
  stat:=DBMS_PIPE.RECEIVE_MESSAGE('the_pipe', 30);
  /*Unpack the information in the pipe and store in my_var*/
  DBMS_PIPE.UNPACK_MESSAGE(my_var);
  /*Use the variable value to perform some action*/
  use_variable(my_var);
END;
```

For more information about the DBMS_PIPE package, see the Oracle7 Server Application Developer’s Guide.

Managing Windows

Use the following sections to manage the windows for your runtime display:

- creating new windows
- hiding window scrollbars
- controlling window visibility
- using layers with windows

Creating New Windows

You can create multiple windows for a runtime display. You may want to show different layers of a display in different windows, or chain displays together by opening one from another.

You can use the OG_MAKE_WINDOW function to create a new window. For example:

```sql
PROCEDURE make_win IS
  nu_win   OG_WINDOW;
  win_pos  OG_INCH;
  win_wid  NUMBER;
```
Note: Graphics does not currently recognize default values for arguments in the specs of stored subprograms. Thus, while DBMS_PIPE.SEND_MESSAGE supplied default values for its TIMEOUT and MAXPIPESIZE arguments, you must provide explicit values.

The following procedure can be executed from a second display that shares the same database connection.

PROCEDURE get_var IS
  my_var  NUMBER;
  stat    NUMBER;
BEGIN
  /*Receive the pipe from the server */
  stat:=DBMS_PIPE.RECEIVE_MESSAGE('the_pipe', 30);
  /*Unpack the information in the pipe and store in my_var*/
  DBMS_PIPE.UNPACK_MESSAGE(my_var);
  /*Use the variable value to perform some action*/
  use_variable(my_var);
END;

For more information about the DBMS_PIPE package, see the Oracle7 Server Application Developer's Guide.

Managing Windows

Use the following sections to manage the windows for your runtime display:

• creating new windows
• hiding window scrollbars
• controlling window visibility
• using layers with windows

You can create multiple windows for a runtime display. You may want to show different layers of a display in different windows, or chain displays together by opening one from another.

You can use the OG_MAKE_WINDOW function to create a new window. For example:

PROCEDURE make_win IS
  nu_win   OG_WINDOW;
  win_pos  OG_INCH;
  win_wid  NUMBER;
BEGIN
  /*Other code here*/
END;
Hiding the Window Scrollbars

Window scrollbars are created by default. You can hide the scrollbars in a new window using the OG_WINDOW attribute record OG_SCROLLBARS_WINDOWA.

For example:

```plaintext
PROCEDURE make_window IS
  the_window  OG_WINDOW;
  windo_attr  OG_WINDOW_ATTR;
BEGIN
  windo_attr.name:='Inventory';
  windo_attr.scrollbars:=FALSE;
  windo_attr.width:=4*OG_APP.HSCREEN_RES;
  windo_attr.height:=5*OG_APP.VSCREEN_RES;
  windo_attr.mask:=OG_NAME_WINDOWA+
    OG_SCROLLBARS_WINDOWA+
    OG_SIZE_WINDOWA;
  the_window:=OG_MAKE(windo_attr);
END;
```

You cannot change the attribute records for an existing window. If you wish to hide the scrollbars for an existing window, you must hide or destroy that window and create a new window. Note that you cannot destroy the original default window named 'Main Layout.'

If you wish to hide scrollbars in the initial window of your runtime display, you should create an Open Display trigger that creates a new window without scrollbars, and hides the 'Main Layout' window.

Controlling Window Visibility

You can further manage the windows used in your Runtime application as follows:

```
hide
```

Use this option to temporarily hide a window that you may wish to show again at a later time.
Window scrollbars are created by default. You can hide the scrollbars in a new window using the OG_WINDOW attribute record `OG_SCROLLBARS_WINDOW A`. For example:

```cpp
PROCEDURE make_window IS
    the_window  OG_WINDOW;
    windo_attr  OG_WINDOW_ATTR;
BEGIN
    windo_attr.name:='Inventory';
    windo_attr.scrollbars:=FALSE;
    windo_attr.width:=4*OG_APP.HSCREEN_RES;
    windo_attr.height:=5*OG_APP.VSCREEN.RES;
    windo_attr.mask:=OG_NAME_WINDOWA+
                    OG_SCROLLBARS_WINDOWA+
                    OG_SIZE_WINDOWA;
    the_window:=OG_MAKE(windo_attr);
END;
```

You cannot change the attribute records for an existing window. If you wish to hide the scrollbars for an existing window, you must hide or destroy that window and create a new window. Note that you cannot destroy the original default window named 'Main Layout.' If you wish to hide scrollbars in the initial window of your runtime display, you should create an Open Display trigger that creates a new window without scrollbars, and hides the 'Main Layout' window.

You can further manage the windows used in your Runtime application as follows:

Use this option to temporarily hide a window that you may wish to show again at a later time.
show Use this option to reveal a window that was previously hidden.
destroy Use this option to close a window that you no longer need. This might be a window that has been opened from an existing display.

Note: You cannot destroy the default 'Main Layout' window. If you wish to destroy a default Main Layout window, use OG_CLOSE_DISPLAY, otherwise, use OG_HIDE_WINDOW.

Hiding a Window
You can temporarily hide a window from users, and show it again later. The following example shows a button procedure that is activated when a user selects a chart element:

PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object, hitobj IN og_object, win IN og_window, eveninfo IN og_event) IS
  curr_window  OG_WINDOW;
BEGIN
  curr_window:=OG_GET_WINDOW('Main Layout');
  OG_HIDE_WINDOW('Main Layout');
END;

Showing a Window
After you have hidden a window, you can reveal it again using the OG_SHOW_WINDOW PL/SQL built-in procedure.

For example:

PROCEDURE show_window IS
  back_window  OG_WINDOW;
BEGIN
  back_window:=OG_GET_WINDOW('Details');
  OG_SHOW_WINDOW(back_window);
END;

Destroying a Window
You can destroy any window that you created with OG_MAKE_WINDOW or OG_MAKE(Window) using the OG_DESTROY PL/SQL built-in. OG_DESTROY closes the window without affecting its contents.

Note: You cannot destroy a default 'Main Layout' window.

To close all windows associated with a display, use the OG_CLOSE_DISPLAY PL/SQL built-in procedure.
Hiding a Window

Use this option to reveal a window that was previously hidden.

Showing a Window

Use this option to close a window that you no longer need. This might be a window that has been opened from an existing display.

Note: You cannot destroy the default 'Main Layout' window. If you wish to destroy a default Main Layout window, use OG_CLOSE_DISPLAY, otherwise, use OG_HIDE_WINDOW.

You can temporarily hide a window from users, and show it again later. The following example shows a button procedure that is activated when a user selects a chart element:

```
PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object, hitobj IN og_object, win IN og_window, eveninfo IN og_event) IS
  curr_window  OG_WINDOW;
BEGIN
  curr_window:=OG_GET_WINDOW('Main Layout');
  OG_HIDE_WINDOW('Main Layout');
END;
```

After you have hidden a window, you can reveal it again using the OG_SHOW_WINDOW PL/SQL built-in procedure. For example:

```
PROCEDURE show_window IS
  back_window  OG_WINDOW;
BEGIN
  back_window:=OG_GET_WINDOW('Details');
  OG_SHOW_WINDOW(back_window);
END;
```

You can destroy any window that you created with OG_MAKE_WINDOW or OG_MAKE(Window) using the OG_DESTROY PL/SQL built-in. OG_DESTROY closes the window without affecting its contents.

Note: You cannot destroy a default 'Main Layout' window.

To close all windows associated with a display, use the OG_CLOSE_DISPLAY PL/SQL built-in procedure.
The following procedure destroys a window:

```plaintext
PROCEDURE close_window (the_window IN og_window) IS
BEGIN
    OG_DESTROY(the_window);
END;
```

If your display contains multiple layers, you can activate, hide, and show an existing layer among separate windows. This has the same affect as opening another display. Each layer can presented in a separate window, allowing the user to see multiple layers of a single display.

You can manipulate layers among multiple windows using the PL/SQL built-in procedures `OG_ACTIVATE_LAYER`, `OG_HIDE_LAYER`, and `OG_SHOW_LAYER`. Each of these procedures has an optional window handle argument. If you do not specify a specific window handle for the procedure, Graphics performs the specified action on each window the layer is included in.

Users can only interact with the active layer window. Remember that you cannot hide the active layer. You must first activate a new layer, and then hide the old one.

The following example shows a program unit that activates a second layer of the display to a new window:

```plaintext
PROCEDURE activate_layer IS
    new_layer   OG_LAYER;
    old_layer   OG_LAYER;
    new_window  OG_WINDOW;
BEGIN
    new_layer:=OG_GET_LAYER('layer1');
    old_layer:=OG_GET_LAYER('layer0');
    new_window:=OG_GET_WINDOW('Details');
    OG_ACTIVATE_LAYER(new_layer, new_window);
END;
```
The following procedure destroys a window:

```
PROCEDURE close_window (the_window IN og_window) IS
BEGIN
    OG_DESTROY(the_window);
END;
```

If your display contains multiple layers, you can activate, hide, and show an existing layer among separate windows. This has the same affect as opening another display. Each layer can present in a separate window, allowing the user to see multiple layers of a single display.

You can manipulate layers among multiple windows using the PL/SQL built-in procedures `OG_ACTIVATE_LAYER`, `OG_HIDE_LAYER`, and `OG_SHOW_LAYER`. Each of these procedures has an optional window handle argument. If you do not specify a specific window handle for the procedure, Graphics performs the specified action on each window the layer is included in.

Users can only interact with the active layer window. Remember that you cannot hide the active layer. You must first activate a new layer, and then hide the old one.

The following example shows a program unit that activates a second layer of the display to a new window:

