Building Enterprise Information Systems: A Guide to EIS Development with Microsoft Applications

Microsoft® OPEN EIS PAK

Building Enterprise Information Systems

A Guide to EIS Development with Microsoft Applications

Microsoft Corporation

Microsoft Word was used in the production of this manual.

Document No. XL35340-1192
Printed in the United States of America.

Contents

Introduction

What Is an Enterprise Information System?

An enterprise information system (EIS) is a software system that brings strategic corporate information to company employees in a way that improves their ability to make important business decisions.

The most important goal of an EIS is to optimize the performance of a company, institution, or organization (collectively referred to as an "enterprise"). With an EIS, you can:

- View corporate data in summary or detailed format.
- Easily spot trends.
- Obtain time-critical information without substantial delay.
- Analyze data more accurately and thoroughly.
- Communicate and discuss findings in a timely manner.

Traditionally, an EIS was targeted at the top echelon of corporate executives and supplied information that enabled a precise monitoring and analysis of company statistics such as sales, inventory, and manufacturing figures. For this reason, the acronym EIS originally meant "executive information system."

Today, EIS applications are becoming more common among a broad cross section of individuals, enabling people at all levels to tap into corporate information sources and enhance their decision-making process. EIS applications are typically easy to use, highly graphical, and present data in a format that is easy to understand and draw conclusions from. A well-designed EIS application can have a substantial positive impact on a company's performance, and allows individuals within the company to make decisions quickly and with more accurate information.

A New Approach to EIS

Most traditional EIS's offer a single package so you can access, format, and view data. The problem with many of these packages is that their development tools are limited, and the solutions you build with them are narrow and inflexible. Furthermore, they are typically "one-size-fits-all," and force you to use the same package regardless of your requirements.

In contrast to these traditional systems, with the Microsoft Open EIS approach you can use a broad range of applications and technologies to build powerful and flexible custom EIS applications.

With Microsoft's Open EIS approach, you can tap the power of all of Microsoft's programmable applications and development environments such as Microsoft® Excel, Microsoft Word, Microsoft Access™, Microsoft Project, and Visual Basic™ to build an integrated solution that uses the rich feature set of each application.

Since Microsoft applications can be customized and programmed, you can build a unique user interface and feature set that appears as if it were programmed from scratch, but takes only a fraction of the time. Equally important, Microsoft applications are designed to work together. Using technologies such as dynamic data exchange (DDE) and object linking and embedding (OLE), these applications can exchange data automatically and transparently, allowing you to build EIS solutions that are very easy to use.

Using Microsoft's powerful client-server applications such as SQL Server, these application "front ends" can access data anywhere in your enterprise, and connect to almost any environment.

For additional information on Microsoft's approach to EIS solutions, see the *Introduction to Open EIS* that is included with the Microsoft Open EIS Pak.

What Is This Guide For?

This guide helps you sort through the many possibilities for building a successful EIS application, and shows you how to take advantage of the broad range of Microsoft products in your solution. The material is meant to be mostly an overview of the concepts of EIS and the options available to you, as well as a few approaches you can take while designing and implementing your system. It is not intended to be a comprehensive guide on how to build an EIS application. If you need more information than is available in this guide, see Chapter 13.

Although many of the concepts discussed in this guide can be applied to a variety of computing environments, the focus of the material will be on building EIS solutions that run on the Microsoft Windows ™ operating system.

Who Is This Guide For?

This guide is intended primarily for people who are designing and implementing EIS solutions, corporate systems, and integrated applications. Technical managers may also find the information useful.

To get the most out of this guide, you should be familiar with the Microsoft Windows graphical environment, and should understand the basics of graphical software applications such as spreadsheets, word processors, and project management software.

Although not required, a minimal background in programming with a high-level language such as Basic or an application macro language will be helpful.

All the examples and figures in this guide were prepared using the English language versions of Microsoft products for the Windows operating system. These examples and figures may be different in versions of Microsoft products that have been localized for languages other than English, or in the Apple® Macintosh® versions of the applications. However, most of the concepts discussed in this guide are independent of any particular language or environment.

CHAPTER 1

EIS Terms and Concepts

Since technology is rich in jargon, it helps to familiarize yourself with a few terms. This section presents terms with which you may not be familiar, but which are used quite often in the area of EIS. This is not a comprehensive list, but rather a few of the more commonly used terms and acronyms.

You should review this section now, and use it to refresh your memory later when you encounter an unfamiliar term.

Term	Meaning
Back end	Refers to an application that provides information or services to other applications. Often used in conjunction with the term *front end*. For example, a database application running on a server is often referred to as a back-end service.
Client	A system component that requests services from one or more servers. It can refer to an application or personal computer that receives information for local processing.
Corporate data	Any data within a corporation that is critical to the operation of the corporation. Includes information associated with human resources, budgets, sales, manufacturing, assets, and other kinds of key data.
DDE	Acronym for *dynamic data exchange*. A set of functions built-in to the Windows operating system that allows applications to exchange information with one another.
Drilldown	A capability common in EIS applications that allows a user to progressively move from summary data to more detailed data.
Enterprise	Any organization, institution, or company.
Event	The existence of a condition or set of conditions within a system. For example, a "mouse event" occurs when the user moves a mouse or clicks one of its buttons.
Event-driven	Applications that operate primarily by responding to events are said to be *event-driven*.
Front end	Synonymous with an application user interface. Often used in conjunction with *back end* (see preceding definition of "back end"). For example, a spreadsheet (front end) may connect to a database (back end) to retrieve data.
Integration	In the context of this document, *integration* means that two or more applications are linked together to create a more powerful application.
IS	Acronym for *information systems*. It refers to a department within an enterprise that manages and administers all computer hardware and software for the enterprise. This term is widely used as a replacement for the older MIS acronym, which stands for *management information systems*.

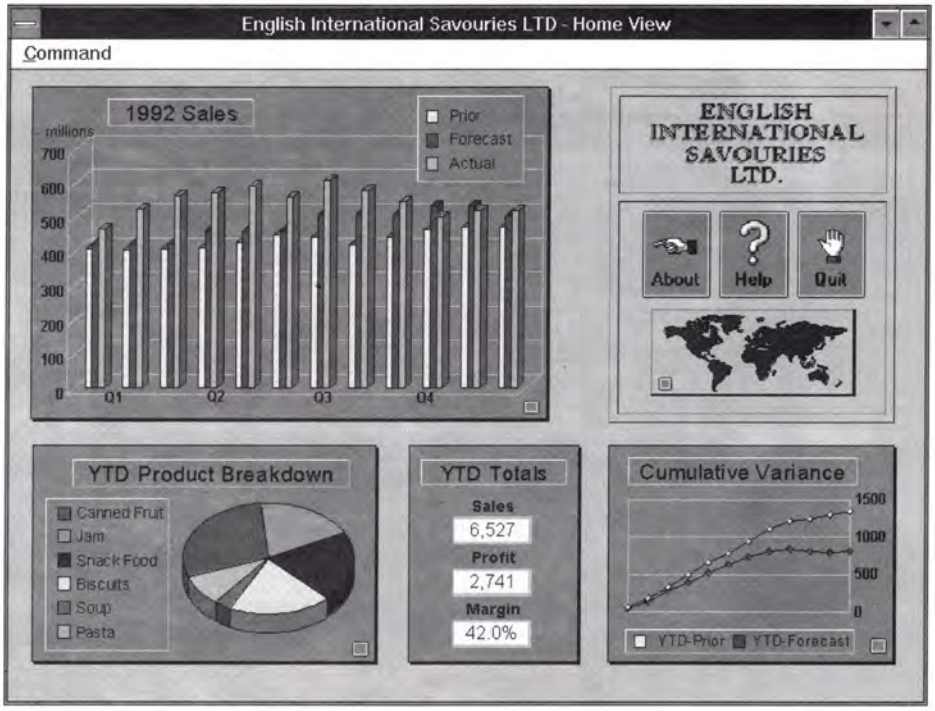
Term	Meaning
Macro	A series of program statements that can be executed by an application to perform various tasks. Macros are written using a simple, high-level programming language.
MAPI	Acronym for *messaging application programming interface*. A standard set of functions that allows applications in the Windows environment to access a variety of messaging services (such as electronic mail) in a consistent manner. MAPI simplifies the development of messaging applications by allowing developers to use the same code to access a variety of messaging systems from different vendors.
Multisource data	Refers to data that exists in a number of different locations and file formats.
Object	Any element of information that can be manipulated by applications as a single unit. Examples of objects include charts, tables, documents, images, text paragraphs, video clips, and sound clips.
ODBC	Acronym for *open database connectivity*. A standard set of functions that allows applications in the Windows environment to access a variety of database management systems in a consistent manner. ODBC simplifies the development of database applications by allowing developers to use the same code to access a variety of databases from different vendors.
OLE	Acronym for *object linking and embedding*. A set of functions built-in to the Windows operating system that allows applications to share objects such as charts, tables, and images.
RDBMS	Acronym for *relational database management system*. Any application designed to manage multiple tables of data and allow users to establish relationships among the data.
Seamless	When two or more software components are integrated such that a user needs no knowledge of how to switch from one to the other, or even that they are using different components, it is said that these components are integrated "seamlessly."
Server	An application or computer that provides services to other applications or computers. Often used in conjunction with the term "client."
Service	One or more operations that are performed upon request by a client.
SQL	Acronym for *Structured Query Language*. A standard language for retrieving information from database management systems.
WOSA	Acronym for *Windows open services architecture*. A set of standard APIs that provides a single, system-level interface for connecting front-end applications to various back-end services and data sources. WOSA APIs have been developed for printing, SNA networking, UNIX® networking, licensing, and many other functions. The ODBC and MAPI interfaces discussed previously are also part of WOSA.

CHAPTER 2

An Overview of an Example EIS

To get a better idea of what EIS is, it helps to start with an overview of an example system. Suppose there is a company with employees that want to know more about its sales operation so they can make better decisions about everything from manufacturing and inventory to product marketing and public relations. Assume for now that this company already has an order entry system installed and keeps its sales data in a database system on a mainframe.

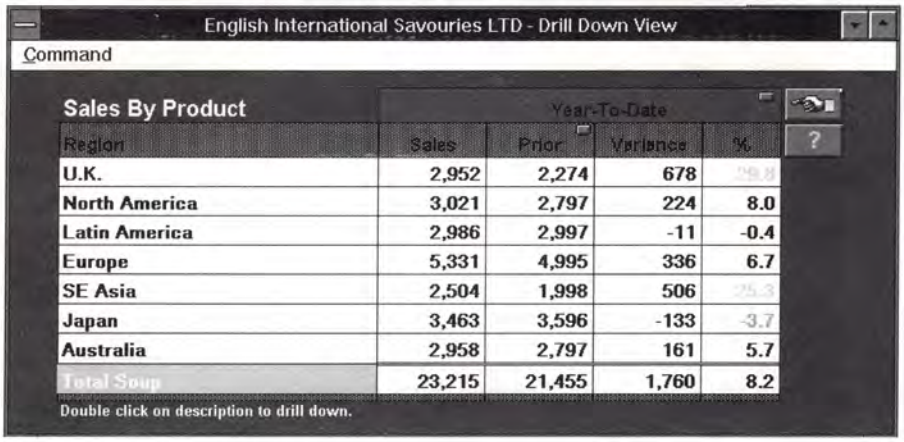
When a user starts the EIS application, the following screen is displayed.



In EIS terminology, this is the *home* screen. Home screens are the top-level screen in a series of screens that display different information. Among the first things to notice is the simplicity of the interface. On this screen (and all other screens) there are just a few buttons and a single menu. Users can click the buttons to get additional information on total sales, product sales, and cumulative variance.

This particular home screen provides critical summary sales information in addition to its screen navigation controls. Each EIS is different, however, and some have home screens with only controls and no summary information.

To obtain more data on cumulative variance, the user clicks the Cumulative Variance Graph button, which displays the following screen.

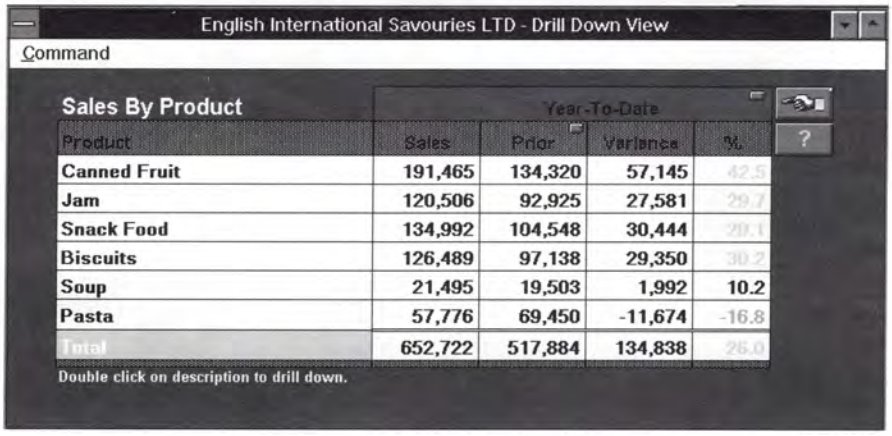


Although the home screen contained this information, it did so only in summary format. The previous screen shows actual figures, providing a more detailed analysis of the data. This successive presentation of greater detail as the user moves through the screen hierarchy is typical for an EIS, and is a recommended approach. In EIS terminology, this is also called *drilldown* to refer to the process of "drilling" through different layers of detail.

Drilldown is recommended because it allows different users to view different levels of detail on different data, depending on their needs. An executive may want to see only summary information, while a product marketing manager may want more specific information to help prepare a promotional campaign. Drilldown also gives your EIS a natural flow and helps makes the user interface more intuitive.

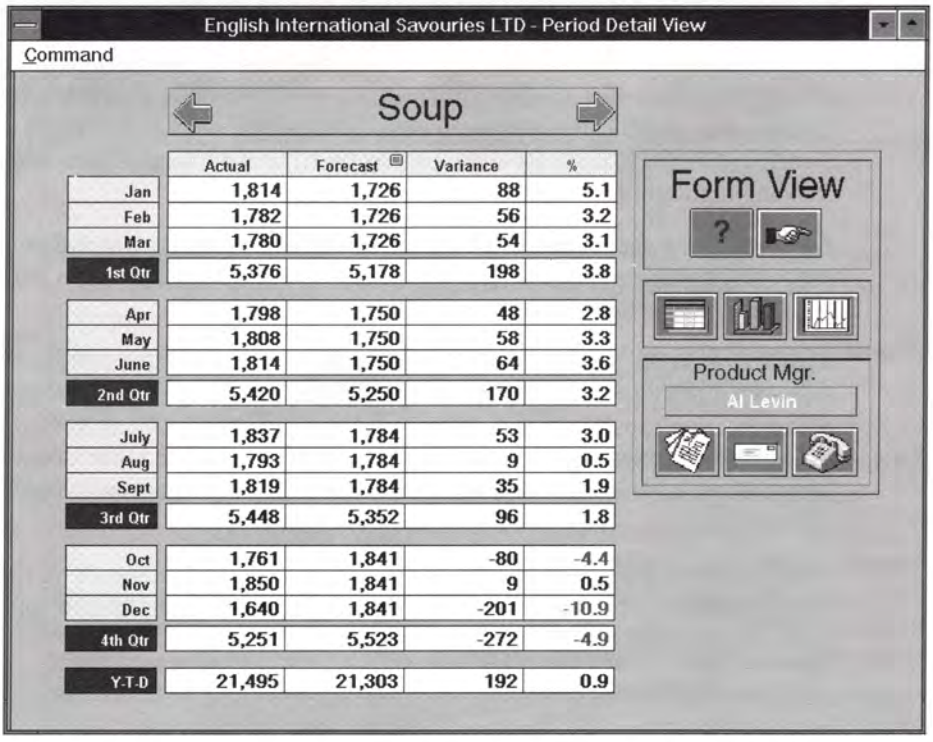
Notice on the preceding screen that there is a message below the Product column which says "Double-click on descriptions to drilldown." If the user double-clicks on the word Soup in the table, the following dialog box is displayed.