```
PROCEDURE activate_layer IS
    new_layer   OG_LAYER;
    old_layer   OG_LAYE R;
    new_window  OG_WINDOW;
BEGIN
    new_layer:=OG_GET_LAYER('layer1');
    old_layer:=OG_GET_LAYER('layer0');
    new_window:=OG_GET_WINDOW('Details');
    OG_ACTIVATE_LAYER(new_layer, new_window);
END;
```
Chapter 16

Integrating Displays

This chapter describes how to integrate a Graphics display with other applications. You can include a Graphics display in a Forms application or a Reports report. Or, you can call other Oracle products such as Forms and Reports from a Graphics display.

Additionally, Graphics functions as an OLE server application. You can include a Graphics display in an OLE container application.

The following topics are covered in this chapter:

- preparing a display for embedding – 16 – 2
- including a display in Forms – 16 – 4
- including a display in Reports – 16 – 15
- including a display in OLE container applications – 16 – 15
- calling other products from Graphics – 16 – 16
Chapter 16

Integrating Displays

This chapter describes how to integrate a Graphics display with other applications. You can include a Graphics display in a Forms application or a Reports report. Or, you can call other Oracle products such as Forms and Reports from a Graphics display.

Additionally, Graphics functions as an OLE server application. You can include a Graphics display in an OLE container application.

The following topics are covered in this chapter:

• preparing a display for embedding
• including a display in Forms
• including a display in Reports
• including a display in OLE container applications
• calling other products from Graphics
Preparing a Display for Embedding

Before you can embed your Graphics display in an application built using a different tool, you should prepare your environment and your display so the tools can work together in an optimal fashion. Listed below are some of the issues to take into consideration:

- locating modules – 16 – 2
- sizing the display – 16 – 2
- executing the display – 16 – 3
- receiving text parameters – 16 – 3
- receiving data – 16 – 3
- building the layout – 16 – 4

Locating Modules

When Graphics is called upon to execute an embedded display, it must be able to find the display. Graphics searches for files in the following order:

1. The directory path prefixed to the file name, if any.
2. Your “current” or “working” directory, as determined by your operating system. For more information, see the Developer/2000 Installation Guide.
3. The path specified by the GRAPHICS25_PATH environment variable.
4. The path specified by the ORACLE_PATH environment variable.

Ensure that you either explicitly specify the location of your display, or that your path variables are correctly set and one of them marks the location of your display. For more information about setting these environment variables, see the Developer/2000 Installation Guide.

Sizing the Display

The size of a display as it appears in an application is not necessarily the size of the display in Graphics. When embedding a display, you need to determine whether you want to clip it, scale it, or create a display of the exact dimensions required to fit in the layout of the embedding application.

Scaling vs. Clipping

Because Developer/2000 components use the same graphic definition system, you can instruct Forms to scale or clip the display to fit into the designated display region. Reports automatically scales the display to fit in the display region.
Preparing a Display for Embedding

Before you can embed your Graphics display in an application built using a different tool, you should prepare your environment and your display so the tools can work together in an optimal fashion. Listed below are some of the issues to take into consideration:

- locating modules
- sizing the display
- executing the display
- receiving text parameters
- receiving data
- building the layout

When Graphics is called upon to execute an embedded display, it must be able to find the display. Graphics searches for files in the following order:

1. The directory path prefixed to the file name, if any.
2. Your "current" or "working" directory, as determined by your operating system. For more information, see the Developer/2000 Installation Guide.
3. The path specified by the GRAPHICS25_PATH environment variable.
4. The path specified by the ORACLE_PATH environment variable.

Ensure that you either explicitly specify the location of your display, or that your path variables are correctly set and one of them marks the location of your display. For more information about setting these environment variables, see the Developer/2000 Installation Guide.

The size of a display as it appears in an application is not necessarily the size of the display in Graphics. When embedding a display, you need to determine whether you want to clip it, scale it, or create a display of the exact dimensions required to fit in the layout of the embedding application.

Because Developer/2000 components use the same graphic definition system, you can instruct Forms to scale or clip the display to fit into the designated display region. Reports automatically scales the display to fit in the display region.
Scaling and clipping each have special considerations. Scaled displays can sometimes appear distorted. In the case of clipped displays, their sizes are calculated beginning in the upper left corner of the display area (coordinates 0,0). This means that if your display objects are displaced from the top and left borders of the Graphics Layout editor, that “empty” space is included in the clipped version of the display.

To avoid distorted displays or unnecessary empty space, you can take the following precautions:

- Move the contents of the display to the upper left corner of the Layout editor.
- Use View—>View Options—>Layout to show the Layout Settings dialog where you can specify the size of the display.

**Executing the Display**

Other Oracle products can invoke either the Graphics Runtime or Batch executables.

**Receiving Text Parameters**

A product that invokes a Graphics display can pass initial values to parameters defined in the display. These parameter values are always passed as datatype CHAR.

When you create a parameter in Graphics, specify a datatype for it in the Parameters dialog. Graphics implicitly converts the incoming CHAR parameter value to the datatype specified for the parameter.

If you want to pass a date from the invoking product to Graphics, you must specify the incoming date format in the Display property sheet. For example, if you wish to send a value of 04–JUL–92 to Graphics, make sure the date format DD–MON–YY is specified in the Parameter Date Format field of the Display property sheet.

For more information about text parameters, see “Text Parameters” on page 16 – 7.

**Receiving Data**

When you embed a Graphics display in another Oracle product, Graphics can use data from that product in place of its own.

When you create the chart in Graphics, check Update Chart on Query Execution in the Data property page of the Chart property sheet. The chart will be updated with the new data when the query is executed against the calling product.

**Note:** Make sure that the query used by the calling product contains the same column names as the query used to create the chart so that Graphics knows what to do with the data it’s sent. For example, suppose your display contains a chart that lists ENAME as a category and SAL as a value.

Integrating Displays 16 – 3
Scaling and clipping each have special considerations. Scaled displays can sometimes appear distorted. In the case of clipped displays, their sizes are calculated beginning in the upper left corner of the display area (coordinates 0,0). This means that if your display objects are displaced from the top and left borders of the Graphics Layout editor, that "empty" space is included in the clipped version of the display.

To avoid distorted displays or unnecessary empty space, you can take the following precautions:

• Move the contents of the display to the upper left corner of the Layout editor.
• Use View —> View Options —> Layout to show the Layout Settings dialog where you can specify the size of the display.

Other Oracle products can invoke either the Graphics Runtime or Batch executables. A product that invokes a Graphics display can pass initial values to parameters defined in the display. These parameter values are always passed as datatype CHAR.

When you create a parameter in Graphics, specify a datatype for it in the Parameters dialog. Graphics implicitly converts the incoming CHAR parameter value to the datatype specified for the parameter.

If you want to pass a date from the invoking product to Graphics, you must specify the incoming date format in the Display property sheet. For example, if you wish to send a value of 04–JUL–92 to Graphics, make sure the date format DD–MON–YY is specified in the Parameter Date Format field of the Display property sheet.

For more information about text parameters, see "Text Parameters" on page 16 – 7.

When you embed a Graphics display in another Oracle product, Graphics can use data from that product in place of its own.

When you create the chart in Graphics, check Update Chart on Query Execution in the Data property page of the Chart property sheet. The chart will be updated with the new data when the query is executed against the calling product.

Note: Make sure that the query used by the calling product contains the same column names as the query used to create the chart so that Graphics knows what to do with the data it's sent. For example, suppose your display contains a chart that lists ENAME as a category and SAL as a value.
If Forms substitutes data that has columns named EMPLOYEE and SALARY, the chart will not be able to find the columns it is expecting.

For more information about data parameters, see “Data Parameters” on page 16–8.

If you assigned aliases in your Graphics query, you may want to remove them for clarity. Assign custom Axis labels via the Axis property sheet.

**Building the Layout**

After invoking Graphics, Reports always receives the contents of the layout following display execution. After invoking Graphics, Forms receives the contents of the layout specified in the OG.OPEN procedure.

**Including a Display in Forms**

When integrating Forms and Graphics, your resulting applications can range from an embedded display that simply charts its own data, supplied by its own query, to an embedded display that receives data from the form, passes it back, and is sensitive to such user actions as mouse clicks and buttons. This means you can control the level of network traffic, the number of times a query is executed, etc., based on the particular requirements of your application.

The level of integration depends on the level of communication you establish between Graphics and Forms. To get Forms to “talk” to Graphics, use the packaged subprograms in the OG.PLL library.

**The OG.PLL Library**

Forms includes the OG.PLL library, a package that includes several procedures and functions you can use to communicate with a display. With this package you can do the following:

- embed the display in the form – 16–5
- activate the display – 16–5
- send PL/SQL to the display – 16–6
- pass data between the form and the display – 16–6

Generally, to communicate between Forms and Graphics, you will include one or more of the OG.PLL procedures and functions in one or more form triggers, so that actions taking place in the form send information and commands to the embedded display.

The OG.PLL library and its subprograms are discussed in more detail in the following sections. Also, for more information about the OG library and its subprograms, see *Forms Advanced Techniques*. 
If Forms substitutes data that has columns named EMPLOYEE and SALARY, the chart will not be able to find the columns it is expecting. For more information about data parameters, see "Data Parameters" on page 16–8.

If you assigned aliases in your Graphics query, you may want to remove them for clarity. Assign custom Axis labels via the Axis property sheet.

After invoking Graphics, Reports always receives the contents of the layout following display execution. After invoking Graphics, Forms receives the contents of the layout specified in the OG.OPEN procedure.

Including a Display in Forms

When integrating Forms and Graphics, your resulting applications can range from an embedded display that simply charts its own data, supplied by its own query, to an embedded display that receives data from the form, passes it back, and is sensitive to such user actions as mouse clicks and buttons. This means you can control the level of network traffic, the number of times a query is executed, etc., based on the particular requirements of your application.

The level of integration depends on the level of communication you establish between Graphics and Forms. To get Forms to "talk" to Graphics, use the packaged subprograms in the OG.PLL library.

Forms includes the OG.PLL library, a package that includes several procedures and functions you can use to communicate with a display. With this package you can do the following:

- embed the display in the form
- activate the display
- send PL/SQL to the display
- pass data between the form and the display

Generally, to communicate between Forms and Graphics, you will include one or more of the OG.PLL procedures and functions in one or more form triggers, so that actions taking place in the form send information and commands to the embedded display.

The OG.PLL library and its subprograms are discussed in more detail in the following sections. Also, for more information about the OG library and its subprograms, see Forms Advanced Techniques.
Embedding the Display

Simply embedding a display in a Forms application without making it sensitive to mouse clicks, without passing data to it or sending data from it — in other words, without establishing communication between the form and the display — is the most basic type of integration. Yet it can still be quite useful. Using an embedded display, you can present pertinent information in a more intuitive fashion.

To embed a display in a form, use the packaged procedure OG.OPEN, included in the OG.PLL library. With this procedure you can specify whether you want the display clipped or scaled, (clipped is the default), whether to automatically refresh the display or not (refreshing the display is the default), and you can specify a parameter list to pass to Graphics.

You can include the OG.OPEN procedure in a number of different triggers; for example, the When–New–Form–Instance trigger will load the display at the successful creation of the form, or the When–Button–Pressed trigger will load the display when the user selects the appropriate button.

Note: The trigger determines when a display is loaded, not when it is actually displayed in the form. If the display is located in a block that is not active when the trigger fires, the display will be loaded, but that block will not receive focus.

Once Forms has invoked the appropriate Graphics executable and loaded the display, it remains in memory until explicitly closed. This has certain ramifications; if you change the chart object and want to test your changes, the display must be closed first. Graphics will not reload an open display. You can close your display using OG.CLOSE, also included in the OG.PLL library. Include OG.CLOSE in a trigger in much the same fashion as OG.OPEN.

To actually exit out of Graphics, use the OG_QUIT built-in procedure in another trigger, such as a Post–Block or Post–Form trigger. OG_QUIT differs from OG.OPEN and OG.CLOSE in that it is not part of the OG.PLL library included with Forms, but instead is a Graphics built-in subprogram, and as such must be passed to Graphics to interpret. OG.PLL includes OG.INTERPRET for that purpose. Use OG.INTERPRET to pass any valid PL/SQL string to Graphics to execute. For more information about passing PL/SQL commands via OG.INTERPRET, see “Sending PL/SQL to the Display” on page 16 – 6.

Activating the Display

However, suppose that your embedded display is actually a drill-down display, and you would like a user running the form to be able to access the detail displays.
Embedding the Display

Activating the Display

16 - 5 Integrating Displays

Simply embedding a display in a Forms application without making it sensitive to mouse clicks, without passing data to it or sending data from it — in other words, without establishing communication between the form and the display — is the most basic type of integration. Yet it can still be quite useful. Using an embedded display, you can present pertinent information in a more intuitive fashion.

To embed a display in a form, use the packaged procedure OG.OPEN, included in the OG.PLL library. With this procedure you can specify whether you want the display clipped or scaled, (clipped is the default), whether to automatically refresh the display or not (refreshing the display is the default), and you can specify a parameter list to pass to Graphics.

You can include the OG.OPEN procedure in a number of different triggers; for example, the When–New–Form–Instance trigger will load the display at the successful creation of the form, or the When–Button–Pressed trigger will load the display when the user selects the appropriate button.

Note: The trigger determines when a display is loaded, not when it is actually displayed in the form. If the display is located in a block that is not active when the trigger fires, the display will be loaded, but that block will not receive focus.

Once Forms has invoked the appropriate Graphics executable and loaded the display, it remains in memory until explicitly closed. This has certain ramifications; if you change the chart object and want to test your changes, the display must be closed first. Graphics will not reload an open display. You can close your display using OG.CLOSE, also included in the OG.PLL library. Include OG.CLOSE in a trigger in much the same fashion as OG.OPEN.

To actually exit out of Graphics, use the OG_QUIT built-in procedure in another trigger, such as a Post–Block or Post–Form trigger. OG_QUIT differs from OG.OPEN and OG.CLOSE in that it is not part of the OG.PLL library included with Forms, but instead is a Graphics built-in subprogram, and as such must be passed to Graphics to interpret. OG.PLL includes OG.INTERPRET for that purpose. Use OG.INTERPRET to pass any valid PL/SQL string to Graphics to execute. For more information about passing PL/SQL commands via OG.INTERPRET, see “Sending PL/SQL to the Display” on page 16 - 6.

However, suppose that your embedded display is actually a drill-down display, and you would like a user running the form to be able to access the detail displays.
The simplest method is to use the OG.MOUSEDOWN and OG.MOUSEUP packaged subprograms provided by the OG.PLL library to notify a display of a mouse event, so that Graphics can take the appropriate action.

When the user takes the action for which you’ve created a trigger (i.e., When–Button–Pressed or When–Mouse–Click), the trigger passes the information directly to Graphics, which responds just as it would if the mouse event had occurred in Graphics runtime.

This trigger need not be attached to a mouse event. If you want users to reach a drill–down display by pressing a button, create the button in the Forms layout and attach a When–Button–Pressed trigger that includes the appropriate procedure.

Both OG.MOUSEUP and OG.MOUSEDOWN give you complete control over the information sent from the form to the display. For example, if you want the user to press a button in the form instead of using a mouse, you can still pass all necessary information to the display — including coordinates, single– or double–click, button one or button two — and Graphics will treat it like an actual mouse event.

The default for all arguments denoting mouse actions is the actual information sent by the mouse.

**Sending PL/SQL to the Display**

When you include a Graphics display in a Forms application, you can use the OG.INTERPRET procedure to instruct the display to execute a specific piece of PL/SQL code. This PL/SQL can range from a Graphics built–in to a user–created function or procedure to a simple, one–line command, but it must be sent as a character string. Enclose all such strings in quotes, and include trailing semicolons as required.

Using OG.INTERPRET, you can execute Graphics program units from the form. Beyond simply changing the data upon which a chart is based, this allows you to use PL/SQL program units in the display to perform dynamic operations such as change a chart bar’s color or move an object.

**Exchanging Data Between the Form and the Display**

In addition to sending mouse actions or PL/SQL to an embedded display, you can pass information either from the form to the display, or from the display to the form. In this section we will discuss how the form either extracts information from a display, or sends information to the display. For more information about passing information from Graphics to Forms, see “Calling Other Products from Graphics” on page 16 – 16.

Regardless of whether you want the form to extract data from the display or to pass data to the display, you will use parameters.
Sending PL/SQL to the Display

Exchanging Data Between the Form and the Display

The simplest method is to use the OG.MOUSEDOWN and OG.MOUSEUP packaged subprograms provided by the OG.PLL library to notify a display of a mouse event, so that Graphics can take the appropriate action.

When the user takes the action for which you've created a trigger (i.e., When–Button–Pressed or When–Mouse–Click), the trigger passes the information directly to Graphics, which responds just as it would if the mouse event had occurred in Graphics runtime.

This trigger need not be attached to a mouse event. If you want users to reach a drill–down display by pressing a button, create the button in the Forms layout and attach a When–Button–Pressed trigger that includes the appropriate procedure.

Both OG.MOUSEUP and OG.MOUSEDOWN give you complete control over the information sent from the form to the display. For example, if you want the user to press a button in the form instead of using a mouse, you can still pass all necessary information to the display — including coordinates, single– or double–click, button one or button two — and Graphics will treat it like an actual mouse event.

The default for all arguments denoting mouse actions is the actual information sent by the mouse.

When you include a Graphics display in a Forms application, you can use the OG.INTERPRET procedure to instruct the display to execute a specific piece of PL/SQL code. This PL/SQL can range from a Graphics built–in to a user–created function or procedure to a simple, one–line command, but it must be sent as a character string. Enclose all such strings in quotes, and include trailing semicolons as required.

Using OG.INTERPRET, you can execute Graphics program units from the form. Beyond simply changing the data upon which a chart is based, this allows you to use PL/SQL program units in the display to perform dynamic operations such as change a chart bar's color or move an object.

In addition to sending mouse actions or PL/SQL to an embedded display, you can pass information either from the form to the display, or from the display to the form. In this section we will discuss how the form either extracts information from a display, or sends information to the display. For more information about passing information from Graphics to Forms, see "Calling Other Products from Graphics" on page 16 – 16.

Regardless of whether you want the form to extract data from the display or to pass data to the display, you will use parameters.
Extracting Parameter Values from a Display

Suppose, for example, that the master chart in your drill–down display is a pie chart. Each slice represents a department, and clicking on a slice displays a chart with detail information for that department.

Now suppose that your form contains a text item echoing the name of the selected department. That information is specific to your display. How do you extract it from the display into the text item?

Use OG.GETCHARPARAM, a function included in the OG.PLL library, which also provides OG.GETNUMPARAM for parameters whose values are numbers.

Passing Parameter Values to a Display

Now suppose that instead of clicking on a pie slice to update the chart, you want your users to enter the department name, thereby updating the detail chart in your display. Or perhaps your form has already executed the query needed by the display, and you want the display to use that data, instead of executing the query again. How do you pass this information to the display?

There are two types of parameters you can use to pass data:

Text parameters

- Contain values that come from variables or constants. They refer to system parameters, user–defined parameters, bind variables, or lexical variables.

Data parameters

- Point to a record group defined in the current form. Use data parameters when you want Graphics to use the result of a query executed in Forms without re–executing it.

The display can then use the information during updates.

Text Parameters

Text parameters can be pre–defined system parameters or user–defined parameters. Each text parameter has a key, or name, and a value.

Pre–defined System Parameters

Pre–defined system parameters have fixed names, or keys, and are usually specified in the command line that calls the tool. For example, PRINT is a parameter key, and ‘yes’ is one possible value; USERID is a key, and ‘scott/tiger@t:London:payroll’ is a valid value.

Note: By default, when Forms invokes Graphics, Graphics logs on to ORACLE with the same CONNECT information as the current Forms session.
Suppose, for example, that the master chart in your drill–down display is a pie chart. Each slice represents a department, and clicking on a slice displays a chart with detail information for that department.

Now suppose that your form contains a text item echoing the name of the selected department. That information is specific to your display. How do you extract it from the display into the text item?

Use OG.GETCHARPARAM, a function included in the OG.PLL library, which also provides OG.GETNUMPARAM for parameters whose values are numbers.

Now suppose that instead of clicking on a pie slice to update the chart, you want your users to enter the department name, thereby updating the detail chart in your display. Or perhaps your form has already executed the query needed by the display, and you want the display to use that data, instead of executing the query again. How do you pass this information to the display?

There are two types of parameters you can use to pass data:

- Contain values that come from variables or constants. They refer to system parameters, user–defined parameters, bind variables, or lexical variables.
- Point to a record group defined in the current form.

Use data parameters when you want Graphics to use the result of a query executed in Forms without re–executing it. The display can then use the information during updates.

Text parameters can be pre–defined system parameters or user–defined parameters. Each text parameter has a key and a value.

Pre–defined System Parameters

Pre–defined system parameters have fixed names, or keys, and are usually specified in the command line that calls the tool. For example, PRINT is a parameter key, and 'yes' is one possible value; USERID is a key, and 'scott/tiger@t:london:payroll' is a valid value.

Note: By default, when Forms invokes Graphics, Graphics logs on to ORACLE with the same CONNECT information as the current Forms session.
User–defined Parameters

User–defined parameters are generally defined during the design phase of an application, and give users more control over the actions of the modules to which they apply. That is also their purpose in an integrated application—i.e., having access to a display parameter from a form gives users more control over the entire integrated application. For more information about creating parameters in a form, see your Forms documentation. For more information about creating parameters in a display, see Chapter 10, “Using Parameters.”.

User–defined parameters passed between products are first converted to the CHAR datatype. However, the invoked product may either implicitly convert the values to the desired datatype, or allow you to treat them as values of another type (e.g., by using PL/SQL conversion functions such as TO_DATE, TO_NUMBER, etc.).

For more information about converting datatypes with a conversion function, see the PL/SQL User’s Guide and Reference.

Data Parameters

Data parameters are record groups you can pass from Forms to Graphics in a tabular format. You may want to do this for the following reasons:

• If the display dynamically builds and executes a query based on a user action in the form, you can provide Graphics with the result of that user action.

• If the display uses data that Forms has already retrieved, you can prevent that product from having to retrieve the data again.

• If the form has created an uncommitted record group you want to display.

As with text parameters, every data parameter has a key and a value. The value of a data parameter specifies the name of the Forms query passed to the display. The key of a data parameter specifies the query name with which it is equated in the invoked product.

When you are passing data from Forms to Graphics, make sure you first execute the query in Forms.

Creating and Populating Parameter Lists

Forms passes information to a display using a parameter list. A parameter list is a series of parameter key and value pairs. (Note: Even a single parameter must be assigned to a parameter list.) Each piece of information you want to pass must be defined as a parameter, then added to a parameter list. The type of parameter you create depends on the information you want to pass. Parameter lists are always passed as string literals.
User-defined parameters are generally defined during the design phase of an application, and give users more control over the actions of the modules to which they apply. That is also their purpose in an integrated application—i.e., having access to a display parameter from a form gives users more control over the entire integrated application. For more information about creating parameters in a form, see your Forms documentation. For more information about creating parameters in a display, see Chapter 10, "Using Parameters."

User-defined parameters passed between products are first converted to the CHAR datatype. However, the invoked product may either implicitly convert the values to the desired datatype, or allow you to treat them as values of another type (e.g., by using PL/SQL conversion functions such as TO_DATE, TO_NUMBER, etc.).

For more information about converting datatypes with a conversion function, see the PL/SQL User's Guide and Reference.

Data parameters are record groups you can pass from Forms to Graphics in a tabular format. You may want to do this for the following reasons:

• If the display dynamically builds and executes a query based on a user action in the form, you can provide Graphics with the result of that user action.
• If the display uses data that Forms has already retrieved, you can prevent that product from having to retrieve the data again.
• If the form has created an uncommitted record group you want to display.

As with text parameters, every data parameter has a key and a value. The value of a data parameter specifies the name of the Forms query passed to the display. The key of a data parameter specifies the query name with which it is equated in the invoked product.

When you are passing data from Forms to Graphics, make sure you first execute the query in Forms.

Forms passes information to a display using a parameter list. A parameter list is a series of parameter key and value pairs. (Note: Even a single parameter must be assigned to a parameter list.) Each piece of information you want to pass must be defined as a parameter, then added to a parameter list. The type of parameter you create depends on the information you want to pass. Parameter lists are always passed as string literals.
To enable you to create and manipulate parameter lists and the parameters they hold, Forms provides several built-in functions and procedures with self-explanatory names:

- CREATE_PARAMETER_LIST (function)
- GET_PARAMETER_LIST (function)
- DESTROY_PARAMETER_LIST (function)
- ADD_PARAMETER (procedure)
- SET_PARAMETER_ATTR (procedure)
- GET_PARAMETER_ATTR (procedure)
- DELETE_PARAMETER (procedure)

**Note:** For Graphics, these functions and procedures are part of the TOOL_INT package. When using them, preface their names with TOOL_INT.; e.g., TOOL_INT.GET_PARAMETER_LIST.

For more information about the TOOL_INT package, see the “Built-in Subprograms” chapter of the *Graphics Reference Manual*.

The following steps illustrate the basic process for creating a parameter list to pass information:

1. Decide what information you will pass.
2. Create the parameter list using the built-in appropriate to your tool.
3. Add items to the list. This procedure requires the following arguments:
   - the name of your parameter list
   - the name of the parameter you want to add
   - the type of the parameter you want to add (text or data)
   - the value of the parameter you want to add

If you are passing command line or executable options, make sure you specify the command line option keyword as the parameter name. See the documentation for the product you are invoking for all valid executable options.

**Invoking the Product**

After you have created and populated the parameter list, invoke Graphics and send the parameter list to it using any one of the following procedures included with OG.PLL:

- OG.OPEN
- OG.INTERPRET
To enable you to create and manipulate parameter lists and the parameters they hold, Forms provides several built-in functions and procedures with self-explanatory names:

- `CREATE_PARAMETER_LIST` (function)
- `GET_PARAMETER_LIST` (function)
- `DESTROY_PARAMETER_LIST` (function)
- `ADD_PARAMETER` (procedure)
- `SET_PARAMETER_ATTR` (procedure)
- `GET_PARAMETER_ATTR` (procedure)
- `DELETE_PARAMETER` (procedure)

Note: For Graphics, these functions and procedures are part of the `TOOL_INT` package. When using them, preface their names with `TOOL_INT.`; e.g., `TOOL_INT.GET_PARAMETER_LIST`.


The following steps illustrate the basic process for creating a parameter list to pass information:

1. Decide what information you will pass.
2. Create the parameter list using the built-in appropriate to your tool.
3. Add items to the list. This procedure requires the following arguments:
   - the name of your parameter list
   - the name of the parameter you want to add
   - the type of the parameter you want to add (text or data)
   - the value of the parameter you want to add

   If you are passing command line or executable options, make sure you specify the command line option keyword as the parameter name. See the documentation for the product you are invoking for all valid executable options.

After you have created and populated the parameter list, invoke Graphics and send the parameter list to it using any one of the following procedures included with OG.PLL:

- `OG.OPEN`
- `OG.INTERPRET`
Creating an Integrated Application

Now that we’ve considered all the issues, how to actually embed a display in a form? The following steps are necessary:

- create the display in Graphics
- draw a chart area in your Forms application
- create a trigger to initialize the Graphics display, and a trigger to close the display and/or exit Graphics
- (optional) create one or more triggers to pass mouse events to the display
- (optional) create one or more triggers to issue PL/SQL commands
- (optional) create one or more triggers to pass data to, and receive data from, the display
- run the form

Create the Display

Create a Graphics display as you normally would. For special considerations, see the section “Preparing the Display for Embedding” on page 16–2. Also see “Passing Data from the Form to the Display” on page 16–6.

Create a drill-down display as you normally would. For special considerations, see the section “Preparing the Display for Embedding” on page 16–2. For more information creating drill-down displays, see “Lesson Four: Create a Drill-down Relationship” on page 2–13.

Draw a Chart Area

Select the Chart Item tool from the Forms Tool palette and draw out a region in the form layout. Either rename the chart area in the Properties window or keep the default name.

Create Triggers to Open and Close the Display

To create a trigger to initialize the display, do the following in the Forms Designer:

1. Attach the OG.PLL library to your form. This library contains the OG package, made up of the subpackages you’ll use to communicate between Forms and Graphics.
   For more information about attaching libraries in Forms, see your Forms documentation.
2. Create a trigger that uses the OG.OPEN procedure to load the Graphics display.
Creating an Integrated Application

Create the Display

1. Draw a Chart Area
2. Create Triggers to Open and Close the Display

Now that we've considered all the issues, how to actually embed a display in a form? The following steps are necessary:

- create the display in Graphics
- draw a chart area in your Forms application
- create a trigger to initialize the Graphics display, and a trigger to close the display and/or exit Graphics
- (optional) create one or more triggers to pass mouse events to the display
- (optional) create one or more triggers to issue PL/SQL commands
- (optional) create one or more triggers to pass data to, and receive data from, the display
- run the form

Create a Graphics display as you normally would. For special considerations, see the section "Preparing the Display for Embedding" on page 16 – 2. Also see "Passing Data from the Form to the Display" on page 16 – 6.

Create a drill-down display as you normally would. For special considerations, see the section "Preparing the Display for Embedding" on page 16 – 2. For more information creating drill-down displays, see "Lesson Four: Create a Drill-down Relationship" on page 2 – 13.

Select the Chart Item tool from the Forms Tool palette and draw out a region in the form layout. Either rename the chart area in the Properties window or keep the default name.

To create a trigger to initialize the display, do the following in the Forms Designer:

1. Attach the OG.PLL library to your form. This library contains the OG package, made up of the subpackages you'll use to communicate between Forms and Graphics.

For more information about attaching libraries in Forms, see your Forms documentation.

2. Create a trigger that uses the OG.OPEN procedure to load the Graphics display.
3. If necessary, create a trigger that uses the OG.CLOSE procedure to close the display. Alternatively, use OG.INTERPRET to sent an exit command to Graphics.

For more information about creating triggers in Forms, see your Forms documentation.

**Example**
The following When–New–Form–Instance trigger opens the display `inv_disp.ogd` in the chart area `CHART_AREA1`, scales the display to fit the space allotted to it in the Forms layout, and automatically refreshes the display (REFRESH=TRUE is the default). No parameter list is passed.

\[ \text{OG.OPEN('inv\_disp.ogd', 'CHART\_AREA1', clip=FALSE)}; \]

**Example**
The following Post–Form trigger closes the display `inv_disp.ogd` in the chart area `CHART\_AREA1`. It takes only two parameters: `display\_name` and `chart\_item\_name`.

\[ \text{OG.CLOSE('inv\_disp.ogd', 'CHART\_AREA1')}; \]

Once again, you are not required to use the Post–Form trigger. Select the trigger best–suited to close the display when and where you want it to close.

**Example**
The following Post–Form trigger uses the Graphics built–in OG_QUIT to exit Graphics:

\[ \text{OG.INTERPRET('inv\_disp.ogd', 'CHART\_AREA1', 'og_quit;')}; \]

Notice the semicolon immediately following “og_quit”. When passing PL/SQL via OG.INTERPRET, you must include the trailing semicolon. Once again, you are not required to use the Post–Form trigger. Select the trigger best–suited to exit Graphics when you want to exit.

Create Trigger(s) to Pass Mouse Events

Create a trigger in Forms that includes either the OG.MOUSEDOWN or OG.MOUSEUP packaged subprogram from the OG.PLL library.

**Example**
The following When–Mouse–Click trigger activates the display `inv_disp.ogd`:

\[ \text{OG.MOUSEDOWN('inv\_disp.ogd', 'CHART\_AREA1')}; \]

The procedure in the example above used only two arguments: `display\_name` and `chart\_area\_name`. Both procedures take several other arguments, including the supposed x and y coordinates of the mouse click, which mouse button was clicked, how many times it was clicked, a parameter list, etc.
3. If necessary, create a trigger that uses the OG.CLOSE procedure to close the display. Alternatively, use OG.INTERPRET to send an exit command to Graphics.

For more information about creating triggers in Forms, see your Forms documentation.

The following When–New–Form–Instance trigger opens the display inv_disp.ogd in the chart area 'CHART_AREA1', scales the display to fit the space allotted to it in the Forms layout, and automatically refreshes the display (REFRESH=TRUE is the default). No parameter list is passed.

```
OG.OPEN('inv Disp.ogd', 'CHART_AREA1', clip=FALSE);
```

The following Post–Form trigger closes the display inv_disp.ogd in the chart area 'CHART_AREA1'. It takes only two parameters: `display_name` and `chart_item_name`.

```
OG.CLOSE('inv Disp.ogd', 'CHART_AREA1');
```

Once again, you are not required to use the Post–Form trigger. Select the trigger best-suited to close the display when and where you want it to close.

The following Post–Form trigger uses the Graphics built-in OG_QUIT to exit Graphics:

```
OG.INTERPRET('inv Disp.ogd', 'CHART_AREA1', 'og_quit;');
```

Notice the semicolon immediately following "og_quit". When passing PL/SQL via OG.INTERPRET, you must include the trailing semicolon.

Once again, you are not required to use the Post–Form trigger. Select the trigger best-suited to exit Graphics when you want to exit.

Create a trigger in Forms that includes either the OG.MOUSEDOWN or OG.MOUSEUP packaged subprogram from the OG.PLL library.

The following When–Mouse–Click trigger activates the display inv_disp.ogd:

```
OG.MOUSEDOWN('inv Disp.ogd', 'CHART_AREA1');
```

The procedure in the example above used only two arguments: `display_name` and `chart_area_name`. Both procedures take several other arguments, including the supposed x and y coordinates of the mouse click, which mouse button was clicked, how many times it was clicked, a parameter list, etc.
Create Trigger(s) to Pass PL/SQL

When you use the OG.INTERPRET packaged subprogram to invoke a Graphics program unit from Forms, you must pass the program unit and any necessary arguments as a string value. In other words, the program unit you are executing must be enclosed in single quotes.

Example

The following, included in a When–Button–Pressed trigger, invokes the 'set_timers' procedure:

OG.INTERPRET('inv_disp.ogd', 'CHART_AREA1', 'set_timers;');

Note the inclusion of the trailing semicolon.

Example

If the program unit you are invoking in Graphics requires that arguments be passed by Forms, use the concatenation operator to create the necessary string value. The following simple example instructs Graphics to execute the program unit “set_param” with an argument supplied from one of the text fields in the form:

OG.INTERPRET ('inv_disp.ogd', 'CHART_AREA1', 'set_param'|| TO_CHAR(:BLOCK1.category_cbox) || ');');

A slightly more intricate example of building a string using the concatenation operator:

OG.INTERPRET('inv_disp.ogd', 'CHART_AREA1', 'set_timer('|| TO_NUM(:BLOCK2.values_plist) || ','|| TO_NUM(:BLOCK3.more_cbox) || ');');

Breaking the above subprogram call into separate parts (each divided by a concatenation operator), you can see the following:

'set_timer(' Contains the procedure name and opening parentheses, enclosed in single quotes.

TO_NUM(:BLOCK2.values_plist) Contains the parameter from Forms, converted to the NUMBER datatype. This is an argument required by the set_timer procedure.

',' Contains the comma that separates the two supplied arguments.

TO_NUM(:BLOCK3.more_cbox) Contains the parameter from Forms, converted to the NUMBER datatype. This is the second argument to the set_timer procedure.
When you use the OG.INTERPRET packaged subprogram to invoke a Graphics program unit from Forms, you must pass the program unit and any necessary arguments as a string value. In other words, the program unit you are executing must be enclosed in single quotes. The following, included in a When–Button–Pressed trigger, invokes the 'set_timers' procedure:

OG.INTERPRET('inv_disp.ogd', 'CHART_AREA1', 'set_timers;');

Note the inclusion of the trailing semicolon.

If the program unit you are invoking in Graphics requires that arguments be passed by Forms, use the concatenation operator to create the necessary string value. The following simple example instructs Graphics to execute the program unit "set_param" with an argument supplied from one of the text fields in the form:

OG.INTERPRET ('inv_disp.ogd', 'CHART_AREA1', 'set_param(' || TO_CHAR(:BLOCK1.category_cbox) || ');');

A slightly more intricate example of building a string using the concatenation operator:

OG.INTERPRET('inv_disp.ogd', 'CHART_AREA1', 'set_timer(' || TO_NUM(:BLOCK2.values_plist) || ', ' || TO_NUM(:BLOCK3.more_cbox) || ');');

Breaking the above subprogram call into separate parts (each divided by a concatenation operator), you can see the following:

- Contains the procedure name and opening parentheses, enclosed in single quotes.
- Contains the parameter from Forms, converted to the NUMBER datatype. This is an argument required by the set_timer procedure.
- Contains the comma that separates the two supplied arguments.
- Contains the parameter from Forms, converted to the NUMBER datatype. This is the second argument to the set_timer procedure.
Create Trigger(s) to Pass Data

When passing data between a form and a display, remember that the form can both extract parameter values from the display and pass text and data parameters to the display. However, the form can only extract a single parameter value per inquiry, while it can only pass parameter lists.

Example

Below is the text of a block-level Post-Query trigger that might be used to pass a department number to Graphics for use in constructing the sal_chart display.

```sql
PROCEDURE pass_param IS
  pl   ParamList  /*Parameter list datatype*/
BEGIN
  /*Create the parameter list, named "my param list".*/
  pl := Create_Parameter_List('my_param_list');
  /*Add a text parameter to the list to supply a value for**the 'dept_num' parameter that Graphics is expecting*/
  Add_Parameter(pl, 'dept_num', TEXT_PARAMETER, TO_CHAR(:dept.deptno));
  /*Call Graphics to populate the chart item*/
  OG.REFRESH('sal_chart', 'dept.chart_item', pl);
  /*Destroy the parameter list*/
  Destroy_Parameter_List(pl);
END
```

Example

The following function in a When-Button-Pressed trigger retrieves the value of the Graphics display parameter param1 and assigns it to the text field 'TEXT_ITEM1' in the form:

```sql
:BLOCK1.TEXT_ITEM1 := OG.GETCHARPARAM
  ('inv_disp.ogd', 'CHART_AREA1', 'param1');
```

Example

Graphics sees the resulting string:

```sql
set_timer(x, y);
```

where the x and y arguments are the numbers sent from the form and converted to the NUMBER datatype.

`) ;’ Contains the procedure’s closing parentheses and semicolon, enclosed in single quotes.
When passing data between a form and a display, remember that the form can both extract parameter values from the display and pass text and data parameters to the display. However, the form can only extract a single parameter value per inquiry, while it can only pass parameter lists.

The following function in a When–Button–Pressed trigger retrieves the value of the Graphics display parameter \texttt{param1} and assigns it to the text field 'TEXT_ITEM1' in the form:

\begin{verbatim}
:BLOCK1.TEXT_ITEM1 := OG.GETCHARPARAM ('inv_disp.ogd', 'CHART_AREA1', 'param1');
\end{verbatim}

Below is the text of a block–level Post–Query trigger that might be used to pass a department number to Graphics for use in constructing the sal_chart display.

\begin{verbatim}
PROCEDURE pass_param IS
  pl   ParamList  /*Parameter list datatype*/
BEGIN
  /*Create the parameter list, named "my param list".*/
  pl := Create_Parameter_List('my_param_list');
  /*Add a text parameter to the list to supply a value for the 'dept_num' parameter that Graphics is expecting*/
  Add_Parameter(pl, 'dept_num', TEXT_PARAMETER, TO_CHAR(:dept.deptno));
  /*Call Graphics to populate the chart item*/
  OG.REFRESH('sal_chart', 'dept.chart_item', pl);
  /*Destroy the parameter list*/
  Destroy_Parameter_List(pl);
END
\end{verbatim}
Example

Below is the text of a block-level Post-Query trigger used to pass data to Graphics. In this example, Forms issues a query and stores the resulting data in a record group. A data parameter is used to pass the record group to Graphics, which uses the data to construct the sal_chart display.

```
Procedure pass_data is
  pl_id  PARAMLIST;
  rg_id  RECORDGROUP := Find_Group('chart_data');
  qry    VARCHAR(2000);
  stat   NUMBER;
BEGIN
  /*Prepare a query in a string based on the last
   **WHERE clause issued.*/
  qry := 'SELECT ename, sal FROM emp '
    || substr(:system.last_query,instr(:system.last_query,'WHERE'))||' order by empno';

  /*Check to see if the record group already exists;
   **if not, create it.*/
  IF Id_Null(rg_id) THEN
    rg_id := CREATE_GROUP_FROM_QUERY('chart_data', qry);
  END IF;

  /*As the record group may have already existed with old
   **data, clear it out and repopulate it based on the new
   **query*/
  DELETE_GROUP_ROW(rg_id, all_rows);
  stat := POPULATE_GROUP_WITH_QUERY(rg_id, qry);

  /*Now create a parameter list*/
  pl_id := CREATE_PARAMETER_LIST('pl_chart_data');

  /*Add a data parameter with the same name as the query in
   **the display*/
  Add_Parameter(pl_id,'query0',DATA_PARAMETER,'chart_data');

  /*Refresh an already-opened display with the new data*/
  OG.REFRESH('sal_chart','dept.chart_item',pl_id);

  /*Destroy the parameter list*/
  DESTROY_PARAMETER_LIST
END
```
Example

16 – 14 Graphics Developer's Guide

Below is the text of a block–level Post–Query trigger used to pass data to
Graphics. In this example, Forms issues a query and stores the resulting
data in a record group. A data parameter is used to pass the record
group to Graphics, which uses the data to construct the
sal_chart
display.

Procedure pass_data is
pl_id  PARAMLIST;
rg_id  RECORDGROUP := Find_Group('chart_data');
qry    VARCHAR(2000);
stat   NUMBER;
BEGIN
/*Prepare a query in a string based on the last
WHERE clause issued.*/
qry := 'SELECT ename, sal FROM emp '||
substr(:system.last_query,instr(:system.last_query,
'WHERE'))||' order by empno';
/*Check to see if the record group already exists;
if not, create it.*/
IF Id_Null(rg_id) THEN
rg_id := CREATE_GROUP_FROM_QUERY('chart_data', qry);
END IF;
/*As the record group may have already existed with old
data, clear it out and repopulate it based on the new
query*/
DELETE_GROUP_ROW(rg_id, all_rows);
stat := POPULATE_GROUP_WITH_QUERY(rg_id, qry);
/*Now create a parameter list*/
pl_id := CREATE_PARAMETER_LIST('pl_chart_data');
/*Add a data parameter with the same name asthe query in
the display*/
Add_Parameter(pl_id,'query0',DATA_PARAMETER,'chart_data');
/*Refresh an already–opened display with the new data*/
OG.REFRESH('sal_chart','dept.chart_item',pl_id);
*/Destroy the parameter list*/
DESTROY_PARAMETER_LIST
END
Run the Form  Make sure any triggers you’ve created compile correctly, and select File—>Run.

Including a Display in Reports

Including a display in a Reports application consists of the following steps:

• create the display in Graphics
• create a Graphics object in the report layout
• set properties for the Graphics object in Reports

Within Reports, you can map columns from your report to columns from the query used to create the display. If the query used to create the display uses parameters, you can map columns used in the report to parameter values.

For more information about including a Graphics display in a report, see your Reports documentation.

Including a Display in OLE Container Applications

Graphics functions as an OLE server. You can include Graphics displays in OLE container applications.

For information about the verbs and methods supported by Graphics, see the Graphics Reference Manual.

Sizing the Display  When you embed a Graphics display in an OLE container application, you are really including an image of the display. Because OLE server applications do not use the same graphic definition system as Graphics does, they are unable to scale the image of a Graphics display to fit into a defined region.

For this reason, you should set a specific size for your display.

1. Move the objects in your display to the upper left corner of the Layout Editor, allowing adequate blank space so that the display is legible.

2. Choose View—>View Options—>Layout to display the Layout Settings dialog.

3. Enter a horizontal and vertical size for the layout.
Run the Form

16 – 15

Integrating Displays

Make sure any triggers you've created compile correctly, and select File —> Run.

Including a Display in Reports

Including a display in a Reports application consists of the following steps:

• create the display in Graphics
• create a Graphics object in the report layout
• set properties for the Graphics object in Reports

Within Reports, you can map columns from your report to columns from the query used to create the display. If the query used to create the display uses parameters, you can map columns used in the report to parameter values.

For more information about including a Graphics display in a report, see your Reports documentation.

Including a Display in OLE Container Applications

Graphics functions as an OLE server. You can include Graphics displays in OLE container applications.

For information about the verbs and methods supported by Graphics, see the Graphics Reference Manual.

When you embed a Graphics display in an OLE container application, you are really including an image of the display. Because OLE server applications do not use the same graphic definition system as Graphics does, they are unable to scale the image of a Graphics display to fit into a defined region.

For this reason, you should set a specific size for your display.

1. Move the objects in your display to the upper left corner of the Layout Editor, allowing adequate blank space so that the display is legible.
2. Choose View —> View Options —> Layout to display the Layout Settings dialog.
3. Enter a horizontal and vertical size for the layout.
4. Choose the OK button to accept the layout size and close the dialog. The layout size will be restricted to the dimensions you have defined. Make sure the region you designate in your OLE container application is large enough to accommodate the size of the display.

**Embedding a Display**

To include a Graphics display in an OLE container application, such as a Microsoft Excel spreadsheet:

1. Create the display in Graphics.
2. Draw a chart area in your OLE container application.
3. Set the appropriate properties for the included display.

See the documentation for the OLE container application for instructions on how to include an OLE server object.

**Calling Other Products From Graphics**

You may find that the majority of your displays are embedded in other applications, but you can also invoke other products from a display. For example, a stand-alone display may chart the latest data from a form.

The following topics are covered in this section:

- preparing to invoke other Oracle products – 16 – 16
- invoking Forms from Graphics – 16 – 20
- invoking Reports from Graphics – 16 – 20
- invoking Oracle Book from Graphics – 16 – 21

**Preparing to Invoke Other Oracle Products**

Preparing to invoke other Oracle products from a Graphics display is quite similar to preparing to invoke a display from another product. However, instead of using, for example, the packaged procedures Forms provides in the OG.PLL library, you use the built-in package TOOL_INT.

With this package you can do the following:

- create a parameter list to pass information to another product
- add items to the list
- invoke the other product from your display

4. Choose the OK button to accept the layout size and close the dialog. The layout size will be restricted to the dimensions you have defined. Make sure the region you designate in your OLE container application is large enough to accommodate the size of the display.

To include a Graphics display in an OLE container application, such as a Microsoft Excel spreadsheet:

1. Create the display in Graphics.
2. Draw a chart area in your OLE container application.
3. Set the appropriate properties for the included display.

See the documentation for the OLE container application for instructions on how to include an OLE server object.

Calling Other Products From Graphics

You may find that the majority of your displays are embedded in other applications, but you can also invoke other products from a display. For example, a stand–alone display may chart the latest data from a form.

The following topics are covered in this section:

• preparing to invoke other Oracle products
• invoking Forms from Graphics
• invoking Reports from Graphics
• invoking Oracle Book from Graphics

Preparing to invoke other Oracle products from a Graphics display is quite similar to preparing to invoke a display from another product. However, instead of using, for example, the packaged procedures Forms provides in the OG.PLL library, you use the built-in package TOOL_INT.

With this package you can do the following:

• create a parameter list to pass information to another product
• add items to the list
• invoke the other product from your display

Passing Information to Other Products

Passing information from a Graphics display to another tool’s module requires the use of text and data parameters, as covered in “Exchanging Data Between the Form and the Display”, on 16 – 6.

Example

Suppose you want a Graphics display to invoke Reports with the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>scott/tiger</td>
</tr>
<tr>
<td>DESTYPE</td>
<td>printer</td>
</tr>
<tr>
<td>COPIES</td>
<td>2</td>
</tr>
<tr>
<td>my_param</td>
<td>67</td>
</tr>
</tbody>
</table>

In addition, you want Reports to base its report on data provided by Graphics. Assume the query defined in your report is named ‘Q_1’ and the query defined in your Graphics display is named ‘query0’.

The following procedure creates a parameter list named ‘my_plist’ and adds the above parameters to it:

```java
PROCEDURE create_plist IS
  the_list   tool_int.paramlist;
BEGIN
  the_list:=TOOL_INT.CREATE_PARAMETER_LIST('my_plist');
  TOOL_INT.ADD_PARAMETER(the_list, 'userid',
                          TOOL_INT.TEXT_PARAMETER, 'scott/tiger');
  TOOL_INT.ADD_PARAMETER(the_list, 'destype',
                          TOOL_INT.TEXT_PARAMETER, 'printer');
  TOOL_INT.ADD_PARAMETER(the_list, 'copies',
                          TOOL_INT.TEXT_PARAMETER, '2');
  TOOL_INT.ADD_PARAMETER(the_list, 'my_param',
                          TOOL_INT.TEXT_PARAMETER, '67');
  TOOL_INT.ADD_PARAMETER(the_list, 'Q_1',
                          TOOL_INT.DATA_PARAMETER, 'query0');
END;
```

Using the Same Login Session

When you invoke another product from Graphics, you can effectively share your login session with the invoked product. To do this, you must pass the login information you supplied with Graphics to the other product in the form of a parameter. This can also be handy for suppressing the Connect dialog box.
Passing Information to Other Products

Example Using the Same Login Session

Passing information from a Graphics display to another tool's module requires the use of text and data parameters, as covered in "Exchanging Data Between the Form and the Display", on 16–6.

Suppose you want a Graphics display to invoke Reports with the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>scott/tiger</td>
</tr>
<tr>
<td>DESTYPE</td>
<td>printer</td>
</tr>
<tr>
<td>COPIES</td>
<td>2</td>
</tr>
<tr>
<td>my_param</td>
<td>67</td>
</tr>
</tbody>
</table>

In addition, you want Reports to base its report on data provided by Graphics. Assume the query defined in your report is named 'Q_1' and the query defined in your Graphics display is named 'query0'.

The following procedure creates a parameter list named 'my_plist' and adds the above parameters to it:

```plaintext
PROCEDURE create_plist IS
  the_list   tool_int.paramlist;
BEGIN
  the_list:=TOOL_INT.CREATE_PARAMETER_LIST('my_plist');
  TOOL_INT.ADD_PARAMETER(the_list, 'userid', TOOL_INT.TEXT_PARAMETER, 'scott/tiger');
  TOOL_INT.ADD_PARAMETER(the_list, 'destype', TOOL_INT.TEXT_PARAMETER, 'printer');
  TOOL_INT.ADD_PARAMETER(the_list, 'copies', TOOL_INT.TEXT_PARAMETER, '2');
  TOOL_INT.ADD_PARAMETER(the_list, 'my_param', TOOL_INT.TEXT_PARAMETER, '67');
  TOOL_INT.ADD_PARAMETER(the_list, 'Q_1', TOOL_INT.DATA_PARAMETER, 'query0');
END;
```

When you invoke another product from Graphics, you can effectively share your login session with the invoked product. To do this, you must pass the login information you supplied with Graphics to the other product in the form of a parameter. This can also be handy for suppressing the Connect dialog box.
Forms, Reports, and Oracle Book all use the USERID executable option to establish a database connection. Graphics stores its login information in the form of properties for the application. You can use the property built-in function OG_GET_ATTR to retrieve this information, and then include it in a parameter list as a text parameter.

The sample procedure below shows how to pass a remote database connect string from Graphics to Reports:

```sql
PROCEDURE run_form IS
    rept_list  tool_int.paramlist;
    uname      VARCHAR2;
    pword      VARCHAR2;
    cnxn       VARCHAR2;
BEGIN
    /*create the parameter list*/
    rept_list:=TOOL_INT.CREATE_PARAMETER_LIST('list1');

    /*find OG login information*/
    uname:=OG_GET_AP_USERNAME;
    pword:=OG_GET_AP_PASSWORD;
    cnxn:=OG_GET_AP_CONNECTION;

    /*add OG login info to parameter list*/
    TOOL_INT.ADD_PARAMETER (form_list, 'userid',
        TOOL_INT.TEXT_PARAMETER, uname || '/' ||
        pword || '@' || cnxn);

    /*pass Reports the parameter list with login info*/
    TOOL_INT.RUN_PRODUCT(TOOL_INT.REPORTS, 'inventory.rpt',
        TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME, rept_list);
END;
```

Invoking the Product

After you have created and populated the parameter list, you invoke the other Oracle products using the TOOL_INT.RUN_PRODUCT procedure. This procedure requires the following arguments:

- the name of the product you are invoking
- the form/report/document of the product
- the communication mode in which to invoke the product
- the product executable mode to use
- the name of the parameter list to pass to the product
Forms, Reports, and Oracle Book all use the USERID executable option to establish a database connection. Graphics stores its login information in the form of properties for the application. You can use the property built-in function OG_GET_ATTR to retrieve this information, and then include it in a parameter list as a text parameter.

The sample procedure below shows how to pass a remote database connect string from Graphics to Reports:

```sql
PROCEDURE run_form IS
    rept_list  tool_int.paramlist;
    uname      VARCHAR2;
    pword      VARCHAR2;
    cnxn       VARCHAR2;
BEGIN
    /*create the parameter list*/
    rept_list:=TOOL_INT.CREATE_PARAMETER_LIST('list1');
    /*find OG login information*/
    uname:=OG_GET_AP_USERNAME;
    pword:=OG_GET_AP_PASSWORD;
    cnxn:=OG_GET_AP_CONNECTION;
    /*add OG login info to parameter list*/
    TOOL_INT.ADD_PARAMETER (form_list, 'userid', TOOL_INT.TEXT_PARAMETER, uname || '/' || pword || '@' || cnxn);
    /*pass Reports the parameter list with login info*/
    TOOL_INT.RUN_PRODUCT(TOOL_INT.REPORTS, 'inventory.rpt', TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME, rept_list);
END;
```

After you have created and populated the parameter list, you invoke the other Oracle products using the TOOL_INT.RUN_PRODUCT procedure. This procedure requires the following arguments:

- the name of the product you are invoking
- the form/report/document of the product
- the communication mode in which to invoke the product
- the product executable mode to use
- the name of the parameter list to pass to the product
Communication Modes

There are two communication modes in which Graphics can invoke another Oracle product:

- synchronous
- asynchronous

Synchronous

When Graphics invokes another product *synchronously*, the execution of the Graphics display is suspended until the invoked product has finished executing.

**Note:** To pass data to another product, you *must* invoke that product synchronously.

You may want to invoke another product in this way for the following reasons:

- To require the user to enter values in Forms before continuing.
- To pass query data to Reports and run a report based on that query data.

Asynchronous

When Graphics invokes another product *asynchronously*, the execution of the Graphics display continues normally, following the PL/SQL procedure call that invokes the other product. Thus, Graphics and the called product run concurrently.

You cannot pass data to another product if you invoke it asynchronously.

You may want to invoke another product in this way for the following reasons:

- To generate a Reports report to another device, such as a printer.
- To allow the user to view an Oracle Book document during the display execution.

**Note:** If you want to pass a data parameter from Graphics to another product, you must invoke that product in the *synchronous* communication mode.

Execution Modes

There are two execution modes in which Graphics can invoke another Oracle product:

- runtime
- batch
There are two communication modes in which Graphics can invoke another Oracle product:

- **synchronous**
- **asynchronous**

### Synchronous
When Graphics invokes another product synchronously, the execution of the Graphics display is suspended until the invoked product has finished executing.

**Note:** To pass data to another product, you must invoke that product synchronously.

You may want to invoke another product in this way for the following reasons:

- To require the user to enter values in Forms before continuing.
- To pass query data to Reports and run a report based on that query data.

### Asynchronous
When Graphics invokes another product asynchronously, the execution of the Graphics display continues normally, following the PL/SQL procedure call that invokes the other product. Thus, Graphics and the called product run concurrently.

You cannot pass data to another product if you invoke it asynchronously.

You may want to invoke another product in this way for the following reasons:

- To generate a Reports report to another device, such as a printer.
- To allow the user to view an Oracle Book document during the display execution.

**Note:** If you want to pass a data parameter from Graphics to another product, you must invoke that product in the synchronous communication mode.
The behavior in each of these execution modes differs for each product invoked by Graphics. In addition, some Oracle products may not support a particular execution mode. For more information, see the documentation for the product you wish to invoke.

**Invoking Forms from Graphics**

You can invoke Forms from Graphics to display a form containing query data from the display. Remember, when passing data from one product to another, you must invoke the product in synchronous mode.

**Example**

The following procedure invokes a Forms Runtime application and supplies it with query data from the display.

```sql
PROCEDURE open_form (the_form IN VARCHAR2) IS
    form_list  tool_int.paramlist;
    uname      VARCHAR2;
    pword      VARCHAR2;
    cnxn       VARCHAR2;
BEGIN
    /*create the parameter list*/
    form_list:=TOOL_INT.CREATE_PARAMETER_LIST('list2');
    uname:=OG_GET_AP_USERNAME;
    pword:=OG_GET_AP_PASSWORD;
    cnxn:=OG_GET_AP_CONNECTION;
    /*Add a parameter to the parameter list*/
    TOOL_INT.ADD_PARAMETER (form_list, 'userid',
        TOOL_INT.TEXT_PARAMETER, uname || '/' ||
        pword || '@' || cnxn);
    /*pass Forms the parameter list*/
    TOOL_INT.RUN_PRODUCT(TOOL_INT.FORMS, 'the_form',
        TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME, form_list);
END;
```

**Invoking Reports from Graphics**

You can invoke Reports from Graphics and have Reports generate and run a report based on the current display.

For example, you could define a trigger that invokes Reports in batch mode and prints a predefined report. Alternatively, you could invoke Reports runtime to display a report.

**Example**

The following example shows a procedure that creates a parameter list to pass to Reports, and then calls Reports to run the `inventory.rep` report. In addition, the procedure includes the USERID executable argument text parameter to suppress the Reports Connect dialog box.
The behavior in each of these execution modes differs for each product invoked by Graphics. In addition, some Oracle products may not support a particular execution mode. For more information, see the documentation for the product you wish to invoke.

You can invoke Forms from Graphics to display a form containing query data from the display. Remember, when passing data from one product to another, you must invoke the product in synchronous mode.

The following procedure invokes a Forms Runtime application and supplies it with query data from the display.

```sql
PROCEDURE open_form (the_form IN VARCHAR2) IS
  form_list  tool_int.paramlist;
  uname      VARCHAR2;
  pword      VARCHAR2;
  cnxn       VARCHAR2;
BEGIN
  /*create the parameter list*/
  form_list:=TOOL_INT.CREATE_PARAMETER_LIST('list2');

  uname:=OG_GET_AP_USERNAME;
  pword:=OG_GET_AP_PASSWORD;
  cnxn:=OG_GET_AP_CONNECTION;

  /*Add a parameter to the parameter list*/
  TOOL_INT.ADD_PARAMETER (form_list, 'userid',
                           TOOL_INT.TEXT_PARAMETER, uname || '/' ||
                           pword || '@' || cnxn);