This dialog box allows you to select the type of drilldown desired. If the user selects the By Region button, the following screen appears, showing totals for soup sales in the seven regions where the company sells its products.

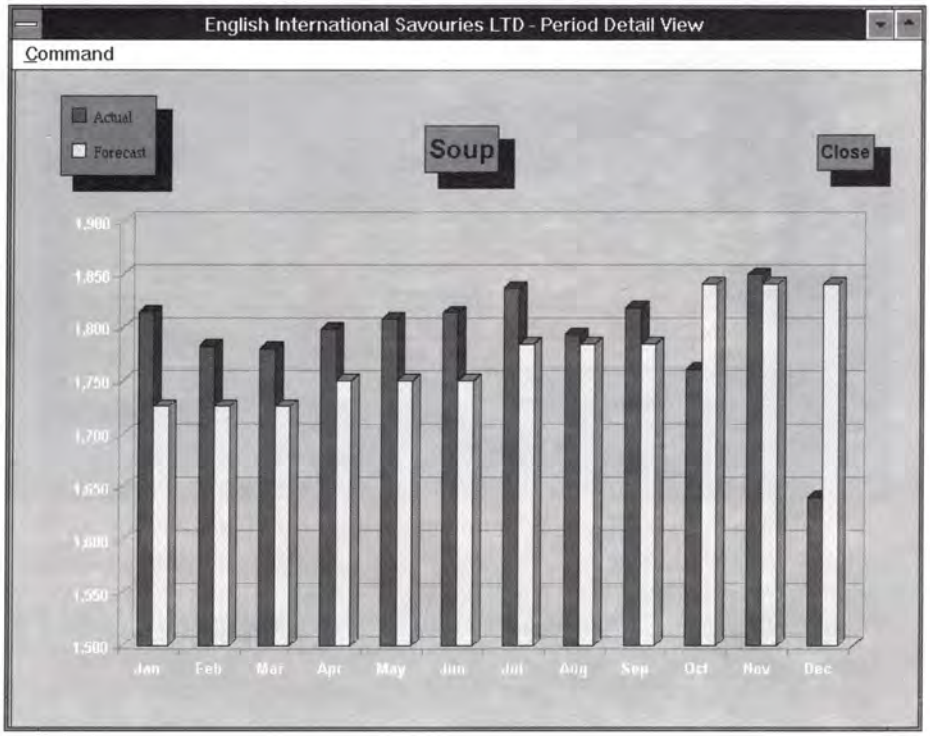


If the user presses the Back button in the regional Soup sales screen, the EIS returns to the Sales By Product table.

Now suppose the user once again double-clicks the word Soup in the Sales By Product table, but this time they choose Periods in the Drilldown dialog box. The following breakdown of yearly soup sales by month appears.



Notice in the preceding Monthly Detail screen that the user now has the option of clicking buttons to perform various actions. Graphs of the data can also be created. For example, if the user clicks the Trend button, the following bar chart appears.



The preceding chart shows the same data as the Monthly Detail table, only in graphical format. This illustrates another principle of a good EIS design: giving users the option of viewing data graphically or numerically, either on the same screen or on different screens. Numeric tables are better for showing precise figures while charts are better for visualizing data and highlighting trends.

When the same data is presented on one screen in both numerical and graphical format, color can be used to mark the association between the rows and columns in the table and the graphics (such as bars or pie slices) in the chart.

Also notice on the Monthly Detail screen that the user can click buttons to automatically create a monthly status report, or send electronic mail to the Soup product manager. The report creation capability starts Microsoft Word, automatically addresses the report using the Word field capability, and embeds the Microsoft Excel table in the document. The E-mail button opens a Microsoft Mail form, so the user can send a message (with the table attached) to other users.

Using different desktop applications to provide different features of an EIS illustrates the synergistic effect that can be obtained by integrating several separate applications together into a single application.

Easy Navigation is Critical

The most successful EIS solutions allow fast and easy navigation through the many different data screens. In the example EIS discussed in this chapter, notice that there is a button (typically Close or Back) on all the screens that allows users to go back to a previous screen. In this manner, users can always retrace their steps. If necessary, they can continue to press the Back button, which eventually takes them back to the home screen, but simply selecting Home from the Command menu will take them directly to the first screen.

Another feature that can make navigation faster and easier is providing a "history" (or log) of previously selected screens. With this capability, the user chooses the desired screen from a list of previously viewed screens and the program "jumps" directly there.

Although a hierarchical "stack" of screens is used for some EIS applications, users should be able to navigate in directions other than up and down through a linear screen hierarchy. The paths a user can take from any particular screen should be determined by the type of data the user would most likely want to see after viewing the current screen.

Chapters 5 and 6 describe additional factors involved in the design of an EIS user interface.

Key Design Points

In the example EIS discussed in this chapter, there are two key design considerations that should be noted:

- Although this example looks and behaves like a completely custom EIS that was programmed from scratch, it was actually created entirely with Microsoft applications. Using the Microsoft Excel customization capabilities and macro language, the major features of this EIS were created in a relatively short amount of time. For example, the buttons were created simply by clicking the button tool on a Microsoft Excel toolbar. The EIS developer then wrote a macro and "attached" it to the button. When the user clicks the button, the macro performs the desired action.

  All of the built-in capabilities of Microsoft Excel, such as charting and analysis functions, are already programmed and ready for EIS developers to use.

- The example EIS consists of an *integrated* set of Microsoft Excel, Microsoft Word, and Microsoft Mail. "Integrated" means that these applications are able to interact with each other in a way that is completely transparent to users. For example, when the user clicks the Email button on the Form View screen, Microsoft Mail is automatically started, the message is automatically addressed, and the sales table is automatically embedded in the message. The user doesn't have to know how to switch between applications, or even know how to use the applications.

These design points help illustrate the Microsoft Open EIS approach. Using a broad range of standard Microsoft applications as your tool set, you can create custom EIS solutions with greater power and flexibility than those created using traditional single-package EIS solutions.

In many traditional EIS applications, the developer is responsible for programming the presentation of information to the user as well as the feature set that gives the user the ability to edit and manipulate the data. With Microsoft's Open EIS approach, many of these features are already provided by standard applications and require no programming by the EIS developer.

Another disadvantage of many traditional EIS solutions is that the data is "locked" inside the application. Users frequently find it difficult or impossible to transfer EIS data to their desktop applications where it can be further manipulated or personalized. In contrast, if an EIS is built with Microsoft applications, the user can take advantage of these applications' ability to easily exchange data with other applications in the Windows environment. Users will thus find it easier to work with the data in a manner with which they are already familiar.

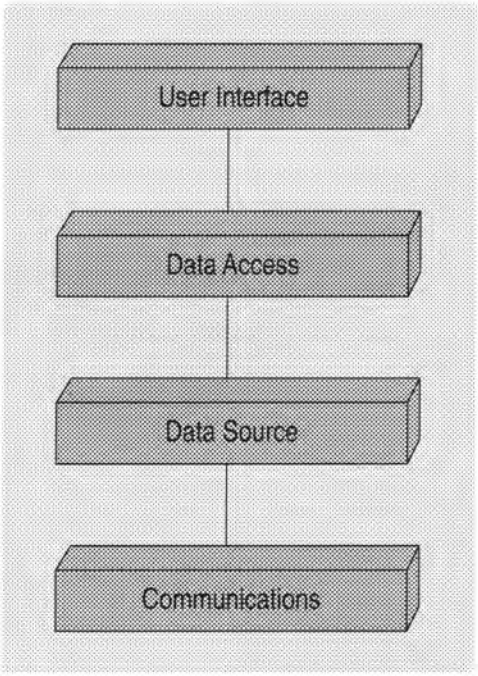
See For Yourself

The Microsoft Open EIS Pak includes a number of working examples that you can run. All of these require Microsoft Excel. Before reading further, you may want to take a few minutes to explore these examples to get a better understanding of how EIS applications work. If you received this guide separately from the Microsoft Open EIS Pak, you can call the number for Microsoft Inside Sales given in Chapter 13 to find out how to order the Microsoft Open EIS Pak.

CHAPTER 3

The Layers of an EIS Solution

Microsoft's approach to EIS is based on an open, four-layer architecture as shown in the following illustration.



These layers offer a useful way to visualize virtually all EIS solutions. The architecture is open because it supports not only Microsoft's products, but a wide variety of products from independent software vendors. In the next few sections, each of the four layers will be described in greater detail.

The User Interface Layer

The user interface layer specifies how data is delivered to users. It is where the user visualizes and manipulates data. This layer involves menus, buttons, dialog boxes, and other elements of a custom EIS interface. In addition to providing the "controls" necessary to interact with data, this layer also handles the formatting of information such as tables and charts so they can be presented to the user.

The user interface layer provides features so you can "personalize" information using desktop applications such as Microsoft Word or Microsoft Excel. One or more applications can be used within this layer, and these applications can be completely customized to provide a unique EIS user interface.

The Data Access Layer

The data access layer is responsible for getting information from various sources such as external databases or local applications and making it available to the user interface layer. It is also responsible for tying multiple applications together into a more cooperative unit that offers greater power to users.

Two technologies that form the core of the data access layer are dynamic data exchange (DDE) and object linking and embedding (OLE), which specialize in linking data from different sources together. New standards such as the open database connectivity (ODBC) and messaging API (MAPI) will also become a central part of this layer.

In some cases, the user will interact directly with the technologies of the data access layer (such as copying and pasting a Microsoft Excel chart using OLE). In other cases, this layer will work in the background and remain completely transparent to the user.

The Data Source Layer

The data source layer is where data is stored so that it can be delivered to an EIS through the technologies of the data access layer. In Microsoft's EIS architecture, the term "data source" has a very broad definition and can mean data stored almost anywhere. For example, Microsoft Project could be considered a data source because it is a source of information to other applications (you can embed objects such as resource charts in other documents). Other examples of data sources include external databases such as SQL Server, DB2®, and Oracle®; or other products that support the data access layer. Most desktop applications such as Microsoft Word and Microsoft Excel can also be considered data sources.

The Communications Layer

The communications layer is responsible for communicating EIS data and other information to users of the EIS application. The primary technology of this layer is electronic mail, which can be used to send documents across a network to other EIS users.

Networking technologies that can be incorporated into this layer are file transfer, messaging, and event notification. These capabilities are typically part of the local area network facilities, and are available using networks such as Microsoft LAN Manager.

The remainder of this guide is dedicated to describing these layers in greater detail and offering advice that will facilitate the successful implementation of an EIS based on this four-layer model.

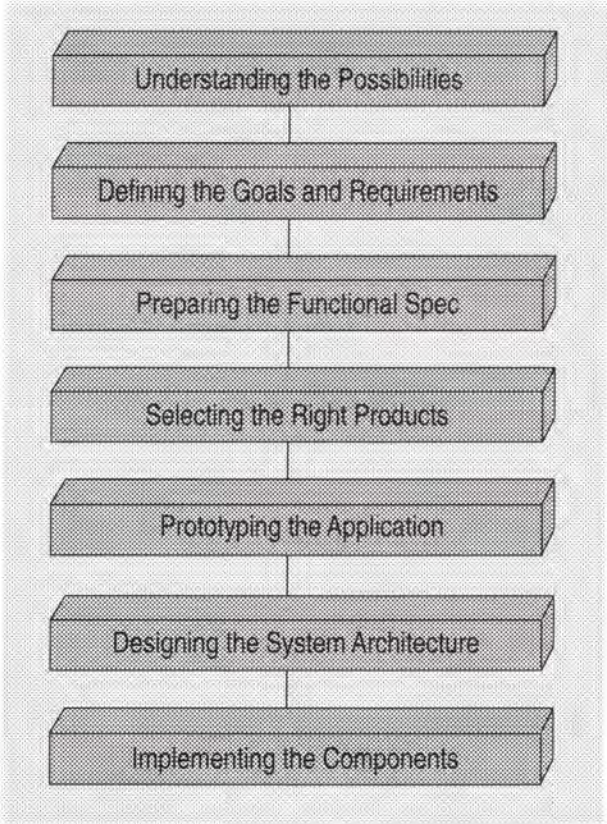
CHAPTER 4

Building EIS Applications — A Process Overview

Building an EIS application is no different from any other software development effort. Proper planning and design almost always lead to a more successful implementation. This chapter describes the major steps involved with planning and building an EIS, and provides a few tips on how to best approach your implementation.

The Basic Steps

When building an EIS, it is usually wise to follow a series of steps that gradually result in a fully implemented product. The more important steps are shown in the following illustration.



These steps represent a generalized model. The actual steps involved with building an EIS usually vary with the scope of the system. Small-scale EIS systems that offer only a few features (such as displaying product sales figures) can often be built using a less rigorous and more ad hoc approach. However, medium- and large-scale systems almost always require a more structured process.

The remainder of this chapter describes in greater detail each of the steps shown in the previous illustration.

Understanding the Possibilities

The first step in developing a successful EIS is to understand what an EIS can do—and what it can't do. It makes little sense to begin designing a solution without knowing what EIS is all about. As a start, you can read through this guide, and look at the videotape and sample applications provided in the Microsoft Open EIS Pak.

When surveying different systems, focus on all the layers discussed in Chapter 3. Although the user interface layer is important (and the easiest to visualize), the data access and data source layers usually define the boundaries of what is possible when you implement your system.

For more information on what can be achieved with EIS, you can contact some of the resources listed in Chapter 13.

Defining the Goals and Requirements

Once you know what you can do with an EIS, the next step is accurately determining the business goals and user requirements of the system. Specifically, how will an EIS help your business? Which features will be most valuable to users?

Although EIS systems can vary greatly, most of them have a common set of general business goals, which include to:

- Quickly obtain basic operational data such as number of employees, product pricing, current budget allocations, and project status.
- Clearly see how an enterprise is performing in selected areas such as sales, manufacturing, or finance.
- Highlight important trends in an enterprise's performance, project current trends to predict future performance, and allow "what-if" data modeling.
- Alert management and analysts to real or potential problems. Also highlight opportunities to enhance good performance.

In addition to these general business goals, you can define a further set of specific requirements for your EIS. These specific goals state how the EIS applies to your particular business. A few examples are to:

- Track rate of sales against current inventory to predict when warehouse shortages might occur and how they can be prevented.
- Track sales volumes on a per-city basis to uncover areas where product interest is highest and where marketing budget should be directed.
- Display manufacturing costs on a monthly basis to uncover seasonal shifts in cost of materials and help manage materials purchasing.
- Display total employee compensation disbursement broken down by engineering, management, and marketing staff, to show how human resource investment is allocated.