  /*pass Forms the parameter list*/
  TOOL_INT.RUN_PRODUCT(TOOL_INT.FORMS, 'the_form',
                        TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME, form_list);
END;
```

You can invoke Reports from Graphics and have Reports generate and run a report based on the current display. For example, you could define a trigger that invokes Reports in batch mode and prints a predefined report. Alternatively, you could invoke Reports runtime to display a report.

The following example shows a procedure that creates a parameter list to pass to Reports, and then calls Reports to run the `inventory.rep` report.

In addition, the procedure includes the USERID executable argument text parameter to suppress the Reports Connect dialog box.
PROCEDURE print_inv_rep IS
    rep_list  tool_int.paramlist;
BEGIN
   /*create a parameter list*/
    rep_list:=TOOL_INT.CREATE_PARAMETER_LIST('list2');

   /*add the USERID executable option to the list*/
    TOOL_INT.ADD_PARAMETER (rep_list, 'userid',
        TOOL_INT.TEXT_PARAMETER, 'scott/tiger');

   /*add the query data parameter to the list*/
    TOOL_INT.ADD_PARAMETER (rep_list, 'Q_inv',
        TOOL_INT.DATA_PARAMETER, 'query_inv');

   /*call Reports batch mode to print the report*/
    TOOL_INT.RUN_PRODUCT(TOOL_INT.REPORTS, 'inventory.rep',
        TOOL_INT.SYNCHRONOUS, TOOL_INT.BATCH,
        TOOL_INT.FILESYSTEM, rep_list);
END;

The execution of the Graphics display is suspended until Reports has finished printing the report.

For more information about Reports executable options, see your Reports documentation.

Invoking Oracle Book from Graphics

You can invoke an instance of Oracle Book runtime to display an Oracle Book document from your Graphics display.

If your Oracle Book document has hypertext targets in it, you can specify that the document be opened at a specific target location using the TARGET executable argument.

Example

The following button procedure opens an Oracle Book document named help.obd at the target name “start”:

PROCEDURE OGBUTTONPROC0(buttonobj IN og_object, hitobj IN og_object, win IN og_window, eventinfo IN og_event)IS
    book_list  tool_int.paramlist;
BEGIN
   /*create a parameter list*/
    book_list:=TOOL_INT.CREATE_PARAMETER_LIST('list1');

   /*add the target executable option to the list*/
    TOOL_INT.ADD_PARAMETER (book_list, 'target',
        TOOL_INT.TEXT_PARAMETER, 'start');
PROCEDURE print_inv_rep IS
rep_list  tool_int.paramlist;
BEGIN
/*create a parameter list*/
rep_list:=TOOL_INT.CREATE_PARAMETER_LIST('list2');
/*add the USERID executable option to the list*/
TOOL_INT.ADD_PARAMETER (rep_list, 'userid',
TOOL_INT.TEXT_PARAMETER, 'scott/tiger');
/*add the query data parameter to the list*/
TOOL_INT.ADD_PARAMETER (rep_list, 'Q_inv',
TOOL_INT.DATA_PARAMETER, 'query_inv');
/*call Reports batch mode to print the report*/
TOOL_INT.RUN_PRODUCT(TOOL_INT.REPORTS, 'inventory.rep',
TOOL_INT.SYNCHRONOUS, TOOL_INT.BATCH,
TOOL_INT.FILESYSTEM, rep_list);
END;

The execution of the Graphics display is suspended until Reports has
finished printing the report.

For more information about Reports executable options, see your
Reports documentation.

You can invoke an instance of Oracle Book runtime to display an Oracle
Book document from your Graphics display.

If your Oracle Book document has hypertext targets in it, you can
specify that the document be opened at a specific target location using
the TARGET executable argument.

The following button procedure opens an Oracle Book document named
help.obd at the target name "start":
PROCEDURE OGBUTTONPROC0(buttonobj IN og_object, hitobj
IN og_object, win IN og_window, eventinfo IN og_event)IS
book_list  tool_int.paramlist;
BEGIN
/*create a parameter list*/
book_list:=TOOL_INT.CREATE_PARAMETER_LIST('list1');
/*add the target executable option to the list*/
TOOL_INT.ADD_PARAMETER (book_list, 'target',
TOOL_INT.TEXT_PARAMETER, 'start');
/*open Oracle Book at the target name specified above*/
TOOL_INT.RUN_PRODUCT(TOOL_INT.BOOK, 'help.obd',
    TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME,
    TOOL_INT.FILESYSTEM, book_list);
END;

See your Oracle Book documentation for other executable arguments you can pass to Oracle Book Runtime when you invoke it.
/* open Oracle Book at the target name specified above */
TOOL_INT.RUN_PRODUCT(TOOL_INT.BOOK, 'help.obd', TOOL_INT.ASYNCHRONOUS, TOOL_INT.RUNTIME, TOOL_INT.FILESYSTEM, book_list);
END;
See your Oracle Book documentation for other executable arguments you can pass to Oracle Book Runtime when you invoke it.
Glossary

Ada  A high-level programming language developed by the U.S. Department of Defense for use in embedded computer systems.

active window/view  A window or view that can receive user input.

alert  A modal window that notifies you of a condition that occurred because of your last action. You must respond to an alert. See also: message box.

alignment  The way in which data is positioned in a field. It may be positioned to the left, right, center, flush/left, flush/right, or flush/center of the defined width of a field.

alias  A secondary name used in a SQL statement to reference a table, view, or column.

ancestor group  A group that owns another group, either directly or indirectly.

anchor  A layout object used to fix a spot on one object to a spot on another object, ensuring the position of the first object in relation to the second object.

anonymous block  A PL/SQL program unit that has no name and does not require the explicit presence of the BEGIN and END keywords to enclose the executable statements. Since they are unnamed, anonymous blocks cannot be referenced by other program units.

antecedent  An object that appears in another object’s group tree at any position above that other object. Every object is an antecedent of its descendants.

application  One or more program modules used to achieve a specific result. Applications can be nested within other applications. For example, an application to control a company’s inventory could consist of various Forms applications for input of data, and various Graphics applications to display the data visually.

argument  1. An optional expression passed to a program unit for the program unit to operate on. See also: parameter. 2. Clauses containing keywords and their specified values.

A high-level programming language developed by the U.S. Department of Defense for use in embedded computer systems.

active window/view
A window or view that can receive user input.

alert
A modal window that notifies you of a condition that occurred because of your last action. You must respond to an alert. See also: message box.

alignment
The way in which data is positioned in a field. It may be positioned to the left, right, center, flush/left, flush/right, or flush/center of the defined width of a field.

alias
A secondary name used in a SQL statement to reference a table, view, or column.

ancestor group
A group that owns another group, either directly or indirectly.

anchor
A layout object used to fix a spot on one object to a spot on another object, ensuring the position of the first object in relation to the second object.

anonymous block
A PL/SQL program unit that has no name and does not require the explicit presence of the BEGIN and END keywords to enclose the executable statements. Since they are unnamed, anonymous blocks cannot be referenced by other program units.

antecedent
An object that appears in another object's group tree at any position above that other object. Every object is an antecedent of its descendants.

application
One or more program modules used to achieve a specific result. Applications can be nested within other applications. For example, an application to control a company's inventory could consist of various Forms applications for input of data, and various Graphics applications to display the data visually.

argument
1. An optional expression passed to a program unit for the program unit to operate on. See also: parameter.

2. Clauses containing keywords and their specified values.

ASCII
attached library  A read only collection of PL/SQL subprograms, usually provided with an application to supply necessary PL/SQL source.

attribute  See property.

attribute mask  In Graphics: The first element in an attribute descriptor. It is a 32–bit mask that informs the toolkit of the attributes that will be set or retrieved by a subsequent function call. Only those elements that have their corresponding bits set in the attribute mask are set or retrieved.

axis chart  A type of chart in which the data is plotted according to where it lies along the chart’s edges, called axes. An axis chart may contain two or three axes, each of which may be one of the following types: discrete, continuous, or date.

background element  The white part of a pattern. You apply color to the background element using the Fill Color palette.

background process  A non–interactive process that runs in an operating system environment and performs some service or action. Certain Oracle products use background processes for different tasks, such as performing and coordinating tasks on behalf of concurrent users of the database, processing and delivering electronic messages, and managing printing services.

baseline value  The value used as the starting point for plotting fields on the continuous axis of a chart.

Batch  The executable that executes an application in batch mode, without using a windowing system.

batch compile  To translate, at once, several PL/SQL program units into a machine executable form.

bind reference  A reference to a parameter used to replace a single literal value (e.g., a character string, number, or date) appearing anywhere in a PL/SQL construct or a SQL SELECT statement. For a bind reference, you must precede the parameter name with a colon (:). See also: lexical reference, parameter.

bind variable  A global variable you can create and use outside of a PL/SQL package.

block  1. The basic program unit in PL/SQL, defined by the keywords DECLARE, BEGIN, EXCEPTION, and END. 2. In Forms, a logical container that relates to a database table or view.

body  1. Contains the same information as the program unit specification, and also includes the actual implementation of the subprogram (i.e., its sequence of actions). In most cases, only a body must be defined for a subprogram. 2. In Reports, a region that contains the bulk of the report layout (text, graphics, data, and computations). The report body appears between the report header and trailer pages.

boilerplate  Text and/or graphics that appear in a report every time it is run. In some products this is called “constant” text or graphics.

border  A user–defined outline around a layout object, usually used to highlight its contents. Borders are drawn such that half the width of the border lies outside the edge of the object, and half inside the edge. See also: edge.
attached library
A read only collection of PL/SQL subprograms, usually provided with an application to supply necessary PL/SQL source.

attribute
See property.

attribute mask
In Graphics: The first element in an attribute descriptor. It is a 32–bit mask that informs the toolkit of the attributes that will be set or retrieved by a subsequent function call. Only those elements that have their corresponding bits set in the attribute mask are set or retrieved.

axis chart
A type of chart in which the data is plotted according to where it lies along the chart's edges, called axes. An axis chart may contain two or three axes, each of which may be one of the following types: discrete, continuous, or date.

background element
The white part of a pattern. You apply color to the background element using the Fill Color palette.

background process
A non–interactive process that runs in an operating system environment and performs some service or action. Certain Oracle products use background processes for different tasks, such as performing and coordinating tasks on behalf of concurrent users of the database, processing and delivering electronic messages, and managing printing services.

baseline value
The value used as the starting point for plotting fields on the continuous axis of a chart.

Batch
The executable that executes an application in batch mode, without using a windowing system.

batch compile
To translate, at once, several PL/SQL program units into a machine executable form.

bind reference
A reference to a parameter used to replace a single literal value (e.g., a character string, number, or date) appearing anywhere in a PL/SQL construct or a SQL SELECT statement. For a bind reference, you must precede the parameter name with a colon (:).

See also: lexical reference, parameter.

bind variable
A global variable you can create and use outside of a PL/SQL package.

block
1. The basic program unit in PL/SQL, defined by the keywords DECLARE, BEGIN, EXCEPTION, and END. 2. In Forms, a logical container that relates to a database table or view.

body
1. Contains the same information as the program unit specification, and also includes the actual implementation of the subprogram (i.e., its sequence of actions). In most cases, only a body must be defined for a subprogram. 2. In Reports, a region that contains the bulk of the report layout (text, graphics, data, and computations). The report body appears between the report header and trailer pages.

boilerplate
Text and/or graphics that appear in a report every time it is run. In some products this is called "constant" text or graphics.

border
A user–defined outline around a layout object, usually used to highlight its contents. Borders are drawn such that half the width of the border lies outside the edge of the object, and half inside the edge. See also: edge.
bounding box  See inner bounding box, outer bounding box.

break chart  An axis chart that displays two categories of data along one axis.

break column  A query column assigned to a break group.

break group  A group used to break or categorize report data. All data in query columns assigned to a break group appear above the data in columns belonging to other groups. Each record fetched for a break group appears only once, while multiple related records in other groups can be displayed.

break report  A report that contains categories that break the report data into sets.

breakpoint  A debug action you can place at any source in a PL/SQL subprogram that interrupts execution of the program unit and passes control to the PL/SQL Interpreter. Breakpoints allow you to incrementally execute a subprogram and search for runtime errors.

built-ins  A collection of extensions to the PL/SQL language that are designed to be used with Developer/2000. Built-ins include datatypes, procedures, functions, exceptions, and constants.

buffer  A temporary storage area for data during the transfer of that data between the computer and a peripheral, or between parts of a computer. A buffer prevents loss of information due to differences in the speed or timing of the transfer and speeds up certain functions.

button  1. An interface element used to select an action item, display a dialog box, or acknowledge the current condition. 2. An object in an application that has a PL/SQL procedure associated with it. The procedure is generally executed upon a user mouse event.

button procedure  A PL/SQL procedure associated with an application object, that is executed when a specific mouse event occurs on the item at runtime (e.g. clicking the button). It does this by establishing a hierarchy for the fetched data containing a break group.

cache (memory)  A temporary storage place for database data that is currently being accessed or changed by users, or for data that Oracle Server requires to support users. See also: buffer. The terms are often used interchangeably.

call interface  An interface that enables you to invoke Developer/2000 component executables from other Oracle products, or a 3GL program.

call stack  A chain of subprogram calls, from the initial entry point down to the currently executing subprogram. Each subprogram call is represented by a frame on the downward-growing call stack, in which newly entered subprograms are added to the bottom of the stack.

canvas  See editor.
bounding box

See inner bounding box, outer bounding box.

break chart
An axis chart that displays two categories of data along one axis.

break column
A query column assigned to a break group.

break group
A group used to break or categorize report data. All data in query columns assigned to a break group appear above the data in columns belonging to other groups. Each record fetched for a break group appears only once, while multiple related records in other groups can be displayed.

break report
A report that contains categories that break the report data into sets.

breakpoint
A debug action you can place at any source in a PL/SQL subprogram that interrupts execution of the program unit and passes control to the PL/SQL Interpreter. Breakpoints allow you to incrementally execute a subprogram and search for runtime errors.

built-ins
A collection of extensions to the PL/SQL language that are designed to be used with Developer/2000. Built-ins include datatypes, procedures, functions, exceptions, and constants.

buffer
A temporary storage area for data during the transfer of that data between the computer and a peripheral, or between parts of a computer. A buffer prevents loss of information due to differences in the speed or timing of the transfer and speeds up certain functions.

button
1. An interface element used to select an action item, display a dialog box, or acknowledge the current condition. 2. An object in an application that has a PL/SQL procedure associated with it. The procedure is generally executed upon a user mouse event.

button procedure
A PL/SQL procedure associated with an application object, that is executed when a specific mouse event occurs on the item at runtime (e.g. clicking the button). It does this by establishing a hierarchy for the fetched data containing a break group.

cache (memory)
A temporary storage place for database data that is currently being accessed or changed by users, or for data that Oracle Server requires to support users. See also: buffer. The terms are often used interchangeably.

call interface
An interface that enables you to invoke Developer/2000 component executables from other Oracle products, or a 3GL program.

call stack
A chain of subprogram calls, from the initial entry point down to the currently executing subprogram. Each subprogram call is represented by a frame on the downward-growing call stack, in which newly entered subprograms are added to the bottom of the stack.

canvas
See editor.
canvas view  In Forms, the background object on which you place the interface objects that users will interact with in the runtime application. A canvas view is usually associated with a window.

category  A label component that allows differentiation between information on the basis of non–hierarchical delineation or compartments rather than sensitivity levels. With classification, they are used as a means of restricting access to information.

category axis  See discrete axis.

cGM  Acronym for Computer Graphics Metafile. A file format recognized internationally and used to store and transport object–oriented graphics from one system or program to another.

CHAR datatype  An Oracle datatype provided for ANSI/ISO compatibility. A CHAR column is a fixed–length column and can contain any printable characters, such as A, 3, &, or blanks, and can have from 1 to 255 characters or can be null.

chart element  A bar, line, symbol, or other graphical object that represents a single value for a field.

chart item  A bordered rectangle of any size used to embed Graphics charts in a form or report.

chart template  A collection of attributes and properties that defines the format of a chart.

check box  A interface element, appearing as a small square, that a user can toggle on or off.

checksum  An error–checking algorithm.

child  An object that is a member of a group, and is immediately below that group object in its group tree. The objects that compose a group object are children of that group. Every object is a child of its parent.

child query  See detail query.

classification  A hierarchical level that denotes the sensitivity of the information that is labeled. Classifications are used as a means of restricting access to information.

clause  A part of a SQL statement that does not constitute the full statement; for example, a “WHERE clause.”

client  A user, software application, or computer that requests the services, data, or processing of another application or computer (the “server”). In a two–task environment, the client is the user process. In a network environment, the client is the local user process and the server may be local or remote.

client–server architecture  Separation of processing between two CPUs, one acting as the server that provides services in a transaction and the other as the client in the transaction, requesting and receiving services. All responsibilities of shared data management can be processed by the computer running the database management system while the workstations running the database application concentrate on the interpretation and display of data.
In Forms, the background object on which you place the interface objects that users will interact with in the runtime application. A canvas view is usually associated with a window.

**category**
A label component that allows differentiation between information on the basis of non–hierarchical delineation or compartments rather than sensitivity levels. With classification, they are used as a means of restricting access to information.

**category axis**
See discrete axis.

**CGM**
Acronym for Computer Graphics Metafile. A file format recognized internationally and used to store and transport object–oriented graphics from one system or program to another.

**CHAR datatype**
An Oracle datatype provided for ANSI/ISO compatibility. A CHAR column is a fixed–length column and can contain any printable characters, such as A, 3, &, or blanks, and can have from 1 to 255 characters or can be null.

**chart element**
A bar, line, symbol, or other graphical object that represents a single value for a field.

**chart item**
A bordered rectangle of any size used to embed Graphics charts in a form or report.

**chart template**
A collection of attributes and properties that defines the format of a chart.

**check box**
A interface element, appearing as a small square, that a user can toggle on or off.

**checksum**
An error–checking algorithm.

**child**
An object that is a member of a group, and is immediately below that group object in its group tree. The objects that compose a group object are children of that group. Every object is a child of its parent.

**child query**
See detail query.

**classification**
A hierarchical level that denotes the sensitivity of the information that is labeled. Classifications are used as a means of restricting access to information.

**clause**
A part of a SQL statement that does not constitute the full statement; for example, a "WHERE clause."

**client**
A user, software application, or computer that requests the services, data, or processing of another application or computer (the "server"). In a two–task environment, the client is the user process. In a network environment, the client is the local user process and the server may be local or remote.

**client–server architecture**
Separation of processing between two CPUs, one acting as the server that provides services in a transaction and the other as the client in the transaction, requesting and receiving services. All responsibilities of shared data management can be processed by the computer running the database management system while the workstations running the database application concentrate on the interpretation and display of data.
Clipboard  A memory buffer. An object remains on the Clipboard until you cut or copy another object, or until you quit the application.

CMDFILE  An argument that allows you to specify a file that contains a set of command line arguments for Developer/2000 executables.

color palette  Contains all the colors available to the windowing system, the drawing surface, or a window and its views.

column  1. A vertical space in a database table that represents a particular domain of data. A column has a column name (e.g., ENAME) and a specific datatype (e.g., CHAR). For example, in a table of employee information, all of the employees’ names would constitute one column. A record group column represents a database column.  
2. In Reports, a data model object created automatically for each column expression in a query’s SELECT list, or created manually to perform summaries, formulas, or act as a placeholder.

command line  An operating–system command line. Most Oracle products can be invoked from a command line using a number of executable arguments.

comment  A language construct for the inclusion of explanatory text in a program, the execution of which remains unaffected.

commit  To make changes to data in the database permanent. Before changes are stored, both the old and new data exist so that changes can be stored or the data can be restored to its prior state.

compile  To translate a source program into a binary, executable format. PL/SQL program units must be compiled before they can be executed.

concurrency  The simultaneous access of the same data by multiple users executing different database applications or the same application. In database software concurrency requires additional logic to ensure that all users see consistent data and that all changes are made in the proper order.

condition  An expression whose value is either true or false, such as X> 100. For example, you can set a condition in a query when you want the query to display only those rows that evaluate as true in your expression.

confine mode  Layout editor mode that prevents child objects from being moved or resized on or outside their parent objects.

connect  To log on to a database. You must connect if you want to create or modify queries or access an application stored in a database.

connect string  The set of parameters, including a protocol, that SQL*Net uses to connect to a specific Oracle instance on the network.

constraint  A rule or restriction concerning a piece of data (such as a NOT NULL restriction on a column) that is enforced at the database level, rather than the object or application level.

construct  A PL/SQL code structure. There are two types of constructs: non–value constructs do not return a value; value constructs return a value.

context area  A work area in which Oracle Server stores the current SQL statement, and if the statement is a query, the result’s column headings and one row of the result. Also referred to as cursor.
Clipboard

A memory buffer. An object remains on the Clipboard until you cut or copy another object, or until you quit the application.

CMDFILE

An argument that allows you to specify a file that contains a set of command line arguments for Developer/2000 executables.

color palette

Contains all the colors available to the windowing system, the drawing surface, or a window and its views.

column

1. A vertical space in a database table that represents a particular domain of data. A column has a column name (e.g., ENAME) and a specific datatype (e.g., CHAR). For example, in a table of employee information, all of the employees’ names would constitute one column. A record group column represents a database column.

2. In Reports, a data model object created automatically for each column expression in a query’s SELECT list, or created manually to perform summaries, formulas, or act as a placeholder.

command line

An operating–system command line. Most Oracle products can be invoked from a command line using a number of executable arguments.

comment

A language construct for the inclusion of explanatory text in a program, the execution of which remains unaffected.

commit

To make changes to data in the database permanent. Before changes are stored, both the old and new data exist so that changes can be stored or the data can be restored to its prior state.

compile

To translate a source program into a binary, executable format. PL/SQL program units must be compiled before they can be executed.

concurrency

The simultaneous access of the same data by multiple users executing different database applications or the same application. In database software concurrency requires additional logic to ensure that all users see consistent data and that all changes are made in the proper order.

condition

An expression whose value is either true or false, such as X> 100. For example, you can set a condition in a query when you want the query to display only those rows that evaluate as true in your expression.

confine mode

Layout editor mode that prevents child objects from being moved or resized on or outside their parent objects.

connect

To log on to a database. You must connect if you want to create or modify queries or access an application stored in a database.

connect string

The set of parameters, including a protocol, that SQL*Net uses to connect to a specific Oracle instance on the network.

constraint

A rule or restriction concerning a piece of data (such as a NOT NULL restriction on a column) that is enforced at the database level, rather than the object or application level.

construct

A PL/SQL code structure. There are two types of constructs: non–value constructs do not return a value; value constructs return a value.

context area

A work area in which Oracle Server stores the current SQL statement, and if the statement is a query, the result’s column headings and one row of the result. Also referred to as cursor.
**continuous axis** An axis that plots values that begin with one value and continue until they reach another value. Thus, these values are mathematically related. Dependent data is usually plotted along this axis, also called the value axis.

**control point** The four corner points of the virtual rectangle that constitutes an object’s ideal shape. When you select an object on the layout, its control points appear. Depending on the object selected, you may be able to modify it by dragging a control point.

**coordinate plane** The two-dimensional grid on which drawing takes place.

**coordinate system** The type of x, y units used to place and size objects in the Layout Editor. The coordinate systems available are 1) character, which uses the platform’s native character cell size, and 2) real, which can be in units of pixels, centimeters, inches, or points.

**coordinates (x, y)** The location points for objects appearing in the layout. x and y coordinates determine the horizontal (x) and vertical (y) placement of objects.

**copy** To store a replica of a selected object on the Clipboard, so that it may be pasted elsewhere in an editor if desired.

**cut** To delete one or more objects and store them in the Clipboard, so that they may be pasted elsewhere in an editor, if desired.

**damage** A view is damaged when it is totally or partially occluded by another view.

**data dictionary** A set of tables and views owned by the database administrator. It is a central source of information for Oracle Server and other relational databases.

**data link** A data model object used to define a master/detail (parent–child) relationship between a group and a query.

**data model** A relational model that defines what data should be fetched from the database, what values should be computed, and how data should be ordered in a report. Reports objects that define the data model are queries, groups, columns, parameters, and links.

**database** A set of dictionary tables and user tables that are treated as a unit.

**datatyp** A standard form of data. Some common Oracle datatypes are CHAR, VARCHAR2, DATE, NUMBER, LONG, RAW, and LONG RAW.

**DATE** An Oracle Server datatype. A date column may contain a date and time between January 1, 4712 BC and December 31, 4712 AD.

**date axis** An axis that plots date values that begin at one date and continue until they reach another date.

**debug** To detect, diagnose, and eliminate runtime errors in programs.

**debug action** A user–specified event placed in a PL/SQL subprogram, usually for the purpose of troubleshooting runtime errors. See also: breakpoint, debug trigger.
An axis that plots values that begin with one value and continue until they reach another value. Thus, these values are mathematically related. Dependent data is usually plotted along this axis, also called the value axis.

The four corner points of the virtual rectangle that constitutes an object's ideal shape. When you select an object on the layout, its control points appear. Depending on the object selected, you may be able to modify it by dragging a control point.

The two-dimensional grid on which drawing takes place.

The type of x, y units used to place and size objects in the Layout Editor. The coordinate systems available are 1) character, which uses the platform's native character cell size, and 2) real, which can be in units of pixels, centimeters, inches, or points.

The location points for objects appearing in the layout. x and y coordinates determine the horizontal (x) and vertical (y) placement of objects.

To store a replica of a selected object on the Clipboard, so that it may be pasted elsewhere in an editor if desired.

Group that owns two or more other groups and performs a mathematical cross product (i.e., correlates values between them).

See matrix cell.

1. A small icon representing the position of the mouse. The shape of the cursor varies, depending on the selected tool. 2. An internal pointer to data retrieved by a query. A cursor points only to one row of data at a time; however, you can use built-in subprograms to move the cursor to any row in the data set.

To delete one or more objects and store them in the Clipboard, so that they may be pasted elsewhere in an editor, if desired.

A view is damaged when it is totally or partially occluded by another view.

A set of tables and views owned by the database administrator. It is a central source of information for Oracle Server and other relational databases.

A data model object used to define a master/detail (parent–child) relationship between a group and a query.

A relational model that defines what data should be fetched from the database, what values should be computed, and how data should be ordered in a report. Reports objects that define the data model are queries, groups, columns, parameters, and links.

A set of dictionary tables and user tables that are treated as a unit.

A standard form of data. Some common Oracle datatypes are CHAR, VARCHAR2, DATE, NUMBER, LONG, RAW, and LONG RAW.

An Oracle Server datatype. A date column may contain a date and time between January 1, 4712 BC and December 31, 4712 AD.

An axis that plots date values that begin at one date and continue until they reach another date.

To detect, diagnose, and eliminate runtime errors in programs.

A user–specified event placed in a PL/SQL subprogram, usually for the purpose of troubleshooting runtime errors. See also: breakpoint, debug trigger.
**debug trigger**  A conditional debug action you can place at any source line in a PL/SQL subprogram. If the specified condition is met, the specified action is taken before execution of the subprogram resumes.

**Debugger**  The Procedure Builder functionality within a Developer/2000 component used for testing and debugging.

**dependent data**  A piece of data that depends on other data for its value. For example, an employee’s years of service may have the value 10, which is dependent upon the value of the employee’s name. Also called value data.

**default**  A value supplied by the system when a user does not specify a required command parameter or attribute.

**descendant**  An object that appears in another object’s group tree at any position below that other object. Every object is a descendant of its antecedents.

**Designer**  The interface used to design different components of a Developer/2000 application.

**detail query**  When defining a master/detail report, the detail query retrieves all related records for each record retrieved by the master, or parent, query.

**dialog box**  A partial screen or window that prompts you to enter information necessary to complete an operation.

**DIANA**  An intermediate representation of a program unit, generated by the compiler. The DIANA includes all of the syntactic and semantic information for a compiled program unit. In order to compile a reference to a program unit, the compiler must have access to the referenced program unit’s DIANA. For example, if program unit A calls program unit B, the DIANA for B must be available before A can be compiled.

**disabled**  An interface element state that means a menu item, button, etc., cannot be used in the current context; i.e., it does not respond to keyboard or mouse input.

**discrete axis**  An axis that plots distinct values at fixed intervals. These values are not mathematically related. Independent data is usually plotted along this axis, also called the category axis.

**display**  A Graphics document containing any combination of charts, images, text objects and shapes.

**display item**  In Forms, an item that stores or displays information that must be fetched or assigned programmatically with PL/SQL.

**downward–growing**  As it pertains to the call stack, in which newly entered subprograms are added to the bottom of the stack. The earliest frame, corresponding to the initial program entry point, is at the top of the stack, while the latest frame, associated with the most deeply nested subprogram call, is at the bottom of the stack.

**drag**  Press and hold down a mouse button while you slide the mouse pointer to a particular location in a window.

**duplicate**  An option that allows you to copy objects directly on the layout without affecting the contents of the Clipboard. See also: copy.
debug trigger
A conditional debug action you can place at any source line in a PL/SQL subprogram. If the specified condition is met, the specified action is taken before execution of the subprogram resumes.

Debugger
The Procedure Builder functionality within a Developer/2000 component used for testing and debugging.

dependent data
A piece of data that depends on other data for its value. For example, an employee's years of service may have the value 10, which is dependent upon the value of the employee's name. Also called value data.

default
A value supplied by the system when a user does not specify a required command parameter or attribute.

descendant
An object that appears in another object's group tree at any position below that other object. Every object is a descendant of its antecedents.

designer
The interface used to design different components of a Developer/2000 application.

detail query
When defining a master/detail report, the detail query retrieves all related records for each record retrieved by the master, or parent, query.

dialog box
A partial screen or window that prompts you to enter information necessary to complete an operation.

dIANA
An intermediate representation of a program unit, generated by the compiler. The DIANA includes all of the syntactic and semantic information for a compiled program unit. In order to compile a reference to a program unit, the compiler must have access to the referenced program unit's DIANA. For example, if program unit A calls program unit B, the DIANA for B must be available before A can be compiled.

disabled
An interface element state that means a menu item, button, etc., cannot be used in the current context; i.e., it does not respond to keyboard or mouse input.

discrete axis
An axis that plots distinct values at fixed intervals. These values are not mathematically related. Independent data is usually plotted along this axis, also called the category axis.

display
A Graphics document containing any combination of charts, images, text objects and shapes.

display item
In Forms, an item that stores or displays information that must be fetched or assigned programmatically with PL/SQL.

downward–growing
As it pertains to the call stack, in which newly entered subprograms are added to the bottom of the stack. The earliest frame, corresponding to the initial program entry point, is at the top of the stack, while the latest frame, associated with the most deeply nested subprogram call, is at the bottom of the stack.

drag
Press and hold down a mouse button while you slide the mouse pointer to a particular location in a window.
duplicate
An option that allows you to copy objects directly on the layout without affecting the contents of the Clipboard. See also: copy.
EBCDIC  Acronym for Extended Binary–Coded Decimal Interchange Code.

dictionary  The boundary of an object.

Edge/Fill/Text Display  A display box on the Layout editor tool palette that shows the current settings for the edge, fill, and text patterns and colors.

definition  A work area in which you perform a specific set of tasks, such as creating a program unit or designing an application.

element  A thing of significance about which you may need to record information. Elements are described by attributes and associations to other elements.

element type  Any element held in the dictionary is classified as being of a particular type. Examples of element type are entity, attribute, program module, table, diagram, text, softbox. Occurrences or instances of these are called elements.

enabled  An interface element state that means that a menu item, button, etc., can be used in the current context, that is, it responds to keyboard or cursor/mouse input.

enclosing object  An object that contains another object. An object is considered to be enclosed by another object only if all of the following are true: both objects belong to the same region (Body, Margin, Header, or Trailer); the outermost of the two objects is a frame or repeating frame; the outermost of the two objects is behind the other object; the innermost of the two objects lies entirely within the borders of the other object.

exception  A warning or error condition generated from a PL/SQL subprogram.

executable argument  An argument you can pass to a product executable, such as ‘OPENFILE=my_disp’ and ‘PRINT=YES’. These arguments usually are specified when the executable is invoked from a command line.

execute  See run.

export  To store a copy of an object, module, selected text or image to a file or database.

expression  A PL/SQL construct combining variables, constants, literals, and operations on their values.

external PL/SQL library  See library.

external query  An ANSI–standard SQL SELECT statement that can be referenced by other Oracle products.

field  1. An interface element in which you enter, edit, or delete data. 2. In Graphics or Reports, a layout object that define how the data for a specific query column appears.

field template  In Graphics, a collection of attributes and properties that determines how dependent data (values) are represented as chart fields, such as bars or lines.

fill area  The area within an object to which a fill (color or pattern) should be applied.

filter  A PL/SQL function or anonymous function that restricts the data displayed by a group.
Glossary – 8 Graphics Developer's Guide

E

EBCDIC
Acronym for Extended Binary–Coded Decimal Interchange Code.

edge
The boundary of an object.

Edge/Fill/Text Display
A display box on the Layout editor tool palette that shows the current settings for the edge, fill, and text patterns and colors.

editor
A work area in which you perform a specific set of tasks, such as creating a program unit or designing an application.

element
A thing of significance about which you may need to record information. Elements are described by attributes and associations to other elements.

element type
Any element held in the dictionary is classified as being of a particular type. Examples of element type are entity, attribute, program module, table, diagram, text, softbox. Occurrences or instances of these are called elements.

enabled
An interface element state that means that a menu item, button, etc., can be used in the current context, that is, it responds to keyboard or cursor/mouse input.

enclosing object
An object that contains another object. An object is considered to be enclosed by another object only if all of the following are true: both objects belong to the same region (Body, Margin, Header, or Trailer); the outermost of the two objects is a frame or repeating frame; the outermost of the two objects is behind the other object; the innermost of the two objects lies entirely within the borders of the other object.

exception
A warning or error condition generated from a PL/SQL subprogram.

executable argument
An argument you can pass to a product executable, such as 'OPENFILE=my_disp' and 'PRINT=YES'. These arguments usually are specified when the executable is invoked from a command line.

execute
See run.

export
To store a copy of an object, module, selected text or image to a file or database.

expression
A PL/SQL construct combining variables, constants, literals, and operations on their values.

external PL/SQL library
See library.

external query
An ANSI–standard SQL SELECT statement that can be referenced by other Oracle products.

F

field
1. An interface element in which you enter, edit, or delete data. 2. In Graphics or Reports, a layout object that define how the data for a specific query column appears.

field template
In Graphics, a collection of attributes and properties that determines how dependent data (values) are represented as chart fields, such as bars or lines.

fill area
The area within an object to which a fill (color or pattern) should be applied.

filter
A PL/SQL function or anonymous function that restricts the data displayed by a group.
flex mode  Layout editor mode that automatically resizes or shrinks parent objects when child objects are moved or resized, and adjusts the positions of other layout objects that might otherwise be affected by the repositioning or resizing of the objects.

focus  The state of an entity that is able to respond to input from the user or the client. If an entity has the keyboard focus, it can receive events when the user presses a key. If a drawn view has the drawing focus, it can respond to client routines that affect drawing.

foreground element  The black part of a pattern. You apply color to the foreground element using the Edge Color palette.

foreign key  A value or column in one table that refers to a primary key in another table.

form layout  In Reports, a default layout style in which labels appear to the left of fields, and each record prints on a separate logical page.

form letter layout  In Reports, a default layout style that intermixes boilerplate and fields, wrapping lines at “word” breaks.

format mask  (Field Properties) A setting that defines the appearance of the value of a field. For example, a format mask is used to specify the display of currency amounts and dates.

format trigger  A PL/SQL function that allows you to dynamically change the formatting attributes of an object.

formula column  A user–created column that gets its data from a PL/SQL function or expression, a user exit, a SQL statement, or any combination of these.

frame  1. In Graphics, the part of a chart template that defines the basic structure of a chart, but does not determine the way the data is plotted. 2. In Reports, a layout object used to enclose other layout objects and control the formatting, frequency, and positioning of several objects simultaneously.

function  A PL/SQL subprogram that performs a specified sequence of actions, and then returns a value.

generate  To save an application to a file or database in a binary format that can be executed by Runtime and Batch executables. See also: run.

Generate  The Forms executable used to save an application to a file or database in a binary format.

global variable  A logical container that exists across an application. When an application uses a global variable, the application maintains the variable until the application is exited, or until another object explicitly removes it. See also: parameter, bind variable, local variable.

grant  To give a user access to a module. Only a module’s creator can grant its access to other users.

gridlines  In Graphics, the lines that separate rows and columns in the Layout editor.
flex mode
Layout editor mode that automatically resizes or shrinks parent objects when child objects are moved or resized, and adjusts the positions of other layout objects that might otherwise be affected by the repositioning or resizing of the objects.

focus
The state of an entity that is able to respond to input from the user or the client. If an entity has the keyboard focus, it can receive events when the user presses a key. If a drawn view has the drawing focus, it can respond to client routines that affect drawing.

foreground element
The black part of a pattern. You apply color to the foreground element using the Edge Color palette.

foreign key
A value or column in one table that refers to a primary key in another table.

form layout
In Reports, a default layout style in which labels appear to the left of fields, and each record prints on a separate logical page.

form letter layout
In Reports, a default layout style that intermixes boilerplate and fields, wrapping lines at "word" breaks.

format mask
(Field Properties) A setting that defines the appearance of the value of a field. For example, a format mask is used to specify the display of currency amounts and dates.

format trigger
A PL/SQL function that allows you to dynamically change the formatting attributes of an object.

formula column
A user–created column that gets its data from a PL/SQL function or expression, a user exit, a SQL statement, or any combination of these.

frame
1. In Graphics, the part of a chart template that defines the basic structure of a chart, but does not determine the way the data is plotted. 2. In Reports, a layout object used to enclose other layout objects and control the formatting, frequency, and positioning of several objects simultaneously.

function
A PL/SQL subprogram that performs a specified sequence of actions, and then returns a value.

generate
To save an application to a file or database in a binary format that can be executed by Runtime and Batch executables. See also: run.

Generate
The Forms executable used to save an application to a file or database in a binary format.

global variable
A logical container that exists across an application. When an application uses a global variable, the application maintains the variable until the application is exited, or until another object explicitly removes it. See also: parameter, bind variable, local variable.

grant
To give a user access to a module. Only a module's creator can grant its access to other users.

gridlines
In Graphics, the lines that separate rows and columns in the Layout editor.

G
group  1. An object that is composed of several other objects.  2. In Reports, a data model object that is created automatically to contain all the columns selected by a query, or created by the user to modify the hierarchy of the data appearing in a report; it is used primarily for creating breaks in a report, as well as for resetting computations.  See also: break group.

group filter  A PL/SQL function that restricts the data fetched by a group.

group tree  A hierarchical model—similar to a “family tree”—that describes the relationships between group objects and the objects that compose them.  See also: antecedent, child, descendant, parent, sibling.

GUI  Acronym for graphical user interface.  Also known as a bitmapped interface.

handle  An internal pointer to an application object.  In order to reference an object in a PL/SQL program unit, you must pass its handle as an argument to the appropriate subprogram.

header  An optional report region that can contain introductory material for the report, including text, graphics, data, and computations.  The report header appears first, before the body and trailer.

heading node  A node in the Object Navigator denoting a type of object which can be contained within or associated with a report.  Selecting a heading node does not select an object in the report.  See also: node.

hidden  (Field Properties)  A property that specifies whether Reports will suppress the indicated field at its location in the layout.  If Hidden is specified, the field’s value will appear only where referenced in text objects (using &fieldname).

ICD  Acronym for interface C definition—a PL/SQL application programming interface to functionality implemented in C.

icon  A graphic representation of a window or tool.

image  A bitmapped object that can be stored and loaded into an application.  The client cannot modify an imported image.

import  To read a module from the file system or database, and incorporate it into an application.

independent data  A piece of data that does not depend on other data for its value.  For example, an employee’s name may have the value Jones, which is independent of the values of other employee’s names or associated data.  Also called category data.

index  An optional structure associated with a table that is used by Oracle Server to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

inner bounding box  The virtual rectangle that constitutes an object’s ideal shape and connects its four control points, regardless of edge thickness or other attribute settings.  See also: outer bounding box.

insert mode  A mode in which each character you enter is inserted at the cursor, pushing the following characters to the right.  The opposite of replace mode.

Interpreter script  A file containing any mixture of PL/SQL source, Procedure Builder commands, and SQL statements.

join  Combining data from two (or more) tables in a single SELECT statement.

justification  See alignment.
group
1. An object that is composed of several other objects. 2. In Reports, a data model object that is created automatically to contain all the columns selected by a query, or created by the user to modify the hierarchy of the data appearing in a report; it is used primarily for creating breaks in a report, as well as for resetting computations. See also: break group.

group filter
A PL/SQL function that restricts the data fetched by a group.

group tree
A hierarchical model—similar to a "family tree"—that describes the relationships between group objects and the objects that compose them. See also: antecedent, child, descendant, parent, sibling.

GUI
Acronym for graphical user interface. Also known as a bitmapped interface.

handle
An internal pointer to an application object. In order to reference an object in a PL/SQL program unit, you must pass its handle as an argument to the appropriate subprogram.

header
An optional report region that can contain introductory material for the report, including text, graphics, data, and computations. The report header appears first, before the body and trailer.

heading node
A node in the Object Navigator denoting a type of object which can be contained within or associated with a report. Selecting a heading node does not select an object in the report. See also: node.

hidden
(Field Properties) A property that specifies whether Reports will suppress the indicated field at its location in the layout. If Hidden is specified, the field's value will appear only where referenced in text objects (using &fieldname).

ICD
Acronym for interface C definition—a PL/SQL application programming interface to functionality implemented in C.

icon
A graphic representation of a window or tool.

image
A bitmapped object that can be stored and loaded into an application. The client cannot modify an imported image.

import
To read a module from the file system or database, and incorporate it into an application.

independent data
A piece of data that does not depend on other data for its value. For example, an employee's name may have the value Jones, which is independent of the values of other employee's names or associated data. Also called category data.

index
An optional structure associated with a table that is used by Oracle Server to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