This step should be largely user-driven. The eventual users of the system should be responsible for determining most of the system goals and requirements.

Preparing the Functional Specification

Once you have a clear idea of your EIS goals and requirements, you can prepare a functional specification. The functional specification describes your EIS at the functional level, describing the features of the system and how the features support the general and specific business goals. For example, the following is a sample section from a functional specification.

Feature 3-1:	Drilldown from regional sales chart to display sales by city.
Goal supported:	See requirement number 5-3 in EIS Requirements document.
Description:	On the regional sales chart (see Feature 2-6), the user can click a button with a caption of "By-City." This will display a new view with a vertical bar chart showing sales in the six cities where product distribution occurs. Using a toolbar, the user will have the option of changing to a different chart format, such as a pie chart. The By-City view will have a "Back" button to return to the Regional Sales view, and a "Help" button that displays a Help screen on how to use the interface.

Note that the preceding feature description is just a few sentences. Since prototyping the application often changes the original feature definition, there is usually no need for greater detail at this point.

When you have completed the functional specification, be sure to circulate it for review among other key individuals involved in building the EIS solution, and especially among key users of the system. Creating an EIS application should always be a team undertaking, and getting the approval of others on the team is essential.

Selecting the Right Products

There are many ways to implement an EIS, and a variety of applications and software tools are available. A common mistake is choosing one or more tools without fully understanding what they can do or how they will relate to the features in the functional specification.

Comparing the features of a wide number of products against your EIS specifications is a good idea. When selecting a product, consider not only whether the product supports a given feature, but to what extent that feature can be used programmatically, and how difficult or easy it will be to achieve the desired functionality.

This step is often performed in parallel with designing the system architecture (see "Designing the System Architecture" later in this chapter) since much of the system design depends on the available tools.

Prototyping the Application

In many cases, developers and users are new to EIS. Although the functional specification describes the system features, it may be useful to construct a prototype of some of the key system components shortly after the products are selected. A prototype enables developers to fully understand how easy or difficult it will be to implement some of the features of the EIS. It can also give users a chance to comment on the usability and usefulness of the design, and assess the fit between the software tools selected, the functional specification, and user needs.

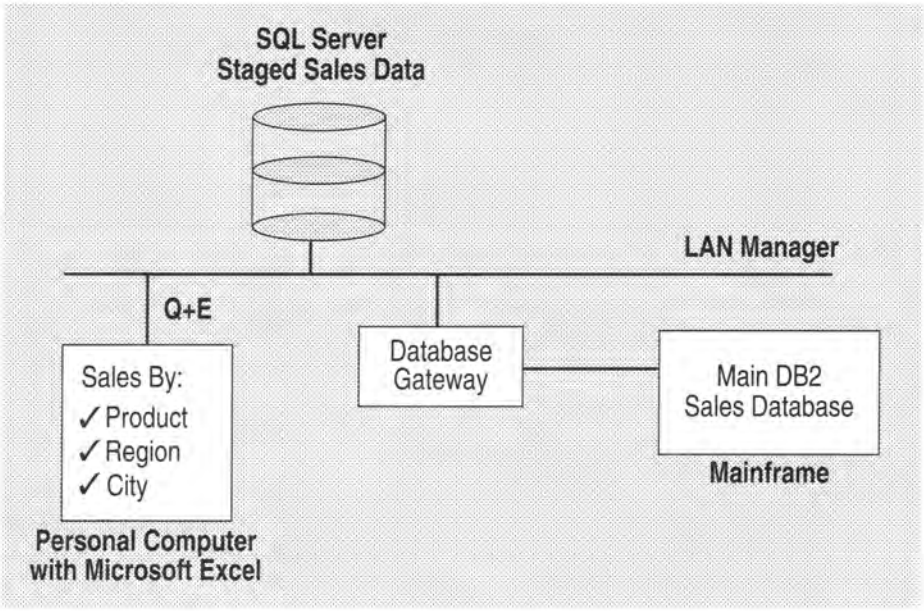
Although prototyping is usually done after product selection, is can be a useful exercise at almost any stage of EIS development.

Designing the System Architecture

For small-scale systems, you can usually begin implementing an EIS solution after you have written the functional specification. Larger systems involving hundreds of client personal computers and multiple sources of data in remote locations usually require an additional document called the system architecture specification that describes how the EIS system will be implemented as a complete system.

For each major feature described in the functional specification (such as viewing sales data), the system architecture specification briefly states how that feature will be implemented, including which software tools will be used to build it, and how the data will be accessed locally or over a network.

Block diagrams showing the links between systems are an important part of the system architecture specification. For example, the following illustration shows how a Microsoft Excel user on a personal computer can access data on a mainframe by transferring the data to a database gateway that can route the data to Microsoft SQL Server, and eventually to Microsoft Excel using Q+E. Chapters 8 and 9 discuss some of these topics in greater detail.



The system architecture specification is also where you can describe application integration at the user interface and data access layers. For example, how will project data from Microsoft Project be made available to an EIS that uses Microsoft Excel as the primary user interface? Will you use OLE, DDE, or a combination of both?

Implementing the Components

After performing the preceding steps, you should have a good idea of what your total EIS design looks like. The next step is to begin implementing the various layers. Always start with a few prototypes. For example, if you are using Visual Basic to design a query tool that retrieves data from SQL Server, build one or more prototypes using the Visual Basic interactive screen design tools. Test these prototypes to determine which ones best address the needs of the EIS users; then begin implementing them.

CHAPTER 5

Designing the User Interface

The user interface is the most visible part of your application. More importantly, it is where the power of your EIS application is put into the hands of users. As a result, the best EIS solutions are those that deliver the right data to the right people quickly and easily, present information in a meaningful way, and offer tools to facilitate browsing and manipulation of the information.

This section provides a few pointers that can help you design the right user interface for your application.

A Few Design Goals

Although every EIS is different, almost all share a few user interface design goals:

- Limit the number of steps required to get to the desired data. A typical view of data (such as a sales chart) should require no more than three or four steps to display.
- Although documentation should be provided, an EIS user interface should be simple enough for a novice user to operate most of the major features without documentation. Clearly marked buttons on each data view are usually the most effective way to achieve this. You may also want to offer online Help to explain some of the details of various features.
- Data should be presented in the clearest and most easily understandable manner. Clearly labeled graphics such as charts should be used whenever possible, and users should be able to quickly spot significant trends and relationships. For example, you can use color to highlight data trends that may represent potential problems.

You may want to run the sample applications provided with the Microsoft Open EIS Pak to get some ideas on user interface design. Another source of information on user interfaces is the *Microsoft Windows Application Design Guide*, available from Microsoft Press. See Chapter 13 for information on how to contact Microsoft Press.

Choosing the User Interface Model

The broad variety of Microsoft applications and tools gives you several different user interface models to choose from. The most common models are:

- Single application.
- Multiple applications with centralized control.
- Multiple applications with decentralized control.

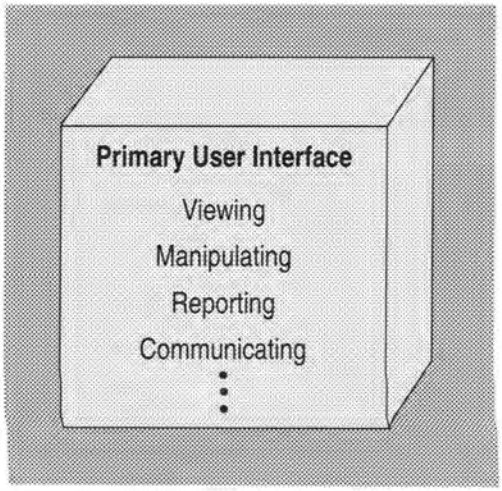
Single Application

The simplest model to implement is the single application model. In this model, a single application is used for everything, including viewing, manipulating, printing, and communicating data.

The primary benefit of the single application model is simplicity of development. As an EIS developer, you need expertise only with a single product, and implementing features is easier because you work in a single environment without the need to integrate multiple applications.

The primary drawback of the single application model is that less power and control can be achieved than with multiple applications. Although products such as Microsoft Excel and Microsoft Access are extremely robust, they don't necessarily offer all the features required by your EIS. For example, Microsoft Excel can be used to display financial tables and charts, access information in remote databases, and even print reports, but if your reports are multipage documents with a high degree of formatting, you will probably want to integrate Microsoft Word into your EIS.

If your EIS needs are fairly simple, you should try using a single application and migrate to multiple applications as needed. For example, if you want an EIS to track sales data by product, location, and sales person, then the single application model is a good choice.



Single Application Model

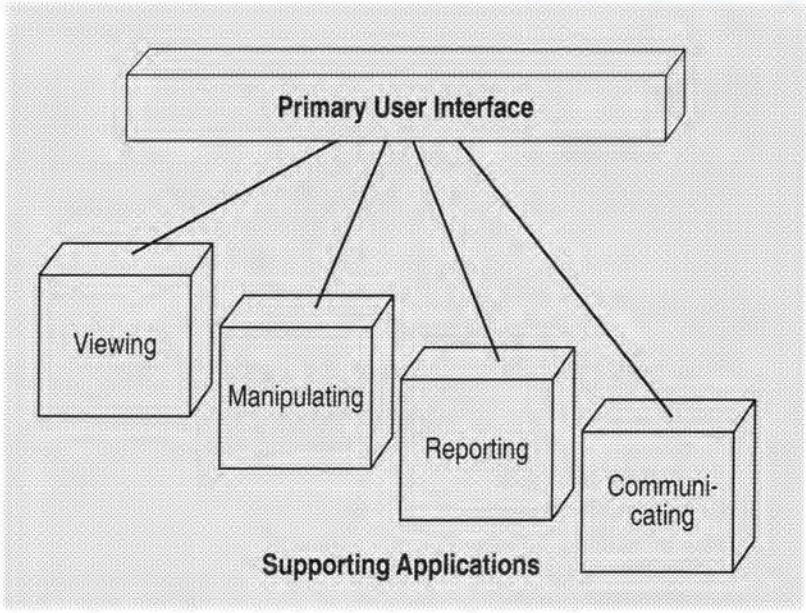
Multiple Applications with Centralized Control

In this model, a single application is used as the primary user interface, but additional products may be used to provide supporting features. These additional applications can run in the background (totally hidden from the user), or can be integrated into the user interface design. Most of the time, the user interacts with a single application.

For example, you can use Microsoft Excel as the primary user interface, and integrate Word and Visual Basic into the design to provide document processing features and visual displys not available in Microsoft Excel. These supporting applications are not treated as peers of the primary application.

Although multiple application models are not as easy to implement as the single application model because of the greater knowledge and integration requirements, this design provides a higher degree of power and control.

You can use this model for medium- to large-scale EIS applications that deal primarily with a single type of information (such as financial data or project data).



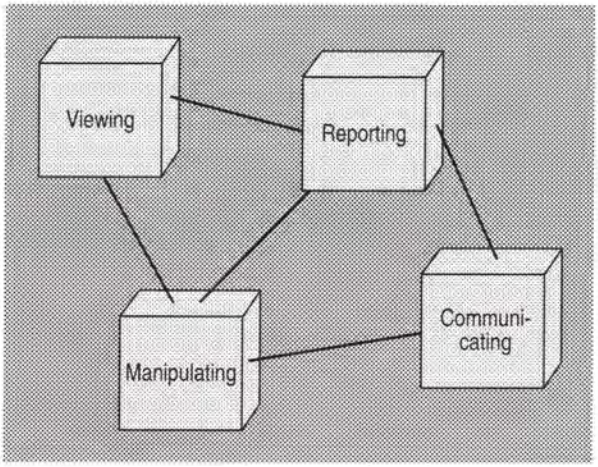
Multiple Applications with Centralized Control

Multiple Applications with Decentralized Control

In this model, you use multiple applications that work together as peers. No single application acts as the "control center" and the user interacts with different applications for different purposes. For example, there might be two primary interfaces such as Microsoft Excel and Microsoft Access that have equal weight in the user interface. Other applications can be integrated to support additional functions.

Most likely, there will be a "home screen" that exists within a single application, but this merely serves as a launching point for other applications. To minimize the amount of user knowledge required, each application should be customized with a simple interface that provides buttons, toolbars, or a limited number of menu choices.

Although most EIS applications will be written using the centralized control model, this model is useful for building EIS applications that deal equally with many different types of information, without emphasizing any one type. For example, if you are building an EIS that integrates financial, project, and human resources data, this model may be more suitable in some cases than the centralized control model.



Multiple Applications with Decentralized Control

Start Simple

Since there are many options, you should take the time to select the one that best fits the business goals of your system and the requirements of the users. It is usually best to start with simpler models and implement greater sophistication as needed.

Choosing the Development Tools

Once you've chosen the basic model of your user interface, you can select the tools you'll use to create it. Microsoft's Open EIS approach enables you to select from an entire spectrum of Microsoft products, including applications such as Microsoft Excel, Microsoft Access, Microsoft Word, and Microsoft Project, and development environments such as Visual Basic and Microsoft C/C++.

As mentioned in the introduction to this guide, the main strength of the Open EIS approach is that you can select the exact tools needed to do the job, and build an EIS with far greater flexibility and power than can be built with traditional EIS product offerings.

Start with Existing Applications

In most cases, the best approach is to start your development effort using "off-the-shelf" desktop applications. In this approach, you rely solely on the customization features, intrinsic functions, and macro languages of these applications to build your entire EIS. Using existing applications offers a fast and simple means of building an EIS, and allows you to tap directly into the full power of Microsoft applications.

Since desktop applications contain a broad range of built-in features, the amount of programming you must do is minimized. With less programming, an EIS can be easier to maintain and enhance than solutions built entirely with programming environments.

As you implement your EIS using these applications, you may find that some features will be difficult to implement using only the customization and macro programming capabilities of the applications. In this case, consider implementing these features with Microsoft Visual Basic or Microsoft C/C++.

Combining desktop applications and programming environments to build your EIS is simplified through a number of built-in capabilities and specialized tools. For example, C programmers can use the Microsoft Excel API to access a wide range of Microsoft Excel features through a C-language API. Visual Basic programmers can use various application APIs such as Microsoft Mail APIs to provide electronic mail or SQL Server APIs for direct database access. Programmers using the WordBasic macro language of Microsoft Word can call procedures in dynamic link libraries (DLLs) from their macros. There are many other examples.

Chapter 6 provides greater detail on how applications and programming environments can be used in an EIS.

Determining the Selection Criteria

Deciding which development tools to use depends on a number of factors. A few of the more important criteria are:

- **Feature set** How much of the desired functionality or user interface features are already available using standard desktop applications? Most EIS applications are designed to present charts (Microsoft Excel), text (Microsoft Word), forms (Microsoft Access and Microsoft Visual Basic), and data (Microsoft SQL Server and Microsoft Access) to users. By using these application capabilities, development and support costs are likely to be much smaller than with purely custom-coded solutions.
- **The implementation time frame** How soon must the EIS be placed in service? If the deadline for delivery of the EIS is relatively short, then using applications as the primary development vehicle is almost always the wisest choice. Less immediate deadlines may leave more time for programming using a traditional programming environment.
- **Performance** The performance requirements of some EIS features may necessitate the selection of one tool over another. You may have to experiment with various prototypes before determining which development tool yields the desired performance.
- **Technical expertise of the implementors** The background of the EIS development team usually plays a role in determining which products to use. A team consisting of seasoned programmers may want to use different tools than a team with less programming experience.
- **Future growth** Using desktop applications as the primary building blocks automatically places your EIS on a future growth path. Each time a new release of an application occurs, your EIS (and its users) automatically inherits the new functionality.