inner bounding box
The virtual rectangle that constitutes an object's ideal shape and connects its four control points, regardless of edge thickness or other attribute settings. See also: outer bounding box.

insert mode
A mode in which each character you enter is inserted at the cursor, pushing the following characters to the right. The opposite of replace mode.

Interpreter script
A file containing any mixture of PL/SQL source, Procedure Builder commands, and SQL statements.

join
Combining data from two (or more) tables in a single SELECT statement.

justification
See alignment.
**Keyword**  1. Part of a command line syntax that must be supplied with a corresponding argument. 2. A required part of a PL/SQL construct.

**Label**  Text identifying the values or meaning of an application’s visual objects.

**Layout**  The area of an editor in which you can create, modify, position, or delete objects.

**Layout Unit**  A unit of measurement for ensuring a consistent measurement system across different display devices. Used to specify a position on the layout, or the dimensions of an object.

**Lexical Reference**  A reference to a parameter used to represent a string of text in a SQL SELECT statement. For a lexical reference, you must precede the parameter name with an ampersand (&). See also: bind reference, parameter.

**Library**  A collection of one or more PL/SQL program units that are stored together in a file or database, and that can be referenced by several applications at once.

**Link**  A data model object used to define a master/detail (parent–child) relationship between a group and a query.

**Link File**  The name of a file to which a boilerplate object is associated. At runtime, Reports imports the file’s contents into your report layout.

**List Box**  A list of items within a dialog box. Usually, the list box offers choices for selection, and it is scrollable.

**List Item**  In Forms, an item that displays a list of choices.

**List of Values**  Also called LOV. A scrollable popup window that provides the user with a single or multi-column selection list.

**Local Database**  1. The database on the computer running the application. 2. The database to which an application is connected. This database parses and executes all SQL statements generated by the application. See also: remote database.

**Local Variable**  A PL/SQL variable declared only within the scope of the current program unit. See also: bind variable, global variable.

**Logical Page**  One page of your actual report (it may be displayed on one or more physical pages).

**Loop**  In a PL/SQL subprogram, a conditional statement that is repeated until the condition is no longer met.

**Mailing Label Layout**  A default layout that displays records in one or more columns on the page. Fields appear directly beneath each other in a single repeating frame.

**Margin**  An optional report region that appears at the top and bottom of each logical page in the body region. The margin may include any layout object, but typically contains boilerplate and fields (for page numbers, page totals, grand totals, and current date and time).

**Master/Detail Layout**  A default layout in which the detail records appear below the master records. Master records appear in form layout style, while detail records appear in tabular layout style.
keyword
1. Part of a command line syntax that must be supplied with a corresponding argument.
2. A required part of a PL/SQL construct.

label
Text identifying the values or meaning of an application's visual objects.

layout
The area of an editor in which you can create, modify, position, or delete objects.

layout unit
A unit of measurement for ensuring a consistent measurement system across different display devices. Used to specify a position on the layout, or the dimensions of an object.

lexical reference
A reference to a parameter used to represent a string of text in a SQL SELECT statement. For a lexical reference, you must precede the parameter name with an ampersand (&).

See also: bind reference, parameter.

library
A collection of one or more PL/SQL program units that are stored together in a file or database, and that can be referenced by several applications at once.

link
A data model object used to define a master/detail (parent–child) relationship between a group and a query.

link file
The name of a file to which a boilerplate object is associated. At runtime, Reports imports the file's contents into your report layout.

list box
A list of items within a dialog box. Usually, the list box offers choices for selection, and it is scrollable.

list item
In Forms, an item that displays a list of choices.

list of values
Also called LOV. A scrollable popup window that provides the user with a single or multi-column selection list.

local database
1. The database on the computer running the application.
2. The database to which an application is connected. This database parses and executes all SQL statements generated by the application.

See also: remote database.

local variable
A PL/SQL variable declared only within the scope of the current program unit.

See also: bind variable, global variable.

logical page
One page of your actual report (it may be displayed on one or more physical pages).

loop
In a PL/SQL subprogram, a conditional statement that is repeated until the condition is no longer met.

mailing label layout
A default layout that displays records in one or more columns on the page. Fields appear directly beneath each other in a single repeating frame.

margin
An optional report region that appears at the top and bottom of each logical page in the body region. The margin may include any layout object, but typically contains boilerplate and fields (for page numbers, page totals, grand totals, and current date and time).

master/detail layout
A default layout in which the detail records appear below the master records. Master records appear in form layout style, while detail records appear in tabular layout style.
matrix cell  A repeat direction that displays each record of a repeating frame as a cell of a matrix. See also: matrix report.

matrix layout  A default layout, resembling a grid, that formats the four sets of data necessary for a matrix report; one set of data displays as a row of labels, one as a column of labels, one set creates the grid, and one set fills the cells of the grid. See also: matrix report.

matrix report  A cross-tabulation of four sets of data: one set of data displays across the page; one set of data displays down the page; one set of data is the cross-product, determining all possible locations in which the across and down data relate and placing a cell in those locations; and one set of data displays as the “filler” of the cells.

megabyte (Mb)  A unit of memory equal to 1,048,576 bytes (1024 x 1024). Often rounded to one million bytes.

message box  A modal window that notifies you of a condition that occurred because of your last action. You must respond to a message box. See also: alert.

modal window  A window that elicits a response from the operator before the application can continue. See also: modeless window.

modeless window  An unrestricted window that can have as much functionality as the runtime window manager can support.

module  1. An object that can be shared by multiple applications. External queries and PL/SQL libraries are examples of modules. 2. In Forms, a major component of an application (i.e., a menu module).

multi-line field  A field in a window or dialog that extends for several lines, allowing you to scroll vertically through the information.

nested matrix  Matrix report in which either the across and/or down dimension contains one or more master/detail relationships. There is no limit to the amount of nesting allowed.

NOCACHE  A datatype indicating that the data will be fetched when the page on which it appears is formatted (instead of fetched and cached until formatted).

node  A specific location in the Object Navigator object hierarchy. Nodes can be expanded to reveal information about an object, such as cross references to other objects. See also: heading node.

non-caching column  A report column that references a database column of datatype NOCACHE.

NULL value  The absence of a value.

object  An item that can be placed on the layout. The following are examples of objects: rectangle, line, ellipse, arc, polygon, polyline, rounded rectangle, freehand, chart, text, symbol, and text field.

object group  A container for a group of objects, used to provide the ability to copy or reference related objects.

object handle  See handle.

Object Navigator  The window containing a hierarchical list of objects for the current session. The list appears in outline form, and enables the user to accomplish several tasks such as creating, editing, renaming, and deleting objects. See also: node, heading node.

OLE  Object Linking and Embedding.
matrix cell
A repeat direction that displays each record of a repeating frame as a cell of a matrix.
See also: matrix report.

matrix layout
A default layout, resembling a grid, that formats the four sets of data necessary for a matrix report; one set of data displays as a row of labels, one as a column of labels, one set creates the grid, and one set fills the cells of the grid.
See also: matrix.

matrix report
A cross-tabulation of four sets of data: one set of data displays across the page; one set of data displays down the page; one set of data is the cross-product, determining all possible locations in which the across and down data relate and placing a cell in those locations; and one set of data displays as the “filler” of the cells.

megabyte (Mb)
A unit of memory equal to 1,048,576 bytes (1024 x 1024). Often rounded to one million bytes.

message box
A modal window that notifies you of a condition that occurred because of your last action. You must respond to a message box.
See also: alert.

modal window
A window that elicits a response from the operator before the application can continue.
See also: modeless window.

modeless window
An unrestricted window that can have as much functionality as the runtime window manager can support.

module
1. An object that can be shared by multiple applications. External queries and PL/SQL libraries are examples of modules.
2. In Forms, a major component of an application (i.e., a menu module).

multi-line field
A field in a window or dialog that extends for several lines, allowing you to scroll vertically through the information.

N
nested matrix
Matrix report in which either the across and/or down dimension contains one or more master/detail relationships. There is no limit to the amount of nesting allowed.

NOCACHE
A datatype indicating that the data will be fetched when the page on which it appears is formatted (instead of fetched and cached until formatted).

node
A specific location in the Object Navigator object hierarchy. Nodes can be expanded to reveal information about an object, such as cross references to other objects.
See also: heading node.

non-caching column
A report column that references a database column of datatype NOCACHE.

NULL value
The absence of a value.

O
object
An item that can be placed on the layout. The following are examples of objects: rectangle, line, ellipse, arc, polygon, polyline, rounded rectangle, freehand, chart, text, symbol, and text field.

object group
A container for a group of objects, used to provide the ability to copy or reference related objects.

object handle
See handle.

Object Navigator
The window containing a hierarchical list of objects for the current session. The list appears in outline form, and enables the user to accomplish several tasks such as creating, editing, renaming, and deleting objects.
See also: node, heading.
OLE Container  An application that can store and display OLE objects.

OLE Server  An application that creates OLE objects.

Oracle Reports Server  A batch report executable found only on Microsoft Windows. The executable is loaded once when a report is launched by another Developer/2000 tool and remains in memory. It enables you to manage the execution of reports; e.g., cancel a report, reposition a report in the queue, etc.

origin  Represented in x and y coordinates. The origin of a view indicates the location of the view on its parent’s canvas. The origin of a canvas indicates the position of drawing on a drawn view.

outer bounding box  The smallest virtual rectangle that completely surrounds an object. See also: inner bounding box.

overflow  When an object’s contents cannot fit on the current logical page. In the event of overflow, some of the contents are moved to the next logical page.

package body  Includes the actual implementation of the package, which may include private subprograms and datatypes. The body is optional if the package consists only of declarations.

package specification  Declares the public interface to the package—that is, the datatypes and subprograms that can be referenced by other program units.

page break  A setting that controls whether to move the current object and its children to the next logical page. You can set page breaks to occur before or after the current object appears in the report output.

palette  An interface element that displays tools, colors, or patterns available in an editor.

pane  A distinct portion of an editor where work is performed, usually resizeable. See also: split bar.

panel  The number of physical pages needed to print one logical page.

parameter  1. A PL/SQL construct used to pass information to a subprogram. For example, in the subprogram call MYPROC(x), x is a parameter. 2. In Developer/2000, parameters are used to pass information between components such as Forms and Graphics. 3. A variable you can change at runtime.

parameter list  A group of parameter names and associated values used to pass information between Developer/2000 components.

parent  An object that is immediately above another object in its group tree. A group object is the parent to each of the objects that compose it. Every object is the parent of its children.
OLE Container
An application that can store and display OLE objects.

OLE Server
An application that creates OLE objects.

Oracle Reports Server
A batch report executable found only on Microsoft Windows. The executable is loaded once when a report is launched by another Developer/2000 tool and remains in memory. It enables you to manage the execution of reports; e.g., cancel a report, reposition a report in the queue, etc.

origin
Represented in x and y coordinates. The origin of a view indicates the location of the view on its parent's canvas. The origin of a canvas indicates the position of drawing on a drawn view.

outer bounding box
The smallest virtual rectangle that completely surrounds an object.

See also: inner bounding box.

overflow
When an object's contents cannot fit on the current logical page. In the event of overflow, some of the contents are moved to the next logical page.

package
A method of encapsulating and storing related procedures, functions, variables, and other package constructs together as a unit in the database. While packages provide the database administrator or application developer organizational benefits, they also offer increased functionality and database performance.

package body
Includes the actual implementation of the package, which may include private subprograms and datatypes. The body is optional if the package consists only of declarations.

package specification
Declares the public interface to the package—that is, the datatypes and subprograms that can be referenced by other program units.

Page break
A setting that controls whether to move the current object and its children to the next logical page. You can set page breaks to occur before or after the current object appears in the report output.

palette
An interface element that displays tools, colors, or patterns available in an editor.

pane
A distinct portion of an editor where work is performed, usually resizeable. See also: split bar.

Panel
The number of physical pages needed to print one logical page.

Parameter
1. A PL/SQL construct used to pass information to a subprogram. For example, in the subprogram call MYPROC(x), x is a parameter. 2. In Developer/2000, parameters are used to pass information between components such as Forms and Graphics. 3. A variable you can change at runtime.

Parameter list
A group of parameter names and associated values used to pass information between Developer/2000 components.

Parent
An object that is immediately above another object in its group tree. A group object is the parent to each of the objects that compose it. Every object is the parent of its children.
paste To place the contents of the Clipboard (cut or copied objects) at the current cursor location.

pattern A graphical property you can apply to the edge or fill of most objects.

Pcode The machine code or instruction list for a compiled program unit. See also: DIANA.

physical page The size of a page that is output by your printer.

plot type The type of element used to plot a field on a chart, such as bar, line, or symbol.

PL/SQL A procedural extension of SQL that provides programming constructs such as conditionals and procedures.

pld file A file that contains a Procedure Builder Interpreter script.

pll file A file that contains a PL/SQL library.

pls file A file that contains a PL/SQL program unit that has been exported to a text file.

pop–up list A list that pops up when the user performs a particular action.

preference A setting that affects the behavior of the application’s interface.

Previewer The interface with which you view and scroll through report output online.

primary key A column in a database table whose members consist of unique values that can be used to identify a row in a table.

printer control code In a printer definition file, an escape sequence to which you assign a printer code that can be referenced in a report.

prt file A file that contains a printer definition in text (e.g., ASCII or EBCDIC) format.

procedure A PL/SQL subprogram that performs a specified sequence of actions.

program unit A code structure you can create using PL/SQL. Anonymous blocks, subprogram specifications and bodies, and package specifications and bodies are examples of program units. See also: subprogram.

property A characteristic of an object that determines the behavior or appearance of that object.

property class A named object that contains a list of properties and their settings used to globally control the appearance and functionality of objects based on that class.

protocol A string you can enter when connecting to a database or server that allows you to connect to a remote database or server.

query A SQL SELECT statement that specifies the data you wish to retrieve from one or more tables or views of a database.

query filter function A PL/SQL function that, when executed, filters data retrieved by a query. Only the data that meet the conditions in the function are charted.

quit An option that terminates the current session and returns the user to the operating system. On some systems, Quit is Exit.

radio button A control (similar to a check box), appearing in sets of two or more, only one of which may be either “on” or “off” at any given time.
paste
To place the contents of the Clipboard (cut or copied objects) at the current cursor location.

pattern
A graphical property you can apply to the edge or fill of most objects.

Pcode
The machine code or instruction list for a compiled program unit. See also: DIANA.

physical page
The size of a page that is output by your printer.

plot type
The type of element used to plot a field on a chart, such as bar, line, or symbol.

PL/SQL
A procedural extension of SQL that provides programming constructs such as conditionals and procedures.

pld file
A file that contains a Procedure Builder Interpreter script.

pll file
A file that contains a PL/SQL library.

pls file
A file that contains a PL/SQL program unit that has been exported to a text file.

pop–up list
A list that pops up when the user performs a particular action.

preference
A setting that affects the behavior of the application's interface.

Previewer
The interface with which you view and scroll through report output online.

primary key
A column in a database table whose members consist of unique values that can be used to identify a row in a table.

printer control code
In a printer definition file, an escape sequence to which you assign a printer code that can be referenced in a report.

prt file
A file that contains a printer definition in text (e.g., ASCII or EBCDIC) format.

procedure
A PL/SQL subprogram that performs a specified sequence of actions.

program unit
A code structure you can create using PL/SQL. Anonymous blocks, subprogram specifications and bodies, and package specifications and bodies are examples of program units. See also: subprogram.

property
A characteristic of an object that determines the behavior or appearance of that object.

property class
A named object that contains a list of properties and their settings used to globally control the appearance and functionality of objects based on that class.

protocol
A string you can enter when connecting to a database or server that allows you to connect to a remote database or server.

Q
query
A SQL SELECT statement that specifies the data you wish to retrieve from one or more tables or views of a database.

query filter function
A PL/SQL function that, when executed, filters data retrieved by a query. Only the data that meet the conditions in the function are charted.

quit
An option that terminates the current session and returns the user to the operating system. On some systems, Quit is Exit.

R
radio button
A control (similar to a check box), appearing in sets of two or more, only one of which may be either "on" or "off" at any given time.
radio group  A set of two or more radio buttons, only one of which may be either “on” or “off” at any given time.

dfd file  A file that contains a single report definition in binary format. .rdf files are used to both run and edit reports.

read consistency  The ability to query the data throughout a database as of a single point in time.

record  One row fetched by a SQL SELECT statement.

record group  An internal data structure object with a column/row framework, that belong to the form module in which they are defined.

reference line  A line that appears at a specified value on a chart.

relational operator  A symbol used in search criteria to indicate a comparison between two values, such as the equal sign in “WHERE DEPTNO = 10.” Rows in which the comparison results in “true” are returned in the result (fetched), while rows in which the comparison returns “false” are rejected from the result.

remote database  A database on a computer other than the local database. Usually a computer on the same network, but at a different node (i.e., a database that you use through a database link). See also: local database.

rep file  A file with an extension of .rep that contains a single report definition in binary format. .rep files are used solely to run reports; you cannot edit a .rep file.

repeating frame  A layout object used to display rows of data that are fetched for a group.

replace mode  A mode in which each character you type replaces the current character at the cursor. The opposite of insert mode.

report–level objects  The objects that are used to define a report’s data model and layout. Report–level objects are assigned to a single report, and cannot be shared by other reports or other Oracle products. Objects include reports, queries, external queries, external PL/SQL libraries, groups, columns, parameters, links, frames, repeating frames, fields, boilerplate, and anchors.

rex file  A file that contains a report definition stored in text (e.g., ASCII or EBCDIC) format. .rex files are used to port reports from one platform to another.

RGB  Refers to the intensities of the red, green, and blue components of a color.

root  The group object to which all other objects in an application belong.

row  1. One set of field values in a table; for example, the fields representing one employee in the example table EMP. 2. To generate a runtime version of an application.

run  To produce application output.

runfile  See rep file.

Runtime  The interface in which users view and interact with a runtime application.

Runtime Parameter Form  A screen or window appearing optionally at runtime in which a user can modify print options and parameters prior to report execution.
radio group
A set of two or more radio buttons, only one of which may be either "on" or "off" at any given time.

rdf file
A file that contains a single report definition in binary format. .rdf files are used to both run and edit reports.

read consistency
The ability to query the data throughout a database as of a single point in time.

record
One row fetched by a SQL SELECT statement.

record group
An internal data structure object with a column/row framework, that belong to the form module in which they are defined.

reference line
A line that appears at a specified value on a chart.

relational operator
A symbol used in search criteria to indicate a comparison between two values, such as the equal sign in "WHERE DEPTNO = 10." Rows in which the comparison results in "true" are returned in the result (fetched), while rows in which the comparison returns "false" are rejected from the result.

remote database
A database on a computer other than the local database. Usually a computer on the same network, but at a different node (i.e., a database that you use through a database link). See also: local database.

rep file
A file with an extension of .rep that contains a single report definition in binary format. .rep files are used solely to run reports; you cannot edit a .rep file.

repeating frame
A layout object used to display rows of data that are fetched for a group.

replace mode
A mode in which each character you type replaces the current character at the cursor. The opposite of insert mode.

report-level objects
The objects that are used to define a report's data model and layout. Report-level objects are assigned to a single report, and cannot be shared by other reports or other Oracle products. Objects include reports, queries, external queries, external PL/SQL libraries, groups, columns, parameters, links, frames, repeating frames, fields, boilerplate, and anchors.

rex file
A file that contains a report definition stored in text (e.g., ASCII or EBCDIC) format. .rex files are used to port reports from one platform to another.

RGB
Refers to the intensities of the red, green, and blue components of a color.

root
The group object to which all other objects in an application belong.

row
1. One set of field values in a table; for example, the fields representing one employee in the example table EMP. 2. To generate a runtime version of an application.

run
To produce application output.

runfile
See rep file.

Runtime
The interface in which users view and interact with a runtime application.

Runtime Parameter Form
A screen or window appearing optionally at runtime in which a user can modify print options and parameters prior to report execution.
schema  A collection of related database objects, usually grouped by database userid. Schema objects includes tables, views, sequences, stored program units, synonyms, indexes, clusters, and database links.

scope  The level at, or range in which, an object operates.

script  See SQL script, Interpreter script.

SELECT statement  A SQL statement that specifies which rows and columns to fetch from one or more tables or views. See also: SQL statement, query.

session  The period between invoking and quitting the executable.

sibling  One of two or more objects that have the same parent object in its group tree. All objects that belong to one group object are siblings.

source  PL/SQL statements provided in a subprogram or package. You must have the source code for a program unit before you can compile it into executable form.

specification  Defines only the names, parameters, and return type (applies to functions only) of the subprogram.