Integrating Multiple Applications

If you use more than one application for your solution, consider how these applications will be integrated to form a single EIS user interface. A few guidelines you may want to follow are to:

- Minimize the number of times users must switch between applications as they use the EIS. Two technologies that facilitate this are dynamic data exchange (DDE) and object linking and embedding (OLE). These technologies allow applications to exchange data automatically so users don't have to be concerned with moving between applications to view or update data. For example, using OLE, you can paste link a Gantt Chart from Microsoft Project into Microsoft Excel so that the user can view the most recent version of the chart without leaving Microsoft Excel. DDE and OLE are discussed in detail in Chapters 7 and 8.
- Maintain a consistent "look and feel" across all applications in your EIS. For example, you may want to use the same custom menu bar in each application, or use buttons, custom toolbars, or other controls that look and work the same way in each application.
- Minimize the amount of knowledge users must have about how to operate the underlying applications. A successful EIS will bring the full power of applications to the user without exposing all the details of how to use each of the application's native features.

Which Applications Should You Use?

The type of application you choose for the user interface depends on many of the preceding considerations, as well as your business objectives and user requirements. The wide variety of Microsoft applications provides a number of options, but a typical EIS solution is based on one or more of the products listed in the following table.

Application	Used for
Microsoft Access	Summarizing transaction data, managing relational data, managing large tables, database reports, forms, main user interface components.
Microsoft Excel	Charting, analysis, financially-oriented reports, main user interface components.
Microsoft Mail	Workgroup and enterprise communications.
Microsoft PowerPoint®	Graphics, electronic film, presentations.
Microsoft Project	Project scheduling and tracking, resource tracking, Gantt and PERT charts.
Microsoft Visual Basic	Custom functionality, specialized tools not provided by applications, control over the Windows environment, custom graphics.
Microsoft Word	Text and document management, reports, hard copy communications.

You can integrate two or more of these applications to build almost any type of EIS. However, as mentioned earlier in this chapter, it is usually wise to choose one of these as the primary user interface and switch to other applications when special features are needed. The primary user interface should be based on the main type of data your EIS will manipulate. For example, if your data is primarily financial, you might consider using Microsoft Excel or Microsoft Access for the primary user interface. If your data is primarily project-oriented, then Microsoft Project may be the best choice.

Often, the decision to use one application over another is made after a certain degree of prototyping has occurred. You may want to take some time to model some EIS features with a few of these applications to see which are suitable for your needs.

Microsoft Excel and Microsoft Access Are Often the Best Choices

The majority of EIS applications use financial data and/or relational database data as the main type of information. Because of this, Microsoft Excel and Microsoft Access are often the best choices as the primary user interface for many EIS applications. Microsoft Excel is designed to handle numerical data and offers strong charting and financial analysis capabilities. Microsoft Access is designed to handle large tables of data, and tables that have relational links between them. Also, both Microsoft Access and Microsoft Excel have flexible user interface building tools.

Microsoft Excel and Microsoft Access are complementary in their functionality, and you can use one or both. The different circumstances under which you may want to use these applications are discussed in Chapter 6.

CHAPTER 6

Implementing the User Interface Layer

Traditionally, building a completely custom user interface layer for an EIS can involve a large amount of custom programming. As a result, the entire development process can take months or even years.

Microsoft offers a different approach. With a broad range of tools, you can develop a complete EIS user interface in a fraction of the time it takes using traditional methods. Since applications such as Microsoft Excel, Microsoft Access, Microsoft Project and Microsoft Word are highly customizable, you can build complete EIS front ends using little programming.

If needed, you can supplement the customization features of Microsoft's applications with programs written in high-level macro languages or Basic-based development environments such as Visual Basic.

This chapter explores some of the available options for building an EIS user interface.

Using Microsoft Excel in Your EIS

Most corporate data is stored in table format. This data includes payroll, sales revenue, manufacturing status, human resources statistics, and many other types of information.

A major goal of most EIS applications is to present this corporate data to users in a graphical and easily visualized format that highlights overall trends and enables fast interpretation and analysis.

One application that can bring these capabilities to an EIS is Microsoft Excel. Microsoft Excel is a powerful spreadsheet application that offers a wide range of features for sophisticated data presentation, formatting, analysis, and charting.

In addition to these features, Microsoft Excel offers complete customization capabilities that greatly facilitate the development of EIS applications. Among the customization capabilities that will be described in this chapter are:

- The Microsoft Excel EIS Builder.
- Native customization capabilities of Microsoft Excel.
- Macro language of Microsoft Excel.
- A combination of the above.

Each of these options is examined more closely in the following sections.

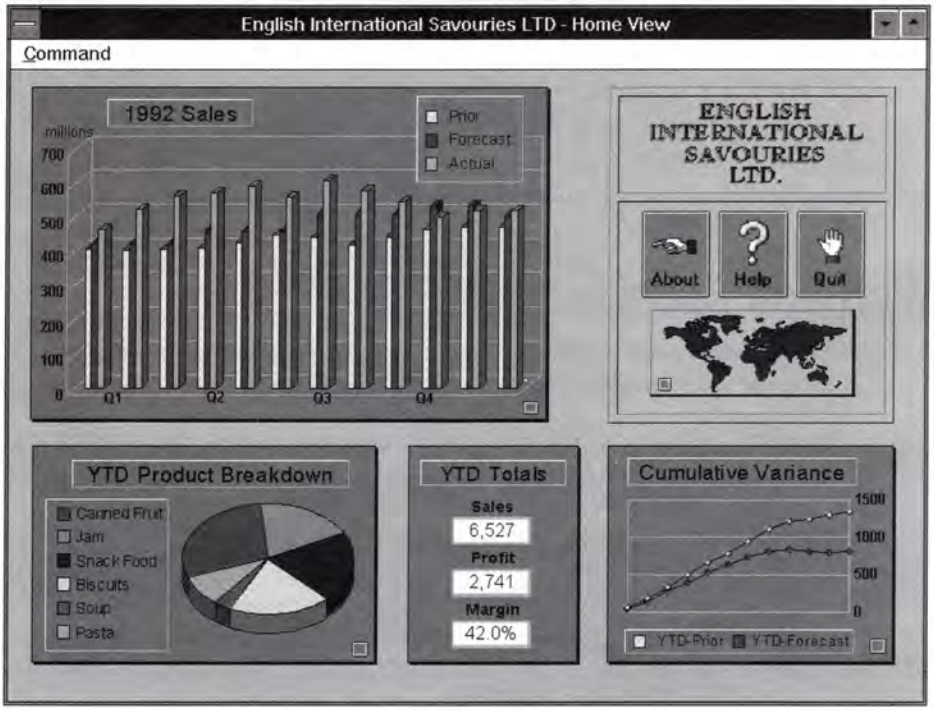
The Microsoft Excel EIS Builder

The Microsoft Excel EIS Builder is a Microsoft Excel add-in application provided with the Microsoft Open EIS Pak that allows you to build Microsoft Excel-based EIS user interfaces. The three major benefits of this add-in application are its speed of development, minimum knowledge requirements, and its ability to incorporate the powerful analysis and charting features of Microsoft Excel.

With the Microsoft Excel EIS Builder, you can take data that already exists in Microsoft Excel worksheets (or portions of worksheets such as tables and charts) and tie this information together into a complete EIS—with no macro programming.

You can easily build the navigational capabilities that allow users to move between screens consisting of tables, charts, and other information. You can also create custom buttons and menus so users can select various options or execute commands.

As you create your user interface, the Microsoft Excel EIS Builder automatically generates the application that handles the user's interaction with the interface. This application generation occurs "behind the scenes," and is completely transparent to the EIS developer. The following screen shows an example of what can be created with the Microsoft Excel EIS Builder.



Multiple Views of Your Data

You manipulate and display data with the Microsoft Excel EIS builder using the concept of *views*. A view is an area of a worksheet that is to be displayed by the EIS. For example, a view could be a range of cells, a chart, or a combination of cells and graphics. You create views simply by selecting the area of the worksheet that you want the EIS to display and then choosing the View command on the EIS Builder menu.

You can add buttons to views so that users can simply click the button to perform an action such as returning to the previous view or displaying more detailed data. In the preceding illustration, the screen is defined as a single view.

The Power of Scripts

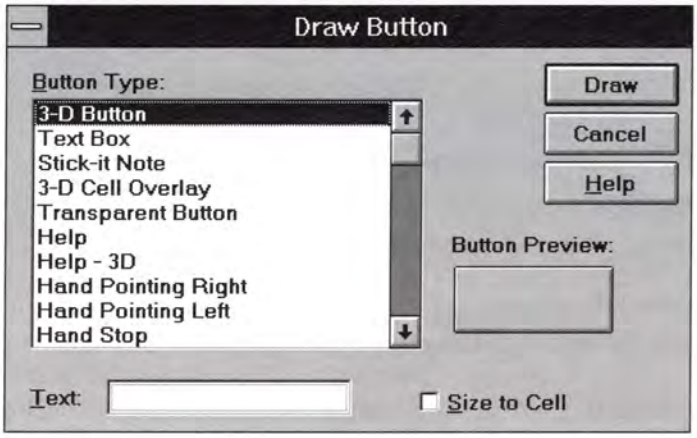
Instead of using macros, the Microsoft Excel EIS Builder uses "scripts." You create scripts simply by selecting actions from a list. These actions have macros associated with them, but they are built in to the system and are completely transparent to the user. You can attach a script to a button, menu item, or other object, so that when a user clicks the object, the script is run.

Using the Microsoft Excel EIS Builder—An Example

As an example of how the EIS Builder works, suppose you wanted to create a new button for your EIS application. After starting the EIS Builder from within Microsoft Excel and opening the project for which you want to add the button, you can follow these steps:

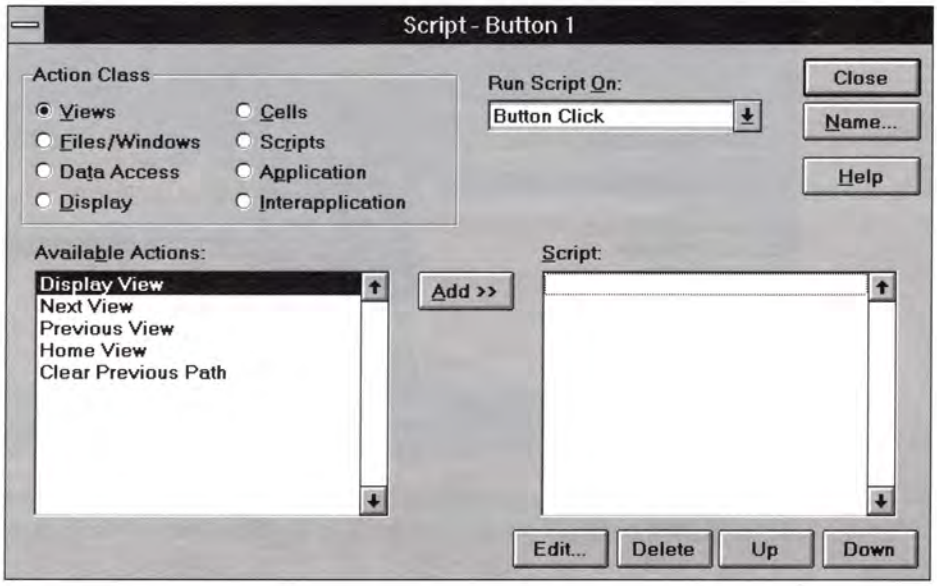
1. From the EIS menu, choose Button.

   The following dialog box appears.



2. From the Button Type box, select a type. Notice the wide variety of button types from which to choose.
3. In the Text box below the list of buttons, type the caption of the button.
4. Press ENTER.

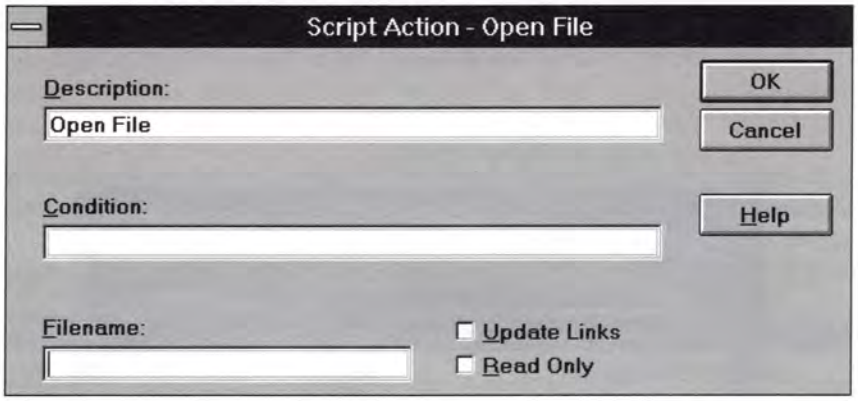
   The following dialog box appears.



In this dialog box, you can build a script by selecting a *class of action* such as Files/Windows, and then selecting the desired action in the list of available actions. Choosing the Add>> button adds the action to the script.

5. Select the Files/Windows button to display the list of available actions associated with files and window manipulation.
6. In the Available Actions box, select Open File.
7. Choose the Add>> button.

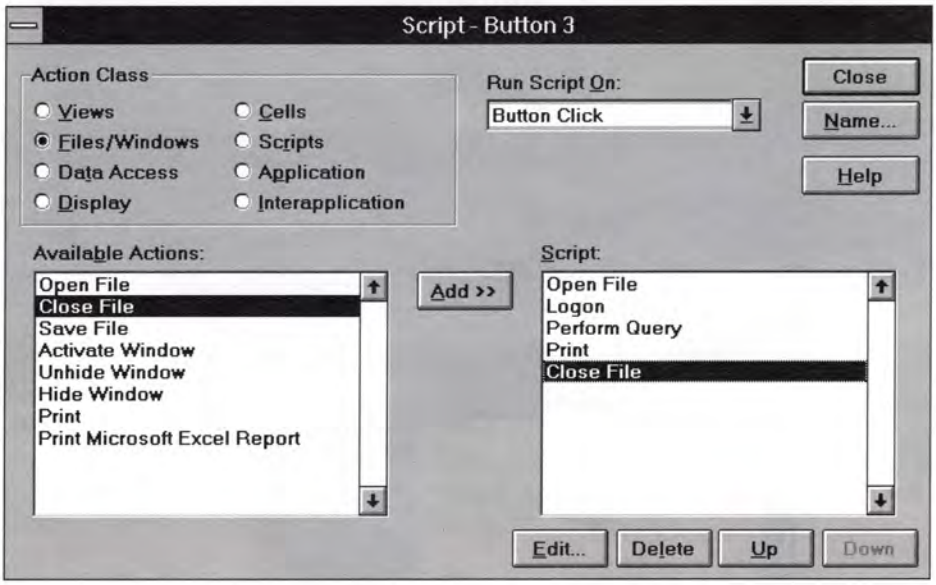
   The following dialog box appears.



8. In the Filename box, enter the filename of the worksheet to open.
9. Choose the OK button.

   This adds the Open File statement to the list of script actions.

The following dialog box shows a complete script that opens a file, logs on to a SQL database, performs a query, and prints the resulting spreadsheet. The script (which was created in less than a minute) is attached to the button you created and is run when the button is pressed by the user.



The most important thing to notice in the preceding example is the speed at which you can develop even complex procedures and add them to your user interface.

A Combination of Tools

The Microsoft Excel EIS Builder can be used by itself to specify an entire user interface without programming a single line of macro code. However, if you need greater control at any time, the EIS Builder allows you to supplement the built-in script actions with your own custom actions that you create with macros. The EIS Builder script library is therefore extensible and can include any macro you could create with the Microsoft Excel macro language.

For More Information

The Microsoft Open EIS Pak includes a copy of the Microsoft Excel EIS Builder, along with full documentation. If you have Microsoft Excel version 4.0 or later, you can start using the EIS Builder to create custom EIS applications. For more information, see the *Microsoft Excel EIS Builder User's Guide*.

If you received this guide separately from the Open EIS Pak, call the Microsoft Inside Sales number listed in Chapter 13. If you are using a version of Microsoft Excel that has been localized for a particular language, contact your Microsoft subsidiary to verify that all the features and products described in the preceding paragraph are available in these versions.

Using the Customization Capabilities of Microsoft Excel

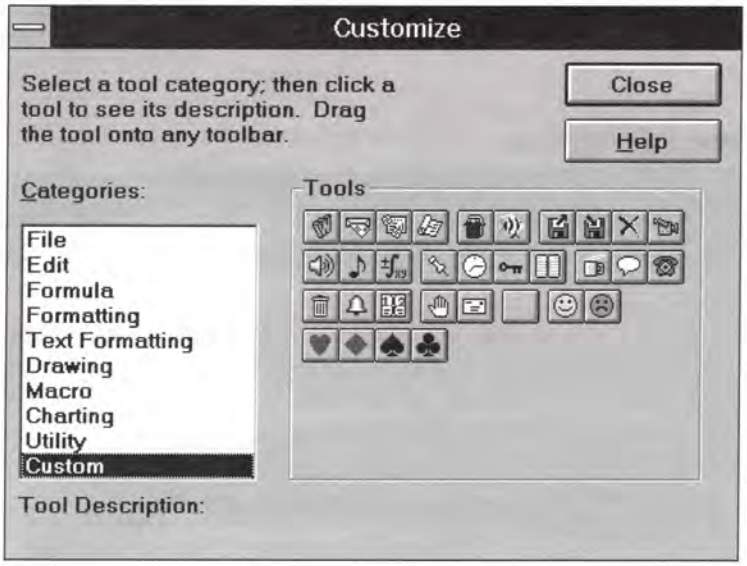
Although you can use Microsoft Excel EIS Builder to create a fully customized Microsoft Excel application, you can customize Microsoft Excel directly without the EIS Builder. Almost every part of the Microsoft Excel user interface can be customized to suit your own needs and create EIS applications that look and feel as if they were programmed from scratch. This section describes some of the Microsoft Excel customization features and describes how they could be used for an EIS application.

Toolbars

Microsoft Excel comes with a number of default toolbars that allow users to perform functions or procedures simply by clicking a button. For example, using the following toolbar, a user could create a chart of the desired format, or change the format of a selected chart.

Toolbars can be useful in an EIS application because they are very intuitive and require almost no knowledge of the application. With Microsoft Excel, you can create a custom toolbar and assign macros to toolbar buttons you create. Each time the user clicks a button, the macro assigned to that button is activated and can perform any action within Microsoft Excel, or even within other Windows programs through dynamic data exchange (see Chapter 7). A number of toolbar button bitmaps are provided with Microsoft Excel, but you can also create your own custom bitmaps to display on the surface of the button.

Custom toolbars can be created completely interactively with no macro programming, or through macros. For example, to add a tool to the standard chart toolbar, you click the toolbar with the right mouse button, select the Customize option in the pop-up menu, drag a tool from the Tools box as shown in the following dialog box, and drop the tool on the chart toolbar.



For more information on customizing toolbars, see Chapter 8 in Book 2 of the *Microsoft Excel User's Guide*.

Using the Macro Language of Microsoft Excel

For the greatest control over your EIS application, you can use the powerful macro language of Microsoft Excel. You can use the macro language to control every aspect of a worksheet programmatically, and access other sources of data in other applications. Later in this book, you'll see the Microsoft Excel macro language in action to access database data, interact with Microsoft Word, and provide other capabilities.

Using Macros

Using macros, you can modify the existing menu bar of Microsoft Excel or create an entirely new menu bar with your own custom menus. This enables users of your EIS system to select menus and commands specific to your EIS application. As with toolbars, you can attach macros to custom menu items to perform actions within your EIS application when a menu item is selected.

To add a menu, you create an area on your macro sheet that contains the information needed for the menu. Then, using the Microsoft Excel ADD.MENU statement part of your macro, the menu and its items are added to the worksheet menu bar. For more information on creating custom menus, see Chapter 8 in Book 2 of the *Microsoft Excel User's Guide*.

Dialog Boxes

You can use a special tool called the Dialog Editor (included with Microsoft Excel) to create custom dialog boxes that can be displayed any time you want to receive input from a user of your EIS application. The Dialog Editor is completely interactive and allows you to create custom dialog boxes by clicking and dragging controls (such as buttons, edit fields, and radio buttons) around your screen.

Once you have created a dialog box, you can copy it and paste it directly into a macro sheet, where it is translated into macro format. You can then create a macro that displays the dialog box when a specific event occurs in your EIS application. For example, the dialog box could be displayed when the user selects a custom EIS menu item, and your macro can be constructed to receive the user's input to the dialog box and take appropriate actions.

Note The Dialog Editor is best used for creating general dialog boxes of any layout, and invoking them from macros. You can use the Microsoft Excel EIS Builder to create specialized user input dialog boxes without macro programming.

Using Microsoft Access in Your EIS

Microsoft Access is an interactive, graphical, relational database management system with which you can combine forms, reports, queries, and modules to create complete relational database applications. With Microsoft Access, you can give EIS users a powerful, intuitive, and highly visual way to store, retrieve, and report on large volumes of information.

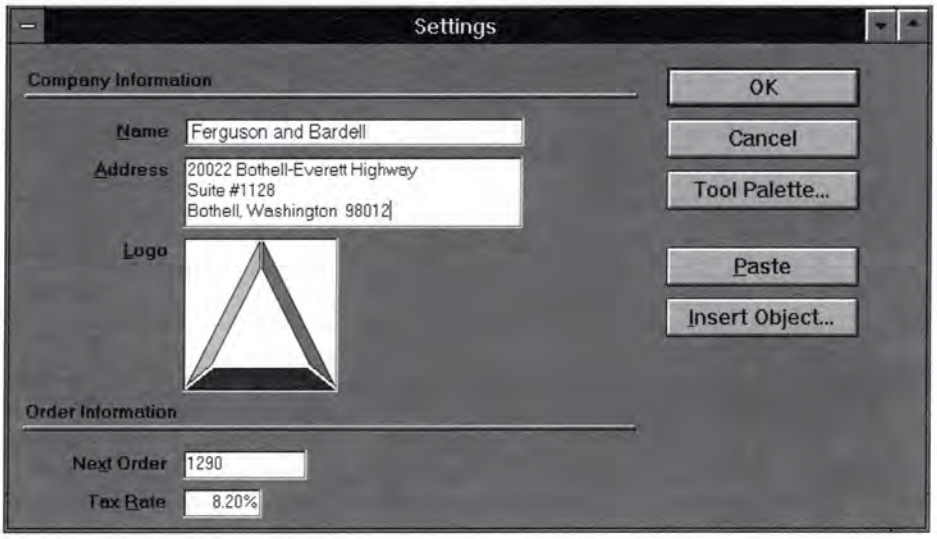
Microsoft Access enables EIS developers to:

- Take full advantage of the graphical nature of Windows to create interactive database front ends that are intuitive, visual, and inviting.
- Seamlessly connect to a wide range of database systems through its graphical query facilities. Connections can be made locally, across different networks, or simultaneously to multiple databases located on different platforms.
- Interactively design reports that can present database data in tabular and graphical formats. You can even include linked objects from other applications, such as Microsoft Excel, in your reports.
- Use a high-level macro language that requires no programming experience to automate various tasks, or use the Basic-based programming environment of Microsoft Access for a high degree of control over data manipulation.

Like Microsoft Excel, Microsoft Access provides a complete environment for developing custom applications that allow users to access and manipulate tabular information. The many features of Microsoft Access, including professional-quality report generation and easily accomplished automation, make it a powerful tool for organizing, finding, and presenting information.

Microsoft Access lets you build user interfaces interactively (with no programming), but also provides a powerful Basic-based programming environment if you need a high degree of control over your application and the data it uses.

With Microsoft Access, you create *forms* interactively by selecting controls such as fields, check boxes, and buttons and placing them in a window. Forms can be used to combine and display data from many different tables stored in different network locations and database formats—at the same time shielding users from the complexities of how and where the data is stored. The following illustration shows a sample form created with Microsoft Access by simply clicking and dragging buttons, text fields, and other controls into a form window.



You can choose tables and queries to include in the form, customize the way the information appears, and customize the menu bar. Once the form is created, you can use the Microsoft Access Basic programming language to attach code to the controls in the form and respond to the user's interaction.

Typically, information is made available to the user with database queries. A Microsoft Access query searches the appropriate tables in the database for the information requested and presents it in a manner dictated by the form to which the query is attached. Queries can be used to combine data from different tables and find exactly the information the EIS user needs.

Microsoft Access gives the user access to information contained in a range of heterogeneous data sources. For example, an EIS built with Microsoft Access could simultaneously access and display information from SQL Server, Paradox, dBASE, and its own relational database system. These different data sources can be tapped directly in their native format, without performing any importing operations.

When Should You Use Microsoft Access?

Before using Microsoft Access, you must decide if Microsoft Access is the best solution for your needs. Microsoft Excel and SQL Server also offer the ability to manipulate data in tabular format, but they each have their own strengths. The following sections compare Microsoft Access with Microsoft Excel and SQL Server.

Microsoft Access and Microsoft Excel

As with Microsoft Excel, Microsoft Access allows users to develop custom forms. In addition, Microsoft Access provides graphical facilities that enable users to create database queries and reports. These queries and reports use a dynamic data dictionary and can simultaneously incorporate information from multiple database platforms.

Making the choice between Microsoft Access and Microsoft Excel is usually easy. If you want to provide EIS users with ad hoc data analysis, charting, and formatting capabilities, Microsoft Excel is most often the best choice. Microsoft Access is usually the best choice for performing operations on relational data; processing transactions; browsing and querying database information; handling voluminous amounts of data; and providing a robust set of user interface controls.

Since the features of these two products are complementary, you don't have to pick one or the other. In many cases, you can take advantage of the unique capabilities of both applications and implement different modules of your EIS using either Microsoft Access or Microsoft Excel.

Note Although Microsoft Excel supports links between cells in different tables, it does not support relational links between columns of data. Relational links are found mostly in database products such as Microsoft Access and Microsoft SQL Server.

Microsoft Access and SQL Server

One consideration that may arise when designing your user interface is whether to use Microsoft Access or SQL Server. Although both are relational database systems, they are quite distinct.

Microsoft Access combines a set of powerful front-end development tools with an RDBMS that is suitable for small- to medium-size workgroups. In contrast, SQL Server is designed primarily as a back-end database server with a large number of high-performance RDBMS features. SQL Server does not come with front-end development tools, but is designed to work with several, including Microsoft Access, Microsoft Visual Basic, and Microsoft Excel.

Microsoft Access also offers an excellent front-end development environment for SQL Server, and these two database products can make a powerful team. You can use Microsoft Access solely as a front end to SQL Server (or via SQL Server to other SQL databases), or you can take advantage of the RDBMS capabilities of Microsoft Access for smaller database applications with up to 15 concurrent users.

Therefore, to build front ends for databases, Microsoft Access is the clear choice since SQL Server is not designed for this purpose. For back-end database functionality, deciding whether to use Microsoft Access or SQL Server depends on a number of factors, as the following sidebar explains.

Consider the Type of Data

When deciding whether to use Microsoft Access or SQL Server as a back-end database for an EIS, an important criteria to consider is the type of data you need to handle.

Generally, Microsoft SQL Server is designed to handle databases that have demanding requirements in the areas of performance, security, administration, and capacity. It can therefore be a better choice for processing large volumes of corporate data or for handling transaction-processing applications such as order entry.

In contrast, Microsoft Access is an ideal solution for databases with small- to medium-scale requirements. It is therefore a sensible choice for databases containing personal, workgroup, or departmental data.

For more information on Microsoft SQL Server, see Chapter 9 or call the number given in Chapter 13 for Microsoft Inside Sales.

Macros and Programs

Microsoft Access offers two ways to build custom applications: macros and programs. In Microsoft Access, a macro is a script that you build interactively by selecting controls and attaching "actions" to these controls. Microsoft Access macros require no programming expertise in the usual sense of the term "macro" and are similar to scripts in the EIS Builder (see "The Microsoft Excel EIS Builder" earlier in this chapter).

For example, you can use the command button tool in the Microsoft Access toolbox to add a button to a form, and then specify which macro to attach to the button in the OnPush property of the button control. When a user clicks the button, the associated macro is executed.

For maximum control, Microsoft Access comes with its own Basic language programming environment that you can use instead of or in addition to macros. Microsoft Access Basic provides a complete development environment that is specialized to processing database data and managing database tables.

Using Microsoft Visual Basic in Your EIS

What is Visual Basic?

Microsoft Visual Basic differs from the other solutions discussed earlier in this chapter because it is not an application. Rather, Visual Basic is a programming system that you can use to build complete applications for the Windows environment.

In Visual Basic, the layout of an application's user interface is designed with a set of interactive graphical tools such as dialog editors. These tools are similar to those provided with Microsoft Access. By selecting controls such as edit boxes, check boxes, and push buttons, you can build a complete user interface interactively.

Once the interface is designed and built, Basic programs can be written to handle the user's interaction with the interface. For example, when a user clicks a button, the program code "attached" to that button will be activated. To use Visual Basic effectively, you need to become proficient in using the programming language of Visual Basic.

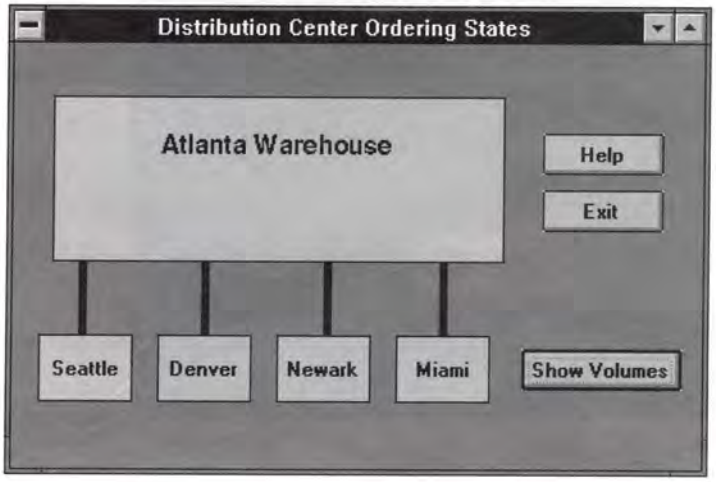
When Should You Use Visual Basic?