split bar  A small rectangle at the bottom of the vertical scrollbar and to the left of the horizontal scrollbar in an editor. Dragging and releasing the mouse button creates a new pane, and divides the scroll bar into separate scroll bars for each pane. See also: pane.

SQL  A standard interface for storing and retrieving information in a relational database. SQL is an acronym for Structured Query Language.

sql file  A that contains a query stored in text (e.g., ASCII or EBCDIC) format.

SQL script  A file containing SQL statements that you can run to perform database administration quickly and easily. Several SQL scripts are shipped with Oracle products.

SQL statement  A SQL instruction to Oracle. A SELECT statement is one type of SQL statement. See also: SELECT statement, query.

stack  See call stack.

statement  A PL/SQL construct used for conditional, iterative, and sequential control, and for error handling. A semi–colon (;) must terminate every PL/SQL statement.

stored program unit  A procedure, function, or package that resides and executes in the Oracle7 Server.

stored subprogram  A procedure or function that resides and executes in the Oracle7 Server.

subprogram  A PL/SQL procedure or function.

summary column  A column, with a type of summary, which computes subtotals, grand totals, running totals, and other summaries of the data in a report, using one of the packaged summary functions provided by Reports.

SYLK  The file format used by the Microsoft Excel program.

syntax  The orderly system by which commands, qualifiers, and parameters are combined to form valid command strings.

system variable  One of the variables that are provided with Developer/2000 (e.g., &Logical Page, DESTYPE).
schema
A collection of related database objects, usually grouped by database userid.

scope
The level at, or range in which, an object operates.

script
See SQL script, Interpreter script.

SELECT statement
A SQL statement that specifies which rows and columns to fetch from one or more tables or views. See also: SQL statement, query.

session
The period between invoking and quitting the executable.

sibling
One of two or more objects that have the same parent object in its group tree. All objects that belong to one group object are siblings.

source
PL/SQL statements provided in a subprogram or package. You must have the source code for a program unit before you can compile it into executable form.

specification
Defines only the names, parameters, and return type (applies to functions only) of the subprogram.

split bar
A small rectangle at the bottom of the vertical scrollbar and to the left of the horizontal scrollbar in an editor. Dragging and releasing the mouse button creates a new pane, and divides the scroll bar into separate scroll bars for each pane. See also: pane.

SQL
A standard interface for storing and retrieving information in a relational database. SQL is an acronym for Structured Query Language.

sql file
A file that contains a query stored in text (e.g., ASCII or EBCDIC) format.

SQL script
A file containing SQL statements that you can run to perform database administration quickly and easily. Several SQL scripts are shipped with Oracle products.

SQL statement
A SQL instruction to Oracle. A SELECT statement is one type of SQL statement. See also: SELECT statement, query.

stack
See call stack.

statement
A PL/SQL construct used for conditional, iterative, and sequential control, and for error handling. A semi–colon (;) must terminate every PL/SQL statement.

stored program unit
A procedure, function, or package that resides and executes in the Oracle7 Server.

stored subprogram
A procedure or function that resides and executes in the Oracle7 Server.

subprogram
A PL/SQL procedure or function.

summary column
A column, with a type of summary, which computes subtotals, grand totals, running totals, and other summaries of the data in a report, using one of the packaged summary functions provided by Reports.

SYLK
The file format used by the Microsoft Excel program.

syntax
The orderly system by which commands, qualifiers, and parameters are combined to form valid command strings.

system variable
One of the variables that are provided with Developer/2000 (e.g., &Logical Page, DESTYPE).
table  A named collection of related information, stored in a relational database or server, in a two-dimensional grid that is made up of rows and columns.

tabular  A default layout displaying labels at the top of the page and rows of data underneath the labels.

tear-off menu  A submenu that the user can remove from its source using a mouse or other pointing device and drag to another part of the display screen.

text item  In Forms, an item that displays a character string value.

title bar  The horizontal area at the top of a window that displays the name of the application or interface element in that window.

title page  See header.

toggle  To turn a setting alternately on or off. For example, you can hide or show the Object Navigator.

tool  An iconic button used to create and manipulate objects in an application.

toolbar  Collection of iconic buttons that perform product commands. Usually aligned horizontally along the top, or vertically down the side of a window.

tool palette  A collection of tools

trailer  An optional report region that can contain closing material for the report, including text, graphics, data, and computations. The report trailer appears last, following the header and body.

transaction  A sequence of SQL statements treated as a single unit.

trigger  A PL/SQL procedure that is executed, or “fired,” upon a specific event.

update  To change the values for table data, in particular, by altering data values using the SQL command UPDATE, but also by deleting values using the SQL command DELETE or by inserting values using the command INSERT.

user exit  A way in which to pass control (and possibly arguments) from Developer/2000 to another Oracle product or 3GL, and then return control (and possibly arguments) back to Developer/2000.

value axis  See continuous axis.

value construct  A PL/SQL code structure that returns a value. Value constructs include functions and expressions.

variable  A named object that can be assigned a value and whose assigned value may change over time. See also: bind variable, global variable, local variable.

VBX  Visual Basic extension. Used as the file extension for special dynamic link libraries distributed on files that contain predefined VBX controls.

VBX control  A widget, built with the Microsoft Visual Basic language, used to enhance the user interface, obtain user input, and display output (e.g. sliders, grids, and control knobs).

view  A virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.
T

table

A named collection of related information, stored in a relational database or server, in a two–dimensional grid that is made up of rows and columns.

tabular

A default layout displaying labels at the top of the page and rows of data underneath the labels.

tear–off menu

A submenu that the user can remove from its source using a mouse or other pointing device and drag to another part of the display screen.

text item

In Forms, an item that displays a character string value.

title bar

The horizontal area at the top of a window that displays the name of the application or interface element in that window.

title page

See header.

toggle

To turn a setting alternately on or off. For example, you can hide or show the Object Navigator.

tool

An iconic button used to create and manipulate objects in an application.

toolbar

Collection of iconic buttons that perform product commands. Usually aligned horizontally along the top, or vertically down the side of a window.

tool palette

A collection of tools.

trailer

A optional report region that can contain closing material for the report, including text, graphics, data, and computations. The report trailer appears last, following the header and body.

transaction

A sequence of SQL statements treated as a single unit.

trigger

A PL/SQL procedure that is executed, or “fired,” upon a specific event.

update

To change the values for table data, in particular, by altering data values using the SQL command UPDATE, but also by deleting values using the SQL command DELETE or by inserting values using the command INSERT.

user exit

A way in which to pass control (and possibly arguments) from Developer/2000 to another Oracle product or 3GL, and then return control (and possibly arguments) back to Developer/2000.

V

value axis

See continuous axis.

value construct

A PL/SQL code structure that returns a value. Value constructs include functions and expressions.

variable

A named object that can be assigned a value and whose assigned value may change over time. See also: bind variable, global variable, local variable.

VBX

Visual Basic extension. Used as the file extension for special dynamic link libraries distributed on files that contain predefined VBX controls.

VBX control

A widget, built with the Microsoft Visual Basic language, used to enhance the user interface, obtain user input, and display output (e.g. sliders, grids, and control knobs).

view

A virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.
visual attribute  The font, color, and pattern properties that you set for objects that appear in your application interface.

wildcard  A character used to mean ‘any one character’ or ‘a contiguous set of characters’ within a word or phase.

window  1. A rectangular area of the desktop that contains an application. Each window has an area where you can interact with the application. Windows can be opened, resized, moved, reduced to an icon, or enlarged to fill the entire desktop. 2. In Forms, an object type you can associate with a canvas view.

WKS  The file format used by some spreadsheet programs, such as SQL*Calc, or by Lotus 1–2–3 programs.

zoom  To expand an object to allow more room for editing the contents of the field.
visual attribute

The font, color, and pattern properties that you set for objects that appear in your application interface.

W

wildcard

A character used to mean 'any one character' or 'a contiguous set of characters' within a word or phase.

window

1. A rectangular area of the desktop that contains an application. Each window has an area where you can interact with the application. Windows can be opened, resized, moved, reduced to an icon, or enlarged to fill the entire desktop. 2. In Forms, an object type you can associate with a canvas view.

WKS

The file format used by some spreadsheet programs, such as SQL*Calc, or by Lotus 1–2–3 programs.

XYZ

zoom

To expand an object to allow more room for editing the contents of the field.
Index

+, meaning in Navigator, 5 – 8
–, meaning in Navigator, 5 – 8

Activating
layers, 6 – 9
tools, 4 – 5
views, 4 – 36
Activating a layer programmatically,
how to, 13 – 7
Activating a timer, 11 – 10
Adding
field template to chart template, 9 – 8
field templates to charts, 8 – 22
layers, 6 – 8
objects to groups, 4 – 24
objects to groups via PL/SQL, 4 – 24
program unit to library, 12 – 7
reference lines, 8 – 19
Administering PL/SQL libraries, 12 – 10
AIFF–c sound files, 1 – 2
Aligning, objects, 4 – 23
Arc objects, creating, 4 – 7
Arc tool, 4 – 7
Arrows, setting, 4 – 30
Artwork, importing, 6 – 4
Artwork only, converting charts to, 8 – 28
Artwork only charts, 8 – 7

ASCII–delimited files, using as queries, 1 – 3
Assigning values to parameters, 10 – 7
Asynchronous mode, product integration,
16 – 19
Attaching PL/SQL libraries, 12 – 9
Attributes
chart templates, 9 – 2
definition of, 3 – 9
setting for lines, 4 – 30
setting for object edges, 4 – 30
setting for text objects, 4 – 11
setting for windows, 15 – 2
Axis
category, 8 – 3, 8 – 5
continuous, 8 – 5
date, 8 – 5
discrete, 8 – 5
setting continuous properties, 8 – 16
setting date properties, 8 – 16
setting discrete properties, 8 – 16
setting frame properties, 8 – 15
setting type of, 8 – 16
value, 8 – 4 to 8 – 5
Axis charts
definition of, 8 – 4
setting properties for, 8 – 15

Background color, removing from
an object, 4 – 27
Background Color palette, 4 – 29
Symbols

A

B
Background element, pattern, 4 – 26
Bar charts, suggested use, 8 – 6
Baseline, setting properties, 8 – 15
Basic tasks, 3 – 3
Batch compiling PL/SQL program units, 12 – 3
Batch executable, 1 – 7
Bevelled edges, setting, 4 – 30
Bind references, creating, 10 – 6
.bmp images, 1 – 2
Boilerplate. See Objects
Bookmarks, in Navigator, 5 – 9
Border, hiding, 4 – 27
Break charts, definition of, 8 – 13
Breakpoints, definition of, 14 – 7
Bring to front, 4 – 19
Built–ins
See also PL/SQL built–ins
layout units, 13 – 5
OG_INCH, 13 – 5
Button procedure, how executed, 11 – 4
Button procedures, definition of, 13 – 2
Buttons
creating, 11 – 11
order of execution, 11 – 4

Categories, 8 – 3
Category axis, 8 – 3, 8 – 5
Category columns, modifying, 8 – 14
Category data
definition of, 3 – 10
plotting subcategories, 8 – 13
CGM drawings, 1 – 2
Changing, displays, 15 – 3
Changing, queries, 7 – 12
Chart frames
applying colors and patterns to, 8 – 22
definition of, 9 – 2
setting depth size for, 8 – 16
setting shadow direction for, 8 – 16
setting shadow size for, 8 – 16
text and line attributes of, 8 – 22
Chart Properties dialog box, 8 – 3
Chart region, definition of, 8 – 2
Chart Template editor
definition of, 3 – 3
splitting into views, 9 – 3
Chart templates, 8 – 7
adding field templates to, 9 – 8
components of, 8 – 7, 9 – 1
creating, 9 – 5
customizing field templates, 9 – 8
customizing the chart frame, 9 – 5
definition of, 3 – 9
deleting, 9 – 7
deleting external, 9 – 11
deleting field templates, 9 – 9
editing frame of, 9 – 3
exporting, 9 – 10
field templates in, 9 – 7
granting access to, 9 – 11
importing, 9 – 11
pre–defined, 8 – 7, 9 – 1
properties and attributes of, 9 – 2
renaming, 9 – 6
renaming external, 9 – 11
saving externally, 9 – 10
selecting customized, 9 – 10
selecting pre–defined, 9 – 10
setting field properties, 9 – 8
setting label rotation, 9 – 8
setting line style, 9 – 8
setting plot type, 9 – 8
types of, 8 – 7, 9 – 1
Chart Values, assigning field
templates to, 8 – 24, 9 – 9
Charts
accessing properties, 8 – 10
adding reference lines, 8 – 19
as artwork, 8 – 7
assigning field templates, 8 – 24, 9 – 9
axis, 8 – 4
basic tasks, 3 – 6
components of, 8 – 2
converting to artwork only, 8 – 28
creating, 8 – 8
creating break, 8 – 13
creating via PL/SQL, 8 – 9
customizing, 8 – 15
definition of, 3–8
deleting, 8–29
deleting reference lines, 8–19
drawing, 8–8
drill–down, 8–26
dynamic representation of, 3–8
dynamic vs. artwork only, 8–7
editing reference lines, 8–19
field mapping for, 8–4
field template properties of, 8–22
filtering data, 8–11
format triggers, 11–8
hiding legend, 8–16
modifying category and value columns, 8–14
modifying properties of, 8–11
plotting two categories, 8–13
properties of, 8–3
reference lines, 8–19
referencing format triggers, 11–8
renaming, 8–28
setting date format for, 8–20
setting depth for, 8–16
setting field template properties, 8–23
setting number format for, 8–20
setting shadow direction for, 8–16
setting shadow size for, 8–16
slitting legend into columns, 8–16
static, 8–7
suggested uses by type, 8–6
types of, 8–6
updating, 8–25
when updated, 11–3
Checking consistency, of PL/SQL program units, 12–6
Child objects, 4–24
  showing in Navigator, 5–8
clear vs. cut, 4–31
Clearing, objects, 4–31
Clipping images, 4–21
Close Display triggers, order of execution, 11–4
Close display triggers, creating, 11–14
Closing, displays, 6–17
Collapsing, nodes in Navigator, 5–8
Color, removing background, 4–27
Color palettes
  editing, 4–28
  importing and exporting, 4–29
  using, 4–26
Colors
  changing via PL/SQL, 4–28
  setting for chart frame, 8–22
  setting for field templates, 8–23
  setting foreground and background, 4–26
Column charts, suggested use, 8–6
Columns, and field templates, 8–22
Combining, displays, 6–6
Command line, assigning parameter values, 10–8
Communication modes, product integration, 16–19
Compiling PL/SQL program units, 12–3
Components
  chart, 8–2
  chart template, 9–2
  display, 3–7
  parameters, 10–2
Components of Graphics, 3–2
Connecting to the database, 7–2
Consistency, checking PL/SQL program units, 12–6
Continuous axis, 3–10, 8–5
properties of, 8–16
Converting charts to artwork only, 8–28
Copying
  objects between displays, 5–6
  program unit from library, 12–9
Copying objects, 4–19
Copyright information, adding to a display, 6–2
Creating
  arc objects, 4–7
  bind references, 10–6
  break charts, 8–13
  buttons, 11–11
  chart templates, 9–5
  charts, 8–8
  close display trigger, 11–14
  displays, 6–2
drill–down charts, 8 – 26
drill–down relationships, 8 – 26
ellipse objects, 4 – 7
format triggers, 11 – 8
freehand objects, 4 – 7
layers, 6 – 8
lexical references, 10 – 6
line objects, 4 – 7
object in Navigator, 5 – 4
objects, 4 – 6
open display trigger, 11 – 5
parameters, 10 – 2
PL/SQL libraries, 12 – 6
PL/SQL program units, 12 – 2
polygon objects, 4 – 8
polyline objects, 4 – 8
post–execution triggers, 11 – 7
queries, 7 – 3
query filters, 8 – 11
rectangle objects, 4 – 8
rounded rectangle objects, 4 – 8
symbol objects with PL/SQL, 4 – 9
text field objects, 4 – 10
text objects, 4 – 10
text objects with PL/SQL, 4 – 12
timer triggers, 11 – 9
views, 4 – 36
Creating an integrated application, Forms and Graphics, 16 – 10
Creating parameter lists, 16 – 8
Cropping images. See Clipping images
Cue cards, how to use, 1 – 9
Custom execution procedure, creating, 7 – 9
Custom queries, 7 – 8
populating the columns, 7 – 9
Customized chart templates, selecting, 9 – 10
Customizing, charts, 8 – 15
cut vs. clear, 4 – 31
Cutting
object in Navigator, 5 – 7
objects, 4 – 31

Data
categories, 3 – 10
classification of, 3 – 10
filtering queries, 7 – 17
passing to other products, 16 – 8
plotting two categories, 8 – 13
specifying, 3 – 5, 7 – 3
values, 3 – 10
Data parameters, passing to other products, 16 – 8
Database
connecting to, 7 – 2
retrieving data from, 7 – 2
saving displays in, 6 – 16
Database login session, sharing between products, 16 – 17
Database pipes, using to share information, 15 – 5
Date axis, 8 – 5
properties of, 8 – 16
Date format
setting for charts, 8 – 20
setting for queries, 7 – 16
DDE package, definition of, 14 – 13
Debug actions, definition of, 14 – 6
Debug levels
definition of, 14 – 10
Interpreter command line, 14 – 10
nesting of, 14 – 10
top level, 14 – 10
DEBUG package, definition of, 14 – 13
Debug triggers, definition of, 14 – 8
Debugging displays, 6 – 17
Debugging program units, 14 – 7
Defining parameters, 10 – 2
Deleting
chart templates, 9 – 7
charts, 8 – 29
color palettes, 4 – 29
displays, 6 – 21
field templates from chart templates, 9 – 9
field templates from charts, 8 – 24
layers, 6 – 11
object in Navigator, 5 – 7

Dashes, setting, 4 – 30
objects, 4–31
objects via PL/SQL, 4–31
parameters, 10–10
PL/SQL libraries, 12–12
PL/SQL program units, 12–5
program units from libraries, 12–8
queries, 7–16
reference lines, 8–19
ruler guides, 4–35
sounds from the display, 6–15
views, 4–4
Depth size, setting for charts, 8–16
Deselecting objects, 4–17
Designer
customizing your work environment, 4–32
executable, 1–7
exiting from the, 6–17
runtime simulation in, 6–17
Designing a visual framework, 3–4
Detaching PL/SQL libraries, 12–10
Discrete axis, 3–10, 8–5
properties of, 8–16
Display triggers, 11–2
Displays
activating in form, 16–5
adding copyright information to, 6–2
chaining, 15–3
checking consistency in, 12–6
close trigger, 11–14
closing, 6–17
combining, 6–6
components of, 3–7
copying objects between, 5–6
creating, 6–3
creating new, 6–2
debugging, 6–17
default name of new, 6–2
definition of, 3–2
deleting, 6–21
designing visual framework, 3–4
generating, 6–18
getting information about, 6–2
granting access to, 6–19
importing, 6–6
importing artwork, 6–4
including in Forms, 16–4
including in OLE Container, 16–15
including in Reports, 16–15
invoking Forms from, 16–20
invoking Oracle Book from, 16–21
invoking Reports from, 16–20
layers, 6–7
managing windows, 15–6
open trigger, 11–5
opening, 6–3
passing data to form, 16–7
preferences, 6–2
preparing for integration, 16–2
preparing for Runtime, 15–1
printing, 6–19
receiving data from from, 16–6
receiving data from other products, 16–3
receiving text parameters from
other products, 16–3
referencing close trigger, 11–15
referencing open procedure, 11–6
renaming, 6–20
reverting to prior saved version, 6–16
revoking access to, 6–19
running, 6–17
saving, 6–15
scaling vs. clipping, 16–2
sharing information between, 15–4
sharing objects between, 5–6
sizing for integration, 16–2
sizing for OLE integration, 16–15
sizing the layout, 15–1
testing, 3–6
testing runtime, 6–17
trigger on close, 11–14
trigger on opening, 11–5
undoing changes, 6–16
views, 4–35
Double–Y charts, suggested use, 8–7
Drawing
arc objects, 4–7
charts, 8–8
ellipse objects, 4–7
freehand objects, 4–7
line objects, 4–7
objects, 4–6
polygon objects, 4–8
polyline objects, 4–8
rectangle objects, 4 – 8
rounded rectangle objects, 4 – 8
setting options, 4 – 37

Drawings
definition of, 3 – 8
deleting, 6 – 6
exporting, 6 – 5
granting access to, 6 – 6
importing, 6 – 4
renaming, 6 – 6

Drill–down charts
creating, 8 – 26
order of execution, 11 – 4
using, 8 – 26

Drill–down relationships, creating, 8 – 26

Duplicating
objects, 4 – 19
objects via PL/SQL, 4 – 20

Dynamic charts, 8 – 7

Executable arguments, passing to other
products, 16 – 7

Executables, 1 – 6

Executing program units, controlling, 14 – 10

Executing queries
event-based, 7 – 13
via PL/SQL, 7 – 14

Execution modes, product integration, 16 – 19

Execution order
Graphics events, 11 – 1
triggers, 11 – 1

Exiting, from the Designer, 6 – 17

Expanding, nodes in Navigator, 5 – 8
exploding a pie slice, how to, 13 – 8

Exporting
color palettes, 4 – 29
drawings and images, 1 – 2, 6 – 5
drawings and images via PL/SQL, 6 – 6
formats supported, 1 – 2
queries, 7 – 13
sounds, 6 – 14

Features
Graphics, 1 – 2
new, 1 – 4
Field mapping, 8 – 4

Field properties, setting in chart template, 9 – 8
Field template editor, 9 – 4

Field templates, 8 – 2, 9 – 2
adding to chart, 8 – 22
adding to chart templates, 9 – 8
applying colors and patterns to, 8 – 23
applying to chart, 8 – 23
assigning to value columns, 8 – 24, 9 – 9
customizing, 8 – 22
customizing in chart template, 9 – 8
deleting from chart templates, 9 – 9
deleting from charts, 8 – 24
in chart templates, 9 – 7
setting colors in chart templates, 9 – 8
setting field properties of, 8 – 23

Index – 6 Graphics Developer’s Guide
rectangle objects, 4 – 8
rounded rectangle objects, 4 – 8
setting options, 4 – 37

Drawings
definition of, 3 – 8
deleting, 6 – 6
exporting, 6 – 5
granting access to, 6 – 6
importing, 6 – 4
renaming, 6 – 6

Drill–down charts
creating, 8 – 26
order of execution, 11 – 4
using, 8 – 26

Drill–down relationships, creating, 8 – 26

Duplicating
objects, 4 – 19
objects via PL/SQL, 4 – 20

Dynamic charts, 8 – 7

Edge
radio button, 4 – 25
setting attributes, 4 – 30
setting color of, 4 – 26
setting pattern of, 4 – 25

Edge/Fill/Text display, 4 – 26
to 4 – 29

Editing
color palettes, 4 – 28
object in Navigator, 5 – 5
parameters, 10 – 5
PL/SQL libraries, 12 – 7
PL/SQL program units, 12 – 3
queries, 7 – 12
sounds, 6 – 13
text objects, 4 – 10

Editor.
See Layout editor

Editors, definition of, 3 – 3

Ellipse objects, creating, 4 – 7
Ellipse tool, 4 – 7

Errors, resolving, 1 – 9

Event–active object, 11 – 11

Exceptions.
See PL/SQL exceptions

Executable arguments, passing to other products, 16 – 7
Executables, 1 – 6
Executing program units, controlling, 14 – 10
Executing queries
event-based, 7 – 13
via PL/SQL, 7 – 14
Execution modes, product integration, 16 – 19
Execution order
Graphics events, 11 – 1
triggers, 11 – 1

Exiting, from the Designer, 6 – 17
Expanding, nodes in Navigator, 5 – 8
exploding a pie slice, how to, 13 – 8

Exporting
chart templates, 9 – 10
color palettes, 4 – 29
drawings and images, 1 – 2, 6 – 5
drawings and images via PL/SQL, 6 – 6
formats supported, 1 – 2
queries, 7 – 13
sounds, 6 – 14
setting label rotation for, 8 – 23
setting line style for, 8 – 23
setting patterns in chart templates, 9 – 8
setting plot type for, 8 – 23
setting properties in chart templates, 9 – 8
uses of, 8 – 22
Fields
  in charts, 8 – 2
  mapping for charts, 8 – 4
File system, saving displays to, 6 – 15
Files
  importing and exporting, 1 – 2
  specifying as queries, 7 – 4
  types of, used as queries, 1 – 3
Fill
  radio button, 4 – 25
  setting color of, 4 – 26
  setting pattern of, 4 – 25
Fill color, removing from an object, 4 – 27
Filter functions, using, 7 – 17
Filters, query, 8 – 11
Font attributes, setting for chart frames, 8 – 22
Foreground Color palette, 4 – 29
Foreground element, pattern, 4 – 26
Format triggers
  creating, 11 – 8
  referencing, 11 – 8
  when executed, 11 – 3
Formats, setting date for queries, 7 – 16
Forms
  embedding an Graphics display in, 16 – 4
  invoking from Graphics, 16 – 20
Frame editor, 9 – 3
Frames
  setting depth size, 8 – 16
  setting shadow direction, 8 – 16
  setting shadow size, 8 – 16
Freehand objects, creating, 4 – 7
Freehand tool, 4 – 7
Functionality of Graphics, 1 – 2
Functions. See PL/SQL functions
Gantt charts, suggested use, 8 – 7
Generating displays, 6 – 18
Getting information about displays, 6 – 2
GO command, controlling program unit execution, 14 – 11
Goto Mark command, using, 5 – 9
Granting access
  to chart templates, 9 – 11
  to color palettes, 4 – 29
  to displays, 6 – 19
  to drawings and images, 6 – 6
  to PL/SQL libraries, 12 – 11
Graphics
  basic tasks, 3 – 3
  components of, 3 – 2
  editors, 3 – 3
  executables, 1 – 6
  exiting from, 6 – 17
  extensible architecture of, 1 – 3
  full–screen Layout editor in, 1 – 3
  functionality of, 1 – 2
  including in Forms, 16 – 4
  including in OLE Container, 16 – 15
  installing, 1 – 7
  integration with other Oracle products, 1 – 3
  intelligent defaulting in, 1 – 3
  invoking Forms from, 16 – 20
  invoking from Reports, 16 – 15
  invoking Oracle Book from, 16 – 21
  invoking Reports from, 16 – 20
  major features of, 1 – 2
  overview of, 1 – 1
  point–and–click interface of, 1 – 2
  portability of, 1 – 3
  receiving data from other products, 16 – 3
  receiving text parameters, 16 – 3
  support for non–Oracle data sources, 1 – 3
  uses for, 1 – 2
Graphics Runtime
  chaining displays, 15 – 3
  hiding windows in, 15 – 7
managing windows, 15 – 6
preparing displays for, 15 – 1
sharing data between displays, 15 – 4
showing layers in windows, 15 – 9
Grid
description, 4 – 3
hiding and showing, 4 – 34
settings for, 4 – 34
Grid snap
description, 4 – 3
using, 4 – 35
Grouped objects, selecting within, 4 – 17
Grouping objects, 4 – 24

Handles
definition of, 13 – 5
using in PL/SQL, 13 – 6
Help system, using, 1 – 8
Hiding
object borders, 4 – 27
scrollbars, 15 – 7
windows, 15 – 7
Hiding and showing
elements on the layout, 4 – 34
layers, 6 – 10
objects in Navigator, 5 – 8
High–low charts, suggested use, 8 – 7
how to activate a new layer, 13 – 7
how to explode a pie slice, 13 – 8

Images
clipping, 4 – 21
definition of, 3 – 8
deleting, 6 – 6
exporting, 1 – 2, 6 – 5
granting access to, 6 – 6
importing, 1 – 2, 6 – 4
renaming, 6 – 6
Importing
chart templates, 9 – 11
color palettes, 4 – 29
drawings, 6 – 4
drawings and images, 1 – 2, 6 – 4
drawings and images via PL/SQL, 6 – 5
formats supported, 1 – 2
images, 6 – 4
queries, 7 – 7
sounds, 6 – 13
Information, sharing between
products, 16 – 6, 16 – 17
Installing Graphics, preparation, 1 – 7
Integrated applications, creating, 16 – 10
Integration
activating display in form, 16 – 5
communication modes, 16 – 19
creating parameter lists, 16 – 8
designing a layout for, 16 – 4
executing program units, 16 – 6
execution modes, 16 – 19
including a display in Forms, 16 – 4
including a display in OLE Container, 16 – 15
including a display in Reports, 16 – 15
invoking Forms from Graphics, 16 – 20
invoking Oracle Book from Graphics, 16 – 21
invoking Reports from Graphics, 16 – 20
locating files, 16 – 2
Oracle products, 16 – 1
passing data, 16 – 8
passing data from display to form, 16 – 7
passing data from form to display, 16 – 6
passing executable arguments, 16 – 7
passing parameters, 16 – 7
passing strings, 16 – 6
preparing a display for, 16 – 2
receiving data, 16 – 3
receiving text parameters, 16 – 3
scaling vs. clipping displays, 16 – 2
sharing information, 16 – 6, 16 – 17