Although you can build complete applications with Visual Basic, it is often more useful for building special features that are not available in applications such as Microsoft Excel. Using the powerful programming language and interface design capabilities of Visual Basic, you can do a number of things you cannot do with the customization capabilities of other applications mentioned in this guide, including:

- Use a wide variety of custom controls such as gauges, meters, and grids that can add unique features and visual appeal to your interface.
- Manipulate the Windows environment and perform operations that might be difficult to accomplish in a macro language. For example, direct input and output to text or binary files, custom graphics, and timer events.
- Build complex forms and dialog boxes that include scroll bars, file browsing list boxes, and other controls that are not supported by Microsoft Excel or Word. (Microsoft Access is another good choice for form creation.)

Using Visual Basic—An Example

The following example application was created solely with Visual Basic. It illustrates the flexibility and power of Visual Basic as well as certain features that don't exist in most macro programming languages. When the application is started, the following screen is displayed.

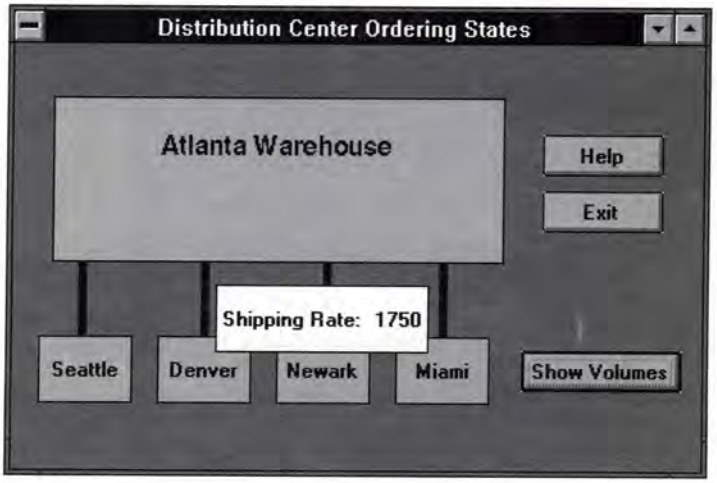


This interface shows a graphical representation of the ordering states of four distribution centers and the inventory state of the warehouse that serves them. When the application is started, it loads the warehouse's inventory, the shipping rates, and the order volume for each distribution center. Lines connecting the warehouse to the centers represent the projected flow of product. The thickness of these lines depends on the order volume, and the color depends on the ability to ship the orders—green if the order volume is less than 75 percent of the shipping rate, yellow if it is less than 90 percent, and red if the order volume is any higher. A manager looking at this graphic can quickly identify a pending order backlog if a line turns red.

The statistics that this application uses are stored and updated in a corporate database. Periodically, the database downloads its information to a personal computer, and the application reads the new data. The frequency at which the data is accessed by the application is determined by the timer control facility of Visual Basic. This control automatically generates an event each time the application must read new data.

If the user wants more information, clicking the Show Volumes button displays the exact order volume of each distribution center and also shows the total inventory above the main warehouse box.

Additionally, clicking the right mouse button on any distribution center displays a small window as in the following screen, which shows the amount of product that can be shipped per week from the warehouse to that distribution center.



How it Works

Visual Basic is an event-driven environment. Each of the graphical objects in the preceding screen (known in Visual Basic as a "form" object) has a number of events associated with it. For example, a button object can be associated with the "click" or "double-click" events.

Events can have code "attached" to them, and this code executes whenever the event occurs. For example, the following Visual Basic code example is attached to the main form's paint event, which occurs when the window needs to be redrawn. The subroutine determines the color code and line width for each distribution center based on the center's current shipping rate and order volume, and then draws the lines that show product flow between the warehouse and the centers.

```
Sub Form_Paint ()
  For I = 1 To 4
    If (CenterVolume(I) < .75 * CenterShipRate(I)) Then
      CenterColor(I) = GREEN
    ElseIf CenterVolume(I) > .9 * CenterShipRate(I) Then
      CenterColor(I) = RED
    Else CenterColor(I) = YELLOW
    End If
    A% = (I - 1) * 1080
    DrawWidth = CenterVolume(I) / 250
    Line (650 + A%, 1920)-(650 + A%, 2650), CenterColor(I)
  Next I
End Sub
```

Other subroutines in the application handle different events and objects. For example, the Show/Hide Volumes button has a subroutine attached to its click event that changes the button caption and displays or hides the volume figures. The preceding code is just one example of the way you can use events to build a Visual Basic application. It also shows how Visual Basic offers greater control over graphics than most macro languages.

SQL Server Programmer's Toolkit for Visual Basic

Visual Basic is an excellent tool for building front ends to databases. You can create special query utilities, data entry systems, and other tools.

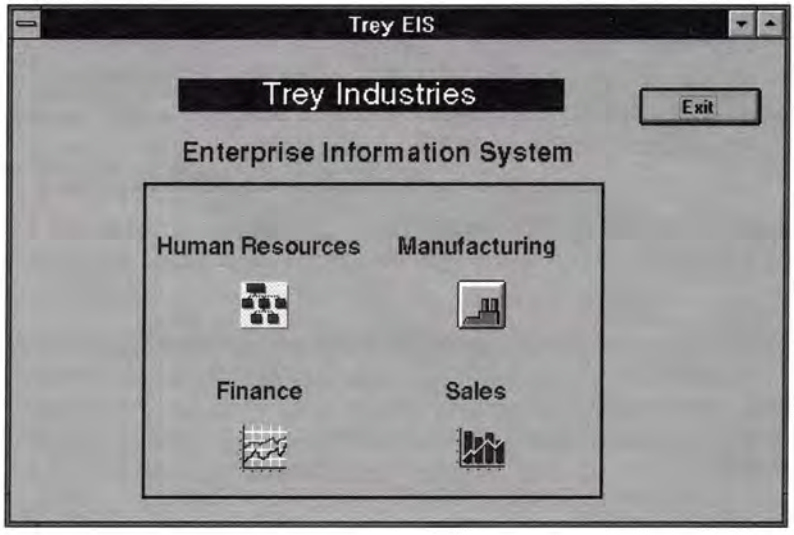
In order to facilitate the process of building front ends to databases that support the SQL language, Microsoft offers the SQL Server Programmer's Toolkit for Visual Basic (VBSQL). VBSQL is a set of functions and routines built on top of the SQL Server DB Library, a library for C programmers. With VBSQL you can:

- Send SQL statements to a SQL Server database.
- Process the results of SQL queries.
- Retrieve and update values from the database directly.
- Manipulate database values from database tables.
- Insert values into a database.
- Move data between SQL Server and Windows applications.

Command Centers

If your EIS provides access to several distinct types of data through different applications, you can use Visual Basic to build a "command center" that starts a specific application depending on the type of data sought by the user. Also called "home screens," these provide a central place for your users to go when they start the EIS or if they want to select what they want to do from several choices.

The following is an illustration of a command center application that was built with Visual Basic.



Support for DDE and OLE

Visual Basic supports both DDE and OLE in its programming language. You can therefore integrate Visual Basic applications with other applications in the Windows environment. For more information on DDE and OLE, see Chapters 7 and 8.

Using Microsoft Project in Your EIS User Interface

Microsoft Project is a planning, scheduling, and tracking application that helps you manage a project's tasks and resources. It includes a wide variety of features for managing projects, such as the ability to allocate and track resources for multiple projects, to compare the current state of a project to earlier states, and to print or plot a variety of reports.

Microsoft Project offers multiple "views" of data, which enable you to choose how information is displayed. Views give you extra flexibility for entering, organizing, and examining data, and enable you to communicate information about a project in a wide variety of formats. With the customization features of Microsoft Project, you can display exactly the information you want, and in the form you choose.

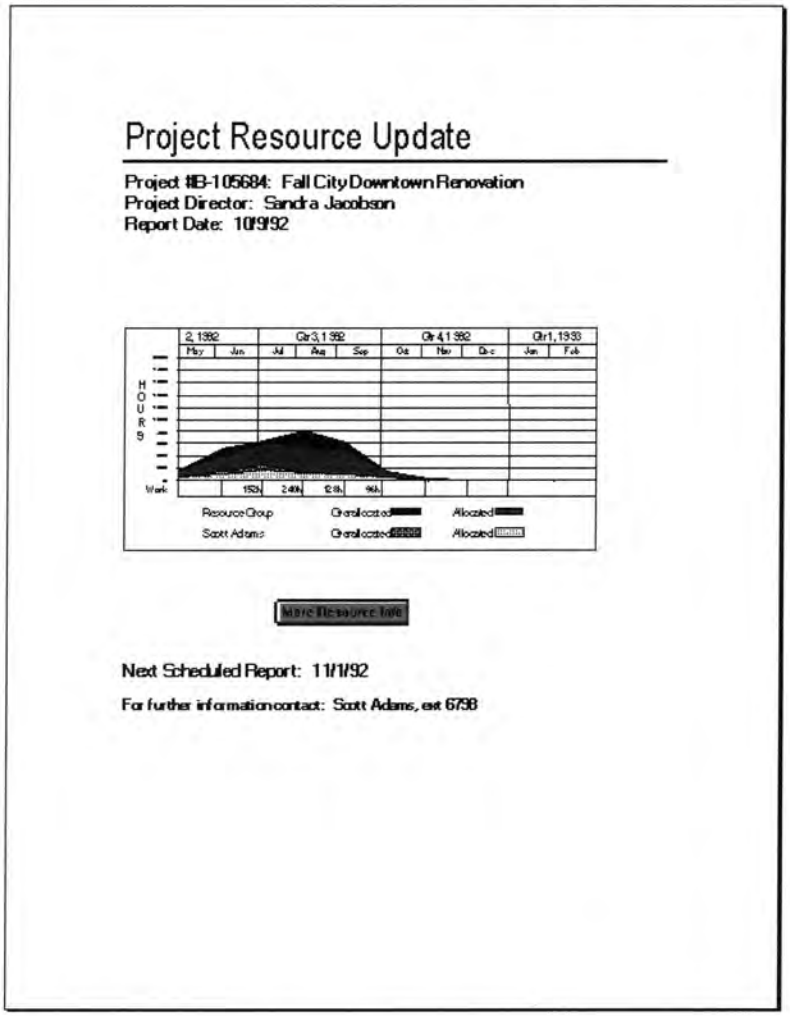
When Should You Use Microsoft Project?

If your EIS needs to access and display information about company projects, then you should consider using Microsoft Project as a component of your user interface.

Microsoft Project gives you a great deal of power and flexibility when managing project-specific data, and you can integrate data contained in Microsoft Project files into other components of your EIS user interface. For example, an EIS can use Microsoft Project to quickly determine if resources have been over-allocated or if a project is in danger of not meeting a deadline.

Since Microsoft Project supports OLE, you can link views of project data to other applications in your EIS (such as Microsoft Word) so that when project information is updated, graphs, charts, and tables in other applications will automatically reflect these updates.

To illustrate this, consider the following report created with Microsoft Word that displays a project's resource graph created with Microsoft Project. The graph is a linked Microsoft Project object and summarizes a project's progress to date and shows resource allocation. Whenever data associated with the graph is changed in Microsoft Project, the graph is dynamically updated. By clicking the More Resource Info button, the user can display other views of the project's resource data (such as a table).



The Microsoft Project Macro Facility

A macro facility is supplied with Microsoft Project that enables you to automate various tasks within the application. The facility supports simple constructs such as scripting, looping, and branching.

Although you can use the Microsoft Project macro language, Microsoft Project can also be controlled using an external macro language such as WordBasic or the macro language of Microsoft Excel. By writing macros in other applications that contain dynamic data exchange (DDE) functions, you can issue remote execution commands to Microsoft Project and exchange information with it. For example, you can:

- Use the commands available in external macro languages to create dialog boxes and controls that provide a custom interface for accessing Microsoft Project from another application such as Microsoft Excel.
- Exchange information with Microsoft Project by running a macro, instead of using OLE to link the two documents. This provides a faster exchange of information when you need to move a lot of data between applications, and you want more than a picture of a Microsoft Project view.

- Access categories of information that are not available by linking to them with OLE. For example, with DDE functions, you can directly access detailed resource assignment fields as well status fields.
- Calculate estimated task durations using Microsoft Excel, and then use a Microsoft Excel macro to send these durations to Microsoft Project to link the tasks and send back the revised project finish date.

For more information on OLE and DDE, see Chapters 7 and 8.

Using Microsoft Word in Your EIS User Interface

Microsoft Word is a powerful word-processing application designed for creating a wide variety of documents. Word can play an important role in an EIS to generate reports, create letters, and process virtually any kind of document.

Typically, Word is used as a tool that is activated by the primary user interface of your EIS when a report or other document must be created. As with other Microsoft applications, Word can be integrated seamlessly into your EIS user interface because it supports both DDE and OLE.

The powerful WordBasic macro language allows you to automate most document generation tasks, reducing or eliminating the need to know how to use a word-processing program.

Word also includes utilities such as Microsoft Draw for illustrations and Microsoft Graph, which can generate charts from tables of numbers. You can, however, link or embed objects from other applications.

When Should You Use Word?

Although Word has powerful document-handling capabilities that enable EIS users to generate reports and letters, Microsoft Excel can generate reports as well. However, the capabilities of Word and Microsoft Excel differ considerably in the area of report generation.

You can use Microsoft Excel to generate reports that display mostly financial information, such as charts and tables of numbers. Text annotation can easily be added to Microsoft Excel reports, but Microsoft Excel is not designed to handle large amounts of text or perform document formatting functions such as pagination or word-wrapping.

You can use Word when you need to take advantage of its powerful formatting and document-handling features. For example, you can use Word in an EIS to automatically generate letters to selected managers. You can build a custom Word-based user interface with buttons or menu choices that allow users to customize standard form letters, pick which type of letter to send, perform mail merge functions using a list of managers, and even print mailing labels for the letters.

Interactive Reports

Another use for Word is to create "interactive" reports with buttons that provide more information when clicked. This was illustrated in the project resource update report shown earlier in this chapter. Choosing the More Resource Info button on the report displays a detailed table of all resource data. The button is actually linked to a WordBasic macro that takes a table of data from Microsoft Project and displays it in a new document window in Microsoft Word.

You can add more buttons to perform additional tasks such as sending electronic mail to another EIS user or group of users.

Using the Customization Capabilities of Word

Building applications with Word involves a similar process as used for Microsoft Excel. You can easily customize the menus and toolbar of Word, and add buttons anywhere in a document that call macros when chosen.

When you build a user interface with Word, you begin by creating a special document template that will become your interface. Word can be customized to a large extent, and users might not be able to recognize that they are actually dealing with a Word document.

Word allows you to change many aspects of your template. You can change or delete any menu item, keystroke accelerator, or toolbar button, and you can add new items, buttons, and shortcuts. For example, if you don't want the default toolbar to be accessible from your interface, you can remove it from the View menu so it can't be displayed.

In addition to the buttons on the toolbar, you can add macro or "go to" buttons anywhere in your document by inserting a special field. Goto buttons jump to a specified point in the document when clicked. Macro buttons perform the same functions that toolbar buttons and menu items do; they call a command or macro. You can place these buttons next to the text or tables to which they relate. For example, a "Graph View" button could be placed next to a graph to allow the user to decide how the information in the graph is presented or a "Send Report" button could send reports to certain staff members.