sharing login sessions, 16 – 17
sizing the display, 16 – 2
Interpreter
displaying, 14 – 3
logging input and output, 14 – 11
JPEG images, 1 – 2
managing windows, 15 – 6
preparing displays for, 15 – 1
sharing data between displays, 15 – 4
showing layers in windows, 15 – 9

Grid
description, 4 – 3
hiding and showing, 4 – 34
settings for, 4 – 34

Grid snap
description, 4 – 3
using, 4 – 35

Grouped objects, selecting within, 4 – 17
Grouping objects, 4 – 24

H
Handles
definition of, 13 – 5
using in PL/SQL, 13 – 6

Help system, using, 1 – 8

Hiding
object borders, 4 – 27
scrollbars, 15 – 7
windows, 15 – 7

Hiding and showing
elements on the layout, 4 – 34
layers, 6 – 10
objects in Navigator, 5 – 8

High–low charts, suggested use, 8 – 7
how to activate a new layer, 13 – 7
how to explode a pie slice, 13 – 8

I
Images
clipping, 4 – 21
definition of, 3 – 8
deleting, 6 – 6
exporting, 1 – 2, 6 – 5
granting access to, 6 – 6
importing, 1 – 2, 6 – 4
renaming, 6 – 6

Importing
chart templates, 9 – 11
color palettes, 4 – 29
drawings, 6 – 4
drawings and images, 1 – 2, 6 – 4
drawings and images via PL/SQL, 6 – 5
formats supported, 1 – 2
images, 6 – 4
queries, 7 – 7
sounds, 6 – 13

Information, sharing between
products, 16 – 6, 16 – 17

Installing Graphics, preparation, 1 – 7

Integrated applications, creating, 16 – 10
Integration
activating display in form, 16 – 5
communication modes, 16 – 19
creating parameter lists, 16 – 8
designing a layout for, 16 – 4
executing program units, 16 – 6
execution modes, 16 – 19
including a display in Forms, 16 – 4
including a display in OLE Container, 16 – 15
including a display in Reports, 16 – 15
invoking Forms from Graphics, 16 – 20
invoking Oracle Book from Graphics, 16 – 21
invoking Reports from Graphics, 16 – 20
locating files, 16 – 2
Oracle products, 16 – 1
passing data, 16 – 8
passing data from display to form, 16 – 7
passing data from form to display, 16 – 6
passing executable arguments, 16 – 7
passing parameters, 16 – 7
passing strings, 16 – 6
preparing a display for, 16 – 2
receiving data, 16 – 3
receiving text parameters, 16 – 3
scaling vs. clipping displays, 16 – 2
sharing information, 16 – 6, 16 – 17
sharing login sessions, 16 – 17
sizing the display, 16 – 2

Interpreter
displaying, 14 – 3
logging input and output, 14 – 11

JPEG images, 1 – 2
Labels
  setting rotation for field, 8 – 23
  setting rotation in chart template, 9 – 8

Layers, 6 – 7
  activating, 6 – 9
  activating via PL/SQL, 6 – 10
  adding, 6 – 8
  deleting, 6 – 11
  hiding and showing, 6 – 10
  hiding and showing via PL/SQL, 6 – 11
  re-ordering, 6 – 11
  receiving mouse events, 11 – 12
  renaming, 6 – 8
  showing in windows, 15 – 9

Layout, 4 – 3
  definition of, 3 – 5
  hiding and showing elements on, 4 – 34
  magnifying, 4 – 33
  setting preferences, 4 – 33
  specifying location on, 13 – 5
  specifying size in, 13 – 5
  splitting into views, 4 – 4
  using grid snap, 4 – 35
  views, 4 – 35

Layout Editor
  color palette, 4 – 26
  creating objects, 4 – 6
  definition of, 3 – 3
  introduction to, 3 – 4
  pattern palette, 4 – 25
  symbol palette, 4 – 14
  Tool Palette, 4 – 4

Layout editor, description, 4 – 2

Layout Units, global variable measure, 13 – 5

Layout units, definition of, 13 – 5

Legend
  hiding and showing, 8 – 16
  moving, 8 – 18, 9 – 3
  splitting into columns, 8 – 16

Lexical references, creating, 10 – 6

Libraries
  See also PL/SQL libraries
  using, 12 – 6

Line charts, suggested use, 8 – 6

Line objects, creating, 4 – 7

Line tool, 4 – 7

Lines
  reference, 8 – 19
  setting attributes, 4 – 30
  setting attributes for chart frames, 8 – 22
  setting style for field template, 8 – 23
  setting style in chart template, 9 – 8

LIST package, definition of, 14 – 13

Loading, program unit from library, 12 – 9

Location, specifying in PL/SQL, 13 – 5

Logging, Interpreter input and output, 14 – 11

Login session, sharing between products, 16 – 17

Magnification
  decreasing, 4 – 33
  fitting layout in window, 4 – 33
  increasing, 4 – 33
  returning to normal, 4 – 34

Magnify tool, 4 – 33

Marking, object in Navigator, 5 – 9

Minus sign, meaning of, 5 – 8

Mixed charts, suggested use, 8 – 6

Modifying
  category columns, 8 – 14
  chart properties, 8 – 11
  objects, 4 – 15
  parameters, 10 – 5
  PL/SQL libraries, 12 – 7
  PL/SQL program units, 12 – 3
  queries, 7 – 12
  value columns, 8 – 14

Modules
  storage and security of, 3 – 3
  types of, 3 – 2

Mouse events, 11 – 4
  how received by an object/layer, 11 – 11

Move backward, 4 – 19

Move forward, 4 – 19

Moving
  chart legend, 8 – 18
  objects, 4 – 18
  objects via PL/SQL, 4 – 18
Multiple objects, selecting, 4 – 17

name resolution, of PL/SQL constructs, 13 – 4

Naming, objects, 4 – 30

Navigator

   See also Object Navigator
   assigning trigger procedures, 11 – 5
   collapsing an object, 5 – 8
   creating an object, 5 – 4
   deleting an object, 5 – 7
   editing an object, 5 – 5
   expanding an object, 5 – 8
   marking an object, 5 – 9
   minus sign, 5 – 8
   plus sign, 5 – 8
   renaming an object, 5 – 6
   reordering an object, 5 – 6
   searching in, 5 – 9
   selecting an object, 5 – 5

New features, 1 – 4

Nodes
   collapsing in Navigator, 5 – 8
   expanding in Navigator, 5 – 8

Number format, setting for charts, 8 – 20

Object Navigator
   assigning trigger procedures in, 11 – 5
   collapsing an object, 5 – 8
   creating an object, 5 – 4
   deleting an object, 5 – 7
   editing an object, 5 – 5
   expanding an object, 5 – 8
   marking an object, 5 – 9
   minus sign, 5 – 8
   object parts, 5 – 4
   plus sign, 5 – 8
   renaming an object, 5 – 6
   reordering an object, 5 – 6
   searching in, 5 – 9
   selecting an object, 5 – 5

Objects
   adding to a group, 4 – 24
   aligning, 4 – 23
   as buttons, 11 – 13
   attributes and properties of, 3 – 9
   bringing to front, 4 – 19
   changing color with PL/SQL, 4 – 28
   clearing, 4 – 31
   collapsing in Navigator, 5 – 8
   copying, 4 – 19
   creating, 4 – 6
   creating in Navigator, 5 – 4
   cutting, 4 – 31
   definition of, 3 – 5
   deleting, 4 – 31
   deleting in Navigator, 5 – 7
   deleting with PL/SQL, 4 – 31
   description of, 4 – 3
   deselecting, 4 – 17
   duplicating, 4 – 19
   duplicating with PL/SQL, 4 – 20
   dynamic representation of, 3 – 8
   editing in Navigator, 5 – 5
   event–active, 11 – 11
   expanding in Navigator, 5 – 8
   grouping, 4 – 24
   handles for, 13 – 5
   hiding borders of, 4 – 27
   marking in Navigator, 5 – 9
   modifying, 4 – 15
   moving, 4 – 18
   moving forward and backward, 4 – 19
   moving with PL/SQL, 4 – 18
   naming, 4 – 30
   pasting, 4 – 19
   receiving mouse events, 11 – 11
   record location of, 5 – 9
   referencing in PL/SQL, 4 – 30
   referencing procedures, 11 – 14
   removing background color, 4 – 27
   removing from a group, 4 – 24
   removing from a group with PL/SQL, 4 – 24
   renaming in Navigator, 5 – 6
   reordering in Navigator, 5 – 6
   reshaping, 4 – 21
   resizing, 4 – 20
   rotating, 4 – 22
   rotating with PL/SQL, 4 – 23
Objects

adding to a group, 4 – 24
adding to a group with PL/SQL, 4 – 24
aligning, 4 – 23
as buttons, 11 – 13
attributes and properties of, 3 – 9
bringing to front, 4 – 19
changing color with PL/SQL, 4 – 28
clearing, 4 – 31
collapsing in Navigator, 5 – 8
copying, 4 – 19
creating, 4 – 6
creating in Navigator, 5 – 4
cutting, 4 – 31
definition of, 4 – 3
deselecting, 4 – 17
duplicating, 4 – 19
duplicating with PL/SQL, 4 – 20
dynamic representation of, 3 – 8
editing in Navigator, 5 – 5
event–active, 11 – 11
expanding in Navigator, 5 – 8

Grouping, 4 – 24
Handles for, 13 – 5
hiding borders of, 4 – 27
marking in Navigator, 5 – 9
modifying, 4 – 15
moving, 4 – 18
moving forward and backward, 4 – 19
moving with PL/SQL, 4 – 18
naming, 4 – 30
pasting, 4 – 19
receiving mouse events, 11 – 11
record location of, 5 – 9
referencing in PL/SQL, 4 – 30
referencing procedures, 11 – 14
removing background color, 4 – 27
removing from a group, 4 – 24
removing from a group with PL/SQL, 4 – 24
renaming in Navigator, 5 – 6
reordering in Navigator, 5 – 6
reshaping, 4 – 21
resizing, 4 – 20
resizing with PL/SQL, 4 – 20
rotating, 4 – 22
rotating with PL/SQL, 4 – 23
Index – 11

searching in Navigator, 5 – 9
selecting, 4 – 16
selecting all, 4 – 17
selecting in Navigator, 5 – 5
selecting within a group, 4 – 17
sending to back, 4 – 19
setting colors for, 4 – 26
setting edge attributes, 4 – 30
setting patterns for, 4 – 25
sharing between displays, 5 – 6
top level, 3 – 2
types of, 3 – 8
undoing actions on, 4 – 30
ungrouping, 4 – 24
using handles for, 13 – 6
OG_INCH, global variable, 13 – 5
OGBUTTONPROCx procedure, 11 – 13
OGFORMATTRIGx procedure, 11 – 8
OGTRIGGERPROCx procedure, 11 – 9
OLE
   including an Graphics display, 16 – 15
   sizing a displays for integration, 16 – 15
OLE2 package, definition of, 14 – 13
Online documentation, opening from a display, 16 – 21
Online help, 1 – 8
Open Display triggers, order of execution, 11 – 2
Open display triggers, creating, 11 – 5
Opening displays, 6 – 3
PL/SQL libraries, 12 – 7
Options
date and number format, 8 – 20
drawing, 4 – 37
ORA_FFI package, definition of, 14 – 13
ORA_NLS package, definition of, 14 – 13
ORA_PROF package, definition of, 14 – 13
Oracle Book, invoking from Graphics, 16 – 21
Oracle Help, using, 1 – 8
Overview of Graphics, 1 – 1

Packages
   See also PL/SQL packages
   using for runtime displays, 15 – 4
Page breaks, hiding and showing, 4 – 34
Page set-up, for printing, 6 – 19
Page setup, 4 – 33
Parameter list, creating, 16 – 8
Parameters
   and drill-down relationships, 10 – 7
   assigning values, 10 – 7
   bind references, 10 – 6
   copying between displays, 5 – 6
   creating, 10 – 2
   defining, 10 – 2
   definition of, 3 – 10
   deleting, 10 – 10
   editing, 10 – 5
   initial value, 10 – 2
   lexical references, 10 – 6
   modifying, 10 – 5
   naming, 10 – 2
   passing to other products, 16 – 7
   receiving from other products, 16 – 3
   referencing, 10 – 5
   renaming, 10 – 5
   setting on mouse event, 11 – 4
   type, 10 – 2
   types of references, 3 – 10
   uses for, 10 – 1
Parent objects, 4 – 24
Pasting objects, 4 – 19
Pattern palette, using, 4 – 25
Patterns
   setting edge and fill, 4 – 25
   setting for chart frame, 8 – 22
   setting for field templates, 8 – 23
PCX images, 1 – 2
PICT images, 1 – 2
Pie charts, suggested use, 8 – 6
Pipes, using to share information, 15 – 5
PL/SQL
- activating layers using, 6–10
- adding child objects using, 4–24
- assigning values to parameters, 10–7
- button procedures, example of, 11–13
- changing colors of objects using, 4–28
- creating charts using, 8–9
- creating queries with, 7–6
- creating symbol objects with, 4–9
- creating text objects using, 4–12
- debugging program units, 14–7
- deleting objects using, 4–31
- duplicating objects using, 4–20
- executing queries using, 7–14
- exporting drawings and images using, 6–6
- hiding and showing layers using, 6–11
- importing drawings and images using, 6–5
- moving objects using, 4–18
- overview, 13–1
- playing sounds using, 6–14
- recording sounds using, 6–13
- removing child objects using, 4–24
- resizing objects using, 4–20
- rotating objects using, 4–23
- same name resolution in, 13–4
- updating charts using, 8–25
- using in custom queries, 7–8
- using in displays, 13–1
- using object handles, 13–5
PL/SQL built-ins, definition of, 13–3
PL/SQL editor, definition of, 3–3
PL/SQL exceptions
- definition of, 13–3
- pre-defined, 13–4
- user-defined, 13–4
PL/SQL functions, definition of, 13–2
PL/SQL Interpreter, displaying, 14–3
PL/SQL libraries
- adding program units to, 12–7
- administering, 12–10
- attaching, 12–9
- copying program unit from, 12–9
- creating, 12–6
- definition of, 3–2, 13–4
- deleting, 12–12
- deleting program units from, 12–8
- detaching, 12–10
file extension for, 12–9
- granting access to, 12–11
- modifying, 12–7
- opening, 12–7
- renaming, 12–11
- revoking access to, 12–11
- saving, 12–8
- using, 12–6
PL/SQL packages
- definition of, 13–3
- using for runtime displays, 15–4
PL/SQL procedures
- button, 13–2
- definition of, 13–2
- trigger, 13–3
- triggers. See Triggers
PL/SQL program units
- adding to library, 12–7
- button procedures, 11–13
- checking consistency of, 12–6
- compiling, 12–3
- copying from library, 12–9
- creating, 12–2
- definition of, 3–11, 13–2
- deleting, 12–5
- deleting from library, 12–8
- editing, 12–3
- OGBUTTONPROCx, 11–13
- OGFORMATTRIGx, 11–8
- OGTRIGGERPROCx, 11–9
- renaming, 12–5
- trigger procedure, 11–9
- working with, 12–1
PL/SQL subprograms, definition of, 13–2
Playing sounds, 6–13
- via PL/SQL, 6–14
.pll file extension, used for, 12–9
Plot type, setting in chart template, 9–8
Plot types, setting for field template, 8–23
Plotting subcategories, 8–13
Plus sign, meaning of, 5–8
Polygon objects, creating, 4–8
Polygon tool, 4–8
Polyline objects, creating, 4–8
Polyline tool, 4–8
Portability, 1 – 3
Post-execute triggers, creating, 11 – 7
Post-execution triggers, 11 – 2
Pre-defined chart templates, 8 – 7, 9 – 1
selecting, 9 – 10
Pre-defined PL/SQL exceptions, 13 – 4
Preferences
drawing options, 4 – 37
initial open display, 6 – 2
layout settings, 4 – 33
magnification, 4 – 33
ruler and grid settings, 4 – 34
setting, 4 – 32
Printing
displays, 6 – 19
setting up pages for, 6 – 19
PRN files, using as queries, 1 – 3
Procedure Builder
debug actions, 14 – 7
displaying the Interpreter, 14 – 3
packages, 14 – 13
Procedures. See PL/SQL procedures
Program unit execution, controlling, 14 – 10
Program units.
See also PL/SQL program units
adding to library, 12 – 7
compiling, 12 – 3
controlling the execution of, 14 – 10
copying between displays, 5 – 6
copying from library, 12 – 9
creating, 12 – 2
debugging, 14 – 7
definition of, 3 – 11
deleting from library, 12 – 8
editing, 12 – 3
types of, 13 – 2
Properties
axis frame, 8 – 15
chart, 8 – 3
chart template, 9 – 2
continuous axis, 8 – 16
date axis, 8 – 16
definition of, 3 – 9
discrete axis, 8 – 16
modifying chart, 8 – 11
reference line, 8 – 19
setting chart frame depth, 8 – 16
setting chart frame shadow direction, 8 – 16
setting chart frame shadow size, 8 – 16
setting for field templates, 8 – 23
Queries
accessing properties, 7 – 11
and charts, 8 – 2
classification of data, 3 – 10
copying between displays, 5 – 6
creating, 7 – 3
creating custom, 7 – 8
creating post-execution triggers, 11 – 7
creating with PL/SQL, 7 – 6
custom execution, 7 – 9
definition of, 3 – 5, 3 – 9
deleting, 7 – 16
editing, 7 – 12
executed when, 11 – 2
executing, 7 – 13
executing via PL/SQL, 7 – 14
exporting, 7 – 13
filter function execution, 11 – 2
filtering data, 7 – 17, 8 – 11
importing, 7 – 7
linking to an external, 7 – 7
modifying, 7 – 12
order of execution, 11 – 2
passing data to other products, 16 – 8
PL/SQL in, 7 – 8
populating custom, 7 – 9
receiving data from other products, 16 – 3
referencing external, 7 – 7
referencing post-execution procedure, 11 – 7
renaming, 7 – 15
setting date format for, 7 – 16
specifying files as, 7 – 4
specifying SQL SELECT statements as, 7 – 4
trigger events, 11 – 2
using external, 7 – 7
Query data
passing to other products, 16 – 8
receiving from other products, 16 – 3
Query filters, creating, 8 – 11
Quick Tour, how to use, 1 – 9
Quitting, Graphics session, 6–17

Recording sounds, 6–12
Rectangle objects, creating, 4–8
Rectangle tool, 4–8
Reference lines
  adding to charts, 8–19
  deleting, 8–19
  editing properties in chart, 8–19
References
  bind, 10–6
  lexical, 10–6
  replacing, 5–7, 12–4
Referencing, external queries, 7–7
Referencing object in program units, 4–30
Referencing parameters, 10–5
Release 2.5, new features with, 1–4
Removing
  objects from groups, 4–24
  objects from groups via PL/SQL, 4–24
  queries, 7–16
  views, 4–36
Renaming
  chart templates, 9–6
  charts, 8–28
  color palettes, 4–29
  displays, 6–20
  drawings and images, 6–6
  layers, 6–8
  object in Navigator, 5–6
  PL/SQL libraries, 12–11
  PL/SQL program units, 12–5
  queries, 7–15
  sounds in the display, 6–14
Reordering, object in Navigator, 5–6
Replacing references in a program unit, 5–7, 12–4
Reports
  including a display in, 16–15
  invoking from Graphics, 16–20
RESET command, controlling program unit execution, 14–11
Reshape tool, 4–21
Reshaping objects, 4–21
Resizing
  objects, 4–20
  objects via PL/SQL, 4–20
  views, 4–36
Resolving errors, 1–9
Revoking access
  to displays, 6–19
  to PL/SQL libraries, 12–11
Rotate tool, 4–22
Rotating
  objects, 4–22
  objects via PL/SQL, 4–23
Rounded rectangle objects, creating, 4–8
Rounded rectangle tool, 4–8
Ruler Guides
  creating, 4–35
  deleting, 4–35
Ruler guides
  description, 4–3
  hiding and showing, 4–34
  using, 4–35
Ruler markers, description, 4–3
Rulers
  description, 4–3
  hiding and showing, 4–34
  settings for, 4–34
Running displays in the Designer, 6–17
Runtime, executable, 1–7

Saving
  chart templates, 9–10
  displays, 6–15
  modules, 3–3
  PL/SQL libraries, 12–8
Scatter charts, suggested use, 8–6
Scrollbars, hiding, 15–7
Search path, product integration, 16–2
Searching for objects, 5–9
Security, modules, 3–3
Quitting, Graphics session, 6 – 17

Recording sounds, 6 – 12

Rectangle objects, creating, 4 – 8

Rectangle tool, 4 – 8

Reference lines
  adding to charts, 8 – 19
  deleting, 8 – 19
  editing properties in chart, 8 – 19

References
  bind, 10 – 6
  lexical, 10 – 6
  replacing, 5 – 7, 12 – 4

Referencing, external queries, 7 – 7

Referencing object in program units, 4 – 30

Referencing parameters, 10 – 5

Release 2.5, new features with, 1 – 4

Removing
  objects from groups, 4 – 24
  objects from groups via PL/SQL, 4 – 24
  queries, 7 – 16
  views, 4 – 36

Renaming
  chart templates, 9 – 6
  charts, 8 – 28
  color palettes, 4 – 29
  displays, 6 – 20
  drawings and images, 6 – 6
  layers, 6 – 8
  object in Navigator, 5 – 6
  PL/SQL libraries, 12 – 11
  PL/SQL program units, 12 – 5
  queries, 7 – 15
  sounds in the display, 6 – 14

Reordering, object in Navigator, 5 – 6

Replacing references in a program unit, 5 – 7, 12 – 4

Reports
  including a display in, 16 – 15
  invoking from Graphics, 16 – 20

RESET command, controlling program unit execution, 14 – 11

Revoking access
  to displays, 6 – 19
  to PL/SQL libraries, 12 – 11

Rotate tool, 4 – 22

Rotating
  objects, 4 – 22
  objects via PL/SQL, 4 – 23

Rounded rectangle objects, creating, 4 – 8

Rounded rectangle tool, 4 – 8

Reshape tool, 4 – 21

Reshaping objects, 4 – 21

Resizing
  objects, 4 – 20
  objects via PL/SQL, 4 – 20
  views, 4 – 36

Resolving errors, 1 – 9

Revoking access
to displays, 6 – 19
to PL/SQL libraries, 12 – 11

Ruler Guides
  creating, 4 – 35
  deleting, 4 – 35

Ruler guides
  description, 4 – 3
  hiding and showing, 4 – 34
  using, 4 – 35

Ruler markers, description, 4 – 3

Rulers
  description, 4 – 3
  hiding and showing, 4 – 34
  settings for, 4 – 34

Running displays in the Designer, 6 – 17

Runtime, executable, 1 – 7

Saving
  chart templates, 9 – 10
  displays, 6 – 15
  modules, 3 – 3
  PL/SQL libraries, 12 – 8

Scatter charts, suggested use, 8 – 6

Scrollbars, hiding, 15 – 7

Search path, product integration, 16 – 2

Searching for objects, 5 – 9

Security, modules, 3 – 3
Select tool, 4 – 16
Selecting
  customized chart template, 9 – 10
  multiple objects, 4 – 17, 5 – 5
  object in Navigator, 5 – 5
  object within a group, 4 – 17
  objects, 4 – 16
  parent group of child objects, 4 – 24
  pre-defined chart template, 9 – 10
Send to back, 4 – 19
Set mark command, using, 5 – 9
Setting
  attributes for text objects, 4 – 11
  object colors, 4 – 26
Shadow direction, specifying for charts, 8 – 16
Shadow size, setting for charts, 8 – 16
Showing
  layers, 6 – 10
  subobjects in Navigator, 5 – 8
Single Rotate tool, 4 – 22
Single Select tool, 4 – 17
Size
  of symbol objects, 4 – 14
  specifying in layout units, 13 – 5
Snap. See Grid snap
Sounds
  administering, 6 – 14
  copying between displays, 5 – 6
  definition of, 3 – 11
  deleting from the display, 6 – 15
  exporting, 6 – 14
  importing, 6 – 13
  playing and editing, 6 – 13
  recording, 6 – 12
  recording via PL/SQL, 6 – 13
  renaming in display, 6 – 14
Specifying data, 3 – 5
Specifying location on layout, 13 – 5
Specifying size on layout, 13 – 5
SQL SELECT statements, 7 – 1
Status line
  description, 4 – 5
  hiding and showing, 4 – 34
STEP command, controlling program unit execution, 14 – 10
Storing, modules, 3 – 3
Strings, passing between products, 16 – 6
Subprograms. See PL/SQL subprograms
Suggested uses for chart types, 8 – 6
SYLK files, using as queries, 1 – 3
Symbol objects
  changing size of, 4 – 14
  creating via PL/SQL, 4 – 9
  placing on the layout, 4 – 14
  selecting, 4 – 14
Symbol palette, 4 – 14
  hiding and showing, 4 – 34
Symbol tool, 4 – 14
Symbols, creating programmatically, 4 – 15
Synchronous mode, product integration, 16 – 19
Table charts, suggested use, 8 – 6
Tables, demo database, 1 – 7
Templates, chart, 3 – 9
Testing runtime displays, 6 – 17
Testing the display, 3 – 6
Text attributes, setting for chart frame, 8 – 22
Text field objects
  creating, 4 – 10
  entering default text in, 4 – 11
Text Field tool, 4 – 10
Text objects
  creating, 4 – 10
  creating via PL/SQL, 4 – 12
  definition of, 3 – 8
  editing, 4 – 10
  setting attributes for, 4 – 11
Text parameters, passing to other products, 16 – 7
Text tool, 4 – 10
TEXT_IO package, definition of, 14 – 13
TIFF images, 1 – 2
Timers
  activating automatically, 11 – 10
  activating via PL/SQL, 11 – 10
copying between displays, 5 – 6
creating triggers, 11 – 9
order of execution, 11 – 3
referencing procedures, 11 – 10
using a triggers, 11 – 9
Tool Palette, using, 4 – 4
Tool palette
definition of, 3 – 5
hiding and showing, 4 – 34
in Chart Template editor, 9 – 5
TOOL_ENV package, definition of, 14 – 13
TOOL_ERR package, definition of, 14 – 13
TOOL_INT package, definition of, 16 – 16
TOOL_RES package, definition of, 14 – 13
Tools
activating, 4 – 5
using, 4 – 5
Top level, definition of, 14 – 10
Trigger procedures, definition of, 13 – 3
Triggers
assigning in Navigator, 11 – 5
button procedure execution, 11 – 4
chart updates, 11 – 3
Close display, 11 – 14
close display, 11 – 4
creating button procedures, 11 – 13
creating format, 11 – 8
creating post–execution, 11 – 7
creating timer, 11 – 9
display, 11 – 2
format, 11 – 8
format execution, 11 – 3
mouse events, 11 – 4
Open display, 11 – 5
open display, 11 – 2
order of execution, 11 – 1
post–execute query, 11 – 6
query execution, 11 – 2
referencing button procedures, 11 – 14
referencing close display, 11 – 15
referencing format, 11 – 8
referencing in Navigator, 11 – 5
referencing open display, 11 – 6
referencing post–execution, 11 – 7
referencing timers, 11 – 10
timer execution, 11 – 3
timers, 11 – 9
types of, 11 – 2
Tutorial, before you begin, 1 – 7
Tutorial, 2 – 1
Undoing actions, 4 – 30
Ungrouping objects, 4 – 24
Updating
charts, 8 – 25
via the menu, 8 – 25
Updating charts
order of execution, 11 – 3
via PL/SQL, 8 – 25
User actions, assigning parameter
values, 10 – 7
User–defined PL/SQL exceptions, 13 – 4
Using
close display triggers, 11 – 14
cue cards, 1 – 9
help system, 1 – 8
online help, 1 – 8
open display triggers, 11 – 5
Oracle Help, 1 – 8
post–execute triggers, 11 – 6
Quick Tour, 1 – 9
tutorial triggers, 11 – 9
tutorial, 2 – 1
Value axis, 8 – 4, 8 – 5
Value columns
and field templates, 8 – 22
assigning field templates to, 8 – 24, 9 – 9
modifying, 8 – 14
Value data, definition of, 3 – 10
Values, 8 – 4
Views, 4 – 4
activating, 4 – 36
creating, 4 – 36
in Chart Template editor, 9 – 3
removing, 4 – 36
resizing, 4 – 36
Tutorial, before you begin, 1 – 7

Tutorial, 2 – 1

Undoing actions, 4 – 30

Ungrouping objects, 4 – 24

Updating
- charts, 8 – 25
- via the menu, 8 – 25

Updating charts
- order of execution, 11 – 3
- via PL/SQL, 8 – 25

User actions, assigning parameter values, 10 – 7

User–defined PL/SQL exceptions, 13 – 4

Using
- close display triggers, 11 – 14
- cue cards, 1 – 9
- format triggers, 11 – 8
- help system, 1 – 8
- online help, 1 – 8
- open display triggers, 11 – 5
- Oracle Help, 1 – 8
- post–execute triggers, 11 – 6
- Quick Tour, 1 – 9
- timer triggers, 11 – 9
- tutorial, 2 – 1

Value axis, 8 – 4, 8 – 5

Value columns
- and field templates, 8 – 22
- assigning field templates to, 8 – 24, 9 – 9
- modifying, 8 – 14

Value data, definition of, 3 – 10

Values, 8 – 4

Views, 4 – 4
- activating, 4 – 36
- creating, 4 – 36
- in Chart Template editor, 9 – 3
- removing, 4 – 36
- resizing, 4 – 36
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>using, 4 – 35</td>
<td></td>
</tr>
<tr>
<td>Visual framework, designing, 3 – 4</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td>destroying, 15 – 7</td>
<td></td>
</tr>
<tr>
<td>hiding and showing, 15 – 7</td>
<td></td>
</tr>
<tr>
<td>hiding scrollbars, 15 – 7</td>
<td></td>
</tr>
<tr>
<td>managing, 15 – 6</td>
<td></td>
</tr>
<tr>
<td>setting attributes, 15 – 2</td>
<td></td>
</tr>
<tr>
<td>showing layers in, 15 – 9</td>
<td></td>
</tr>
<tr>
<td>using, 4 – 37</td>
<td></td>
</tr>
<tr>
<td>WKS files, using as queries, 1 – 3</td>
<td></td>
</tr>
<tr>
<td>Zooming, display layout, 4 – 33</td>
<td></td>
</tr>
</tbody>
</table>
W

Z
Chart Components
Chart Components

- Continuous (or Value) Axis
- Grid Lines
- Fields
- Frame
- Legend
- Discrete (or Category) Axis
  - Tick Marks
  - Discrete (or Category) Axis
  - Axis Label
  - Tick Labels
# Editing Chart Properties

<table>
<thead>
<tr>
<th><strong>Property/Setting</strong></th>
<th><strong>Pop-up Menu</strong></th>
<th><strong>Property Page</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis labels</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Axis position/direction</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Axis type</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Baseline value</td>
<td>Frame</td>
<td>Axis Frame</td>
</tr>
<tr>
<td>Category data</td>
<td>Properties</td>
<td>Categories</td>
</tr>
<tr>
<td>Chart type/subtype</td>
<td>Properties</td>
<td>Chart</td>
</tr>
<tr>
<td>Column or bar shadow</td>
<td>Frame</td>
<td>Frame</td>
</tr>
<tr>
<td>Date format</td>
<td>Date Format</td>
<td>N/A</td>
</tr>
<tr>
<td>Gridlines (hide/show)</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Legend (hide/show)</td>
<td>Frame</td>
<td>Frame</td>
</tr>
<tr>
<td>Name</td>
<td>Properties</td>
<td>Chart</td>
</tr>
<tr>
<td>Number format</td>
<td>Number Format</td>
<td>N/A</td>
</tr>
<tr>
<td>Pie slice grouping</td>
<td>Frame</td>
<td>Pie Frame</td>
</tr>
<tr>
<td>Pie slice label format</td>
<td>Frame</td>
<td>Pie Frame</td>
</tr>
<tr>
<td>Pie slice order</td>
<td>Frame</td>
<td>Pie Frame</td>
</tr>
<tr>
<td>Query</td>
<td>Properties</td>
<td>Chart</td>
</tr>
<tr>
<td>Query rows to plot</td>
<td>Properties</td>
<td>Data</td>
</tr>
<tr>
<td>Subcategory value</td>
<td>Properties</td>
<td>Categories</td>
</tr>
<tr>
<td>Template type</td>
<td>Properties</td>
<td>Chart</td>
</tr>
<tr>
<td>Tick marks (hide/show)</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Tick label rotation</td>
<td>Axis</td>
<td>Axis</td>
</tr>
<tr>
<td>Title</td>
<td>Properties</td>
<td>Chart</td>
</tr>
<tr>
<td>Value data</td>
<td>Properties</td>
<td>Values</td>
</tr>
</tbody>
</table>
Editing Chart Properties

- Property/Setting Pop-up Menu
- Property Page
- Axis labels
- Axis position/direction
- Axis type
- Baseline value
- Frame
- Categories
- Chart type/subtype
- Properties
- Categories
- Column or bar shadow
- Frame
- Shadow
- Date format
- Date Format
- N/A
- Gridlines (hide/show)
- Axis
- Axis
- Legend (hide/show)
- Frame
- Frame
- Name
- Properties
- Chart
- Number format
- Number Format
- N/A
- Pie slice grouping
- Frame
- Pie
- Frame
- Pie slice label format
- Frame
- Pie
- Frame
- Pie slice order
- Frame
- Pie
- Frame
- Query
- Properties
- Chart
- Query rows to plot
- Properties
- Data
- Subcategory value
- Properties
- Categories
- Template type
- Properties
- Chart
- Tick marks (hide/show)
- Axis
- Axis
- Tick label rotation
- Axis
- Axis
- Title
- Properties
- Chart
- Value data
- Properties
- Values
Reader’s Comment Form

Name of Document:  Graphics ™ 2.5 Developer’s Guide
Part No. A32482–2

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________

Please send your comments to:
Oracle Graphics Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065 U.S.A.
Fax: (415) 506–7200

If you would like a reply, please give your name, address, and telephone number below:

__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________

Thank you for helping us improve our documentation.