Using OLE with Microsoft Word

Since Microsoft Word supports OLE, you can embed or link pictures, graphs, tables, and other objects from a wide variety of applications into Microsoft Word.

In the Project Resourse Update report shown earlier in this chapter, both the graph and the table are linked Microsoft Excel objects. Either object could be edited from Microsoft Excel, and the changes would be reflected in Word, or the links could be edited from Word, which would start Microsoft Excel automatically.

The WordBasic Macro Language

Microsoft Word supports a Basic-based macro language named "WordBasic." You can create WordBasic macros by either programming the statements of the macro yourself, or using the macro recorder, which automatically records your keystrokes and converts them into a complete WordBasic macro. The macro recorder requires no programming knowledge.

The most frequent uses of macros in a Word user interface will be for manipulation of documents created by the EIS (formatting, printing, generating form letters, and so on) and for communication with other applications in the EIS.

Every menu feature has an associated command that can also be called from WordBasic, and macros can call on other macros as well. Thus, anything that you can normally do from Word, and more, can be done using a macro.

Most of the macros used in an EIS will be attached to macro buttons, toolbar buttons, or custom menu items. You can also use macros that run automatically every time a document is opened or closed.

CHAPTER 7

The Data Access Layer

Frequently, important corporate data resides in different locations and is stored in multiple application-specific file formats. For example, a company might keep its budget data in a Microsoft Excel spreadsheet, its sales figures in a DB2 database on an IBM® mainframe, its human resources data in a SQL Server database, its finance figures in an Rdb database on a DEC® VAX®, and its manufacturing schedules in a Microsoft Project file.

Data managed by different applications in different locations makes it hard for users to visualize information, spot trends, or see key relationships. A major goal of an EIS application is therefore to allow users to view or manage data from a variety of sources, regardless of where it is located or how it is stored.

This chapter provides an overview of techniques for accessing data from different sources and funneling it into an EIS application where it can be manipulated.

Key Data Access Technologies and Applications

Accessing and manipulating multisource data is facilitated by the following three key technologies that should be understood by anyone designing an EIS system. These are:

- Dynamic data exchange (DDE)
- Object linking and embedding (OLE)
- Open database connectivity (ODBC)

These technologies are the core of the data access layer introduced in Chapter 3, and play a central role in many EIS applications. Additionally, two applications (Microsoft Access and the Q+E® facility included with Microsoft Excel) offer an effective means of accessing data in an EIS.

This chapter offers a closer look at these technologies and applications without covering the details of how to implement them. In the next chapter, you'll see actual examples of how they are used in practice.

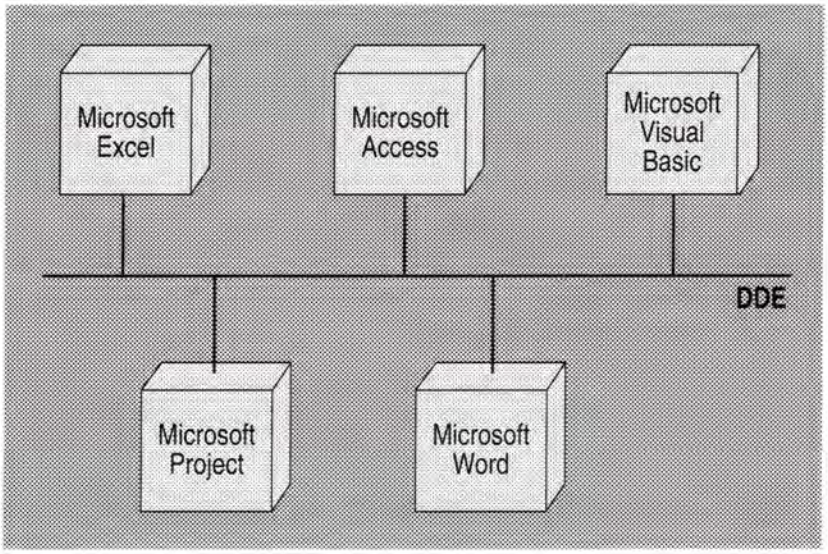
Dynamic Data Exchange

In a manner similar to networking personal computers together, applications themselves can be connected to create a more powerful and cooperative application environment. Establishing such interapplication connections removes many of the boundaries between programs, and relieves users from the burden of handling data manually.

One technology that enables data exchange between applications in the Windows environment is dynamic data exchange (DDE).

DDE acts like a communications protocol through which applications can send each other information in a standard and well-defined manner. Application developers will notice that DDE's programming paradigm is similar to that used in building network solutions, but is much simpler and can be used within macros.

Virtually all Microsoft applications, including Microsoft Excel, Microsoft Access, Microsoft Project, Microsoft Word, and Microsoft Visual Basic support DDE, and many other applications from a broad range of vendors support DDE as well. This means you can tie Microsoft applications together into a single cooperating unit, or connect Microsoft applications to other applications. In a synergistic effect, the sum of these applications working together in an EIS can be much greater than all the applications working individually.



A good illustration of DDE is connecting spreadsheets and databases. Suppose that a company is using a Microsoft SQL Server database to store its sales data, and that several department managers have requested access to this data for forecasting, goal seeking, and charting.

Since Microsoft Excel provides financial analysis functions and sophisticated charting capabilities, it makes sense to build a system that allows these managers to transfer the sales data from the database to Microsoft Excel.

How Does DDE Work?

Windows is a message-based environment. This means that many of the actions performed within the environment are achieved by sending messages back and forth. For example, messages are used to handle mouse events, manipulate dialog boxes, and paint graphics on the screen. A small subset of the Windows message set is specific to DDE, and is used by applications to communicate with each other. DDE is therefore built directly into the Windows environment.

Any application that initiates a DDE transaction is called a *client*, while the application that responds to the request is known as a *server*. The session during which data and messages are exchanged is referred to as a *conversation*.

To start a conversation, the client application sends an INITIATE message to the server. In most cases, the server responds with an ACK (acknowledge) message to indicate that it is ready to communicate. This initial exchange opens a channel of communication, after which a variety of messages can be sent back and forth. There are a total of nine DDE message types, and these are described in the following table.

Message	Description
ACK	Either client or server acknowledges the receipt of a DDE message.
ADVISE	A client tells a server to advise it of any updates to a particular data object held by that server.
DATA	A client is notified of the availability of data in shared memory.
EXECUTE	A client asks a server to execute one or more of its commands.
INITIATE	Sent from a client to a server to initiate a conversation.
POKE	A client indicates to a server that it wants to send unsolicited data to that server.
REQUEST	Sent from a client to a server asking the server to provide data.
TERMINATE	Either a client or a server is asking for the DDE conversation to be stopped.
UNADVISE	A client tells a server that it no longer needs updates regarding the specified data item.

Using these messages, a variety of useful functions can be implemented within your EIS application, each of which establishes a different type of relationship—a cold link, a hot link, or a warm link—between two applications.

Cold Link

In a cold link, two applications establish communications for the purpose of exchanging a single, predefined set of data such as a graphic image or a range of cells in a spreadsheet. Once the data has been transferred, the link is closed (terminated).

Hot Link

A hot link is a permanent data link that enables a client to establish a continuous real-time conversation with a server. This allows a high level of transparent integration between two applications. When a specified set of server data is updated, the data is automatically sent to the client. For example, as data is updated in a database, the corresponding numbers in a spreadsheet table are dynamically modified.

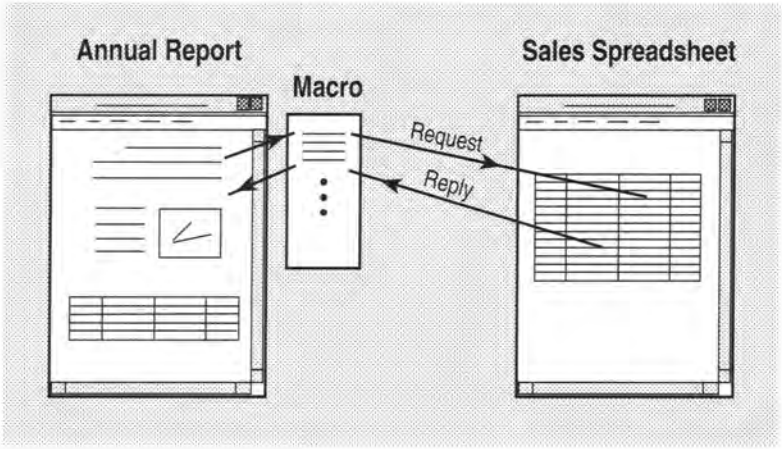
Hot links enable information to become "common" across applications, relieving users and adminstrators from the burden of updating every instance of that information in multiple applications.

Warm Link

A warm link is similar to a hot link, but the client is only advised of any changes to a server's data set. The data is not sent automatically to the client, but is sent if explicitly requested by the client. This allows applications to select which information they want to import.

Remote Command Execution

Frequently, there is a need to have one application send a command to another application. DDE supports this "remote command execution," enabling applications to control each other's actions. For example, in the following illustration, a Microsoft Word macro sends commands to Microsoft Excel to recalculate and sort a table of sales figures. This functionality allows developers to automate many functions that would otherwise be performed manually by users.



Different Levels of Implementation

DDE is not something that can be controlled directly by users. As part of the data access layer, DDE is strictly a programming interface that can be used by all levels of developers. DDE is supported by the macro languages of most Microsoft applications, so you can take advantage of it without being a professional programmer. It is also supported by most other development environments including Microsoft Visual Basic and Microsoft Access.

Note Applications support DDE in different ways. Some of the capabilities previously described may not be available in all applications. Consult your application's product documentation for more information.

Object Linking and Embedding

Object linking and embedding (OLE) is a set of services that provides a powerful means to create documents and views of data that consist of multiple sources of information from different applications. Objects can be almost any type of information, including text, bitmap images, line art, and even voice annotation.

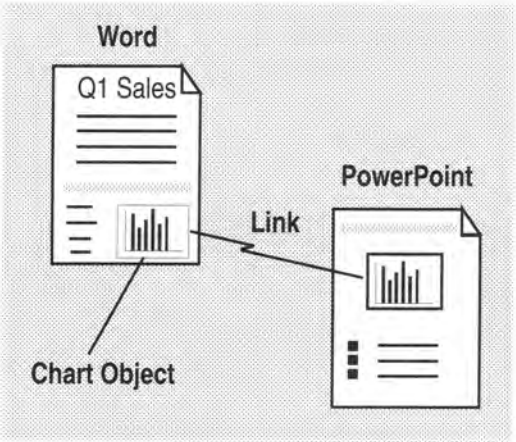
Like DDE, OLE is built into the Windows environment and is supported by most Microsoft applications as well as by hundreds of other applications from different vendors. Also like DDE, OLE allows applications to cooperate in a seamless fashion, tying together the unique features of different applications into a more powerful unit.

OLE Explained

OLE associates two major types of data with an object: presentation data and native data. An object's presentation data is information that is needed only to display the object on the screen, while its native data is all the information needed for an application to edit the object.

In the OLE model, a user can either link or embed an object into their document. Linking is a process whereby only an object's presentation data and a reference (or pointer) to its native data are placed in a document. The actual native data associated with the object exists in some other location such as in a file on disk. Whenever the object is updated by an application, it appears updated within the document. To the user, a linked object acts as if it were wholly contained within the document.

The following illustration shows a link between an object in a Microsoft Word document and a Microsoft PowerPoint presentation. Whenever the object is updated in Word (for example, using Microsoft Graph), the object is automatically updated in the PowerPoint presentation.



In contrast, embedding physically places a copy of an object's presentation data and its native data within a document. All information necessary to edit the object is therefore contained in the document.

Although embedding involves more overhead (it makes the document larger), it allows the object to be transferred along with the document to another computer, and to be edited on that computer. In contrast, transferring a document containing a linked object to another computer will not result in transferring the object's native data to the destination computer. As a result, the object cannot be edited on the destination computer.

However, when used on a single computer, linked objects are more efficient than embedded objects because a single instance of the object's data can serve many different documents.

Using OLE

A major strength of OLE is its ease of use. Using a simple "copy and paste" (or "copy and paste link") operation, as shown in the following procedure, you can embed or link an object in just a few seconds. In OLE terminology, the *source* refers to the document or application from which you copy the object; the *destination* refers to where you paste (or paste link) the object.

▶ **To embed an object in a document**

1. Start the application that was used to create the object that you want to embed.
2. Select the object using the application's method for object selection.
3. From the application's Edit menu, select Copy.
4. Open the document in which you want to embed the object by switching to the application that created the document.
5. From the Edit menu of the application in which you are embedding the object, select Paste.

A copy of the object will now be embedded within the document. Double-clicking the object in the document will start the application that created the object, and you can make any edits you want to that copy of the object. However, these changes will not be reflected in the original object because you are editing a copy of the original. Conversely, any changes made to the original object won't be reflected in the embedded copy of the object because they are different objects.

Linking an object involves a similar process to embedding, but with some slight differences.

▶ **To link an object**

1. Start the application that was used to create the object that you want to link.
2. Select the object using the application's method for object selection.
3. From the application's Edit menu, select Copy.
4. Open the document in which you want to link the object by switching to the application that created the document.
5. From the Edit menu of the application in which you are linking the object, select Paste Special. This will display a dialog box similar to the following illustration.

6. Select the data type of the object you want to link. These will vary among applications. Those supporting embedding provide a data type with the word "object" in it, such as in the preceding dialog box. Select a data type without the word "object" in it, and then click the Paste Link button.

The object should now be linked, and a picture of the object will appear in the destination application. Double-clicking the object within the destination document that received the object will open an editing window for the original object. Any changes made to the object will be reflected in the original object, as well as in the linked object.

Note Some applications may use a different series of steps than the preceding steps, or may not support linked or embedded objects at all. Consult your application's documentation for details on using OLE within that application.

The Significance of OLE for EIS

Because of its power and simplicity, OLE can be used to solve a wide range of problems. For EIS developers, OLE provides an easy means of linking data from multiple applications without having to write a single line of code. You can create screens consisting of objects from multiple source applications, including text, line art, bitmap images, tables, and other objects. Users benefit because they can view data from a wide variety of sources on the screen at one time, showing important relationships among diverse groups of data.

Another benefit of OLE is that it allows you to edit objects without manually starting the application that was used to create the object, and without knowing which file contains the object. For example, if an EIS user double-clicks on a Microsoft Project Gantt chart within a Microsoft Excel worksheet, Microsoft Project will automatically be started and the file containing the object will be open. The user does not need to find the Microsoft Project icon on the desktop or know the name of the file containing the Gantt chart. The user must, however, know how to operate Microsoft Project in order to edit the object.

In the next chapter, a few strategies for implementing OLE within an EIS will be discussed.

DDE versus OLE

DDE and OLE are very similar because they both facilitate the exchange of data between applications. But each has its unique advantages. OLE is good for updating object-level data such as bitmaps, graphs, and tables. When you use OLE to incorporate a linked object (using a "copy and paste link" procedure), the image of the object is automatically redrawn when it is changed within its source application. There is nothing to program because everything happens automatically. OLE can therefore be set up and maintained using a few simple actions within applications.

Because OLE works by itself, most macro languages do not support OLE, so it is not something that you can control programmatically within your EIS using macros. However, if you need greater control over OLE actions, development environments such as Visual Basic allow you to control linked and embedded objects using specialized APIs.

Although requiring more work to implement than OLE, DDE offers a higher degree of programmatic control over data. Many macro languages support DDE, and you can use it to manage links, send and receive data between applications, and start commands in other applications. For example, you can use DDE from a Microsoft Excel macro to start Word, open a report file, format it, and print it. OLE does not provide this capability.

Generally, DDE is more flexible than OLE at handling data, such as a number or character string, at the value level. For example, DDE allows you to determine whether a link should be updated automatically whenever the source data changes, or whether the link should be updated only when the client application requests that an update should occur.

Also be aware that DDE can be slow at handling complex, object-level data. OLE is much more efficient for this purpose. In an EIS, using both DDE and OLE for data exchange between applications can result in a good balance of features and performance.

If you are just starting to build an EIS and are new to the Windows environment, it may be easier to start with OLE because it doesn't require any programming. As you make progress in building your EIS, you can then implement DDE to provide additional features.

ODBC

Open database connectivity is an application programming interface that allows developers to access information in a wide range of databases using a single, consistent set of function calls. By providing a single API for such a wide range of databases, developers can design an application to concurrently access, view, and modify data from multiple data sources.

ODBC is a relatively new specification that is being implemented by many different database and application vendors. Currently, ODBC is supported within Microsoft Visual Basic, Microsoft Access, and many more Microsoft products will soon support it. For a more detailed discussion of ODBC and a list of vendors who have announced their intent to support it, see Chapter 12. Also see Chapter 8 for a brief discussion of support for ODBC within Microsoft Visual Basic.

Q+E

Since most corporate data is stored in databases on remote servers, EIS applications must have a means to access this data and transfer it to the user's local personal computer where it can be displayed or manipulated. Since Microsoft Excel handles tabular data well, it makes sense to transfer database tables into Microsoft Excel, where they can be charted, analyzed, reported, and manipulated in hundreds of ways.

One tool that facilitates the integration of Microsoft Excel with a wide range of databases is a facility known as "Q+E," which provides the ability to:

- Interactively query a variety of database systems, including dBASE, SQL Server, and Rdb.
- Establish DDE links between Microsoft Excel and several different database systems.

Although the first of the preceding features is useful, the second feature is particularly interesting for EIS developers because you can use it to automate queries and make data access easier.

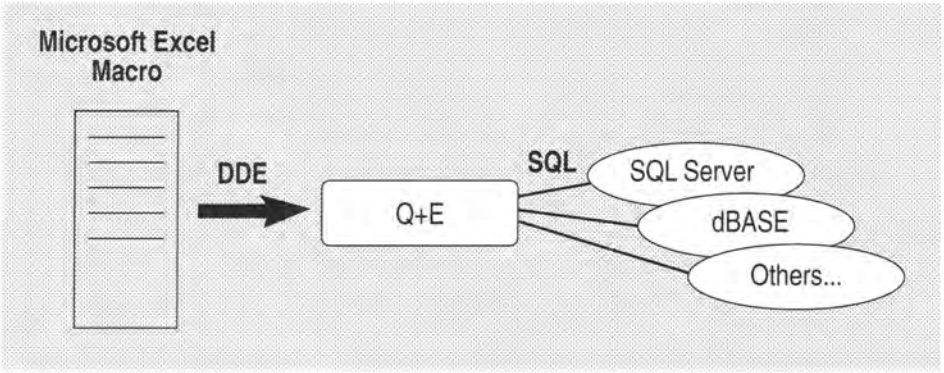
Note Q+E is available in the English, French, and German versions of Microsoft Excel, plus many other versions. For more information on Q+E for your version of Microsoft Excel, check your documentation or contact your local Microsoft subsidiary.

How Does Q+E Work?

Not all database systems support DDE. Instead, most databases have their own method for establishing communication with other applications. Q+E is important because through DDE it gives applications a uniform means of accessing data stored in different databases.

To use Q+E, you write a Microsoft Excel macro that sends DDE commands to Q+E. Typically, these DDE commands contain embedded SQL queries, which ask for specific information from the database. The commands also identify where to display the data that is received back from the database (usually a range of cells inside a Microsoft Excel spreadsheet).

Q+E then "translates" the DDE commands into a communication session with the database, using the database's own communications method. To the Microsoft Excel user, it appears as if Microsoft Excel is talking directly to the database over a DDE link. Q+E simply runs in the "background" and the user does not need to know it is there.



Microsoft Access

Chapter 6 provides a few examples of how Microsoft Access could be used to build an EIS user interface. In addition to its strong front-end development capabilities, Microsoft Access (as its name suggests) can also be used to access data from a number of different data providers. It therefore contributes to both the user interface and data access layers of your EIS.

When Should You Use Microsoft Access?

In most cases, you will install your EIS solution in an enterprise environment that already has one or more database systems such as SQL Server on a personal computer, DB2 on an IBM mainframe, or Rdb on a DEC VAX. In such environments, you must decide how to build a front end to all these databases within your EIS, and how to give your EIS users access to data in these databases while shielding them from the complexity of databases and networks.

In the previous section, Q+E was described as a technology that allows you to connect to multiple databases and bring data into Microsoft Excel. For many EIS systems, Microsoft Excel and Q+E provide enough data management and analysis features to serve as the primary front end to your corporate databases.

You can accomplish similar connectivity to database systems using Microsoft Access. Although Microsoft Access does not have the financial analysis and charting features of Microsoft Excel, it provides some other possibilities:

- Since Microsoft Access provides its own relational database management system, you can use Microsoft Access on a server to *stage* data that can be used in your EIS. This means that selected data from other databases can be pre-processed into database tables that are more efficient for your EIS to use. Microsoft Access can connect to these databases over networks such as Microsoft LAN Manager, Novell® NetWare®, Lantastic®, and Banyan® VINES®. For more information about staging data, see Chapter 10.
- Microsoft Access includes open database connectivity (ODBC) drivers that allow your EIS access to a wide range of corporate databases. Using ODBC, for example, you can use the Microsoft Access Basic-based programming language to connect to several different databases and create a report that contains data from multiple sources.

In addition to its database connectivity features, Microsoft Access also supports DDE and OLE, which allows Microsoft Access database applications to send and receive data with other applications on the desktop.

CHAPTER 8

Implementing the Data Access Layer

When implementing the data access portion of your EIS, two primary goals take precedence:

- An EIS must allow its users to access and view the right data quickly, without having to perform a long series of steps.
- An EIS must hide the underlying complexity of how the data is located, accessed, and processed. In most cases, users don't care where data is located or whether it is stored in a database, document, or spreadsheet file. They simply want a better means to visualize and manipulate data.

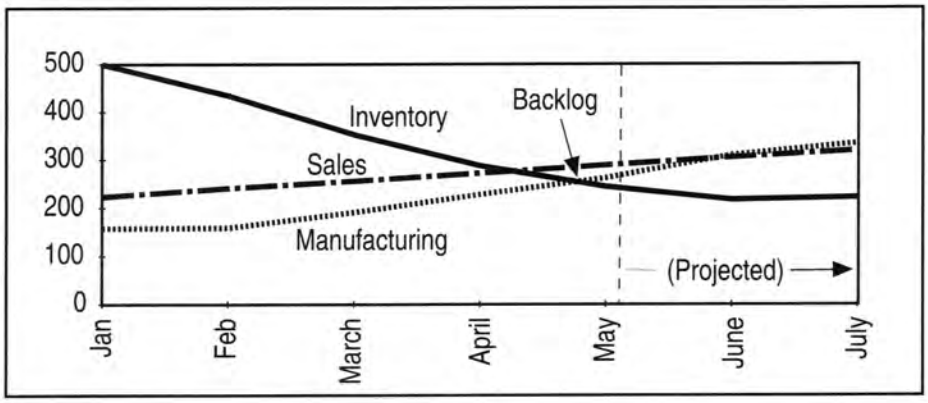
Although there are other secondary goals, focusing on these two will ensure that your EIS development effort is off to a good start. The remainder of this chapter shows you how to use a few technologies and programs—DDE, OLE, and Q+E—that will help you achieve these goals.

Implementing OLE in Your EIS Application

As you have briefly seen in previous chapters, OLE is both powerful and simple to implement. In this section, you'll be taken through a more detailed example of using OLE within an EIS.

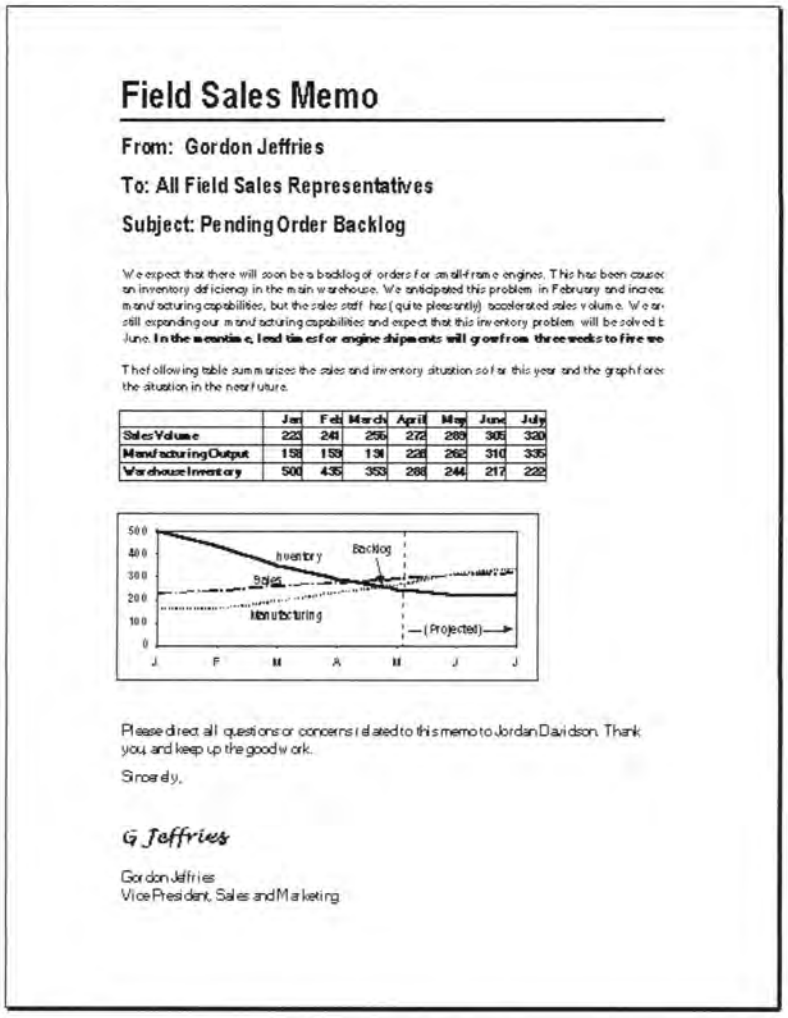
Suppose we have the following Microsoft Excel sales worksheet and its associated chart.

	Jan	Feb	March	April	May	June	July
Sales Volume	223	241	256	272	289	305	320
Manufacturing Output	158	159	191	228	262	310	335
Warehouse Inventory	500	435	353	288	244	217	222



In OLE terminology, the top illustration is called a *table object* and the bottom illustration is called a *chart object*. Recall that an object is simply any element of information that can be manipulated by applications as a single unit.

A typical EIS application will allow users to display this information within Microsoft Excel, but also to manipulate these objects for other purposes. For example, suppose that a vice president wants to communicate the pending order backlog shown in the chart to people in the sales organization. To make this easy, the EIS could allow the vice president to create the following memo just by choosing a button and filling in some text.



Field Sales Memo

From: Gordon Jeffries

To: All Field Sales Representatives

Subject: Pending Order Backlog

We expect that there will soon be a backlog of orders for small-frame engines. This has been caused by inventory deficiency in the main warehouse. We anticipated this problem in February and increased manufacturing capabilities, but the sales staff has (quite pleasantly) accelerated sales volume. We are still expanding our manufacturing capabilities and expect that this inventory problem will be solved by June. **In the meantime, lead times for engine shipments will grow from three weeks to five we**

The following table summarizes the sales and inventory situation so far this year and the graph forecasts the situation in the near future.

	Jan	Feb	March	April	May	June	July
Sales Volume	223	241	256	272	289	305	320
Manufacturing Output	158	159	191	228	262	310	335
Warehouse Inventory	500	435	353	288	244	217	222

Please direct all questions or concerns related to this memo to Jordan Davidson. Thank you, and keep up the good work.

Sincerely,

G Jeffries

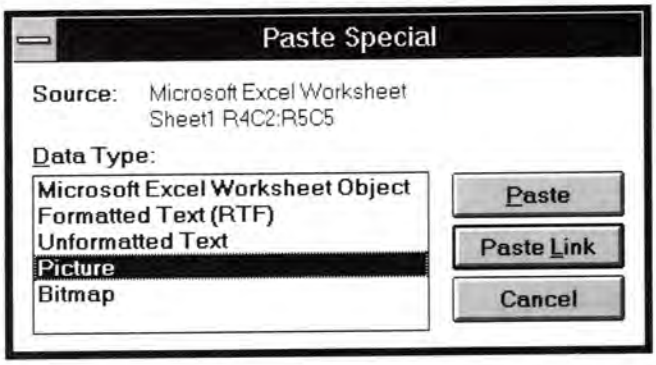
Gordon Jeffries
Vice President, Sales and Marketing

To implement this capability within the EIS, you would first create a standard memo template using Microsoft Word and place a letterhead at the top. The following steps show you how to link the table object within the memo.

▶ To copy the Microsoft Excel table and link it in the memo

1. Highlight all the cells in the table in Microsoft Excel to select the table.
2. From the Microsoft Excel Edit menu, select Copy.
3. Switch to Microsoft Word.
4. From the Microsoft Word Edit menu, select Paste Special.

The following dialog box appears.



5. In the Data Type box, select Picture.
6. Choose the Paste Link button.

A picture of the table appears in the memo, and the object is linked to the original Microsoft Excel table. Double-clicking the object in the document will start Microsoft Excel, and you can make any edits you want to the original object. All changes will be reflected in the linked table within the memo.

Use the same steps to link the chart object within the memo.

The signature at the bottom of the memo was embedded in the memo rather than linked. There is no need to link the signature because it is static and won't change.

▶ **To embed a signature in the memo**

1. Using the Windows Paintbrush™ accessory, sketch a signature using the Brush tool in the tool palette (see the Paintbrush documentation for more information).
2. Using the Paintbrush Pick tool, select the signature.
3. From the Paintbrush Edit menu, select Copy.
4. Switch to Microsoft Word.
5. From the Microsoft Word Edit menu, select Paste. The signature appears in the memo.

There are other ways to include bitmap images in Microsoft Word documents. For example, you could use a scanner to scan a hard-copy signature, and then insert the scanned image file by choosing the Picture item on the Microsoft Word Insert menu. For more information on inserting images, see the *Microsoft Word User's Guide*.

When all of the information is linked or embedded into the memo, you can create the Microsoft Excel macro that opens the memo from within the EIS. To do this, you can create a "Create Memo" button in Microsoft Excel that is attached to a macro that starts Microsoft Word, opens the memo template, and places the text insertion cursor at the point where typing should begin. Although not done in the preceding example, you could provide additional buttons in either Microsoft Excel or Microsoft Word that automatically fill in different "boilerplate" text options and require no modification.

Implementing DDE in Your EIS Application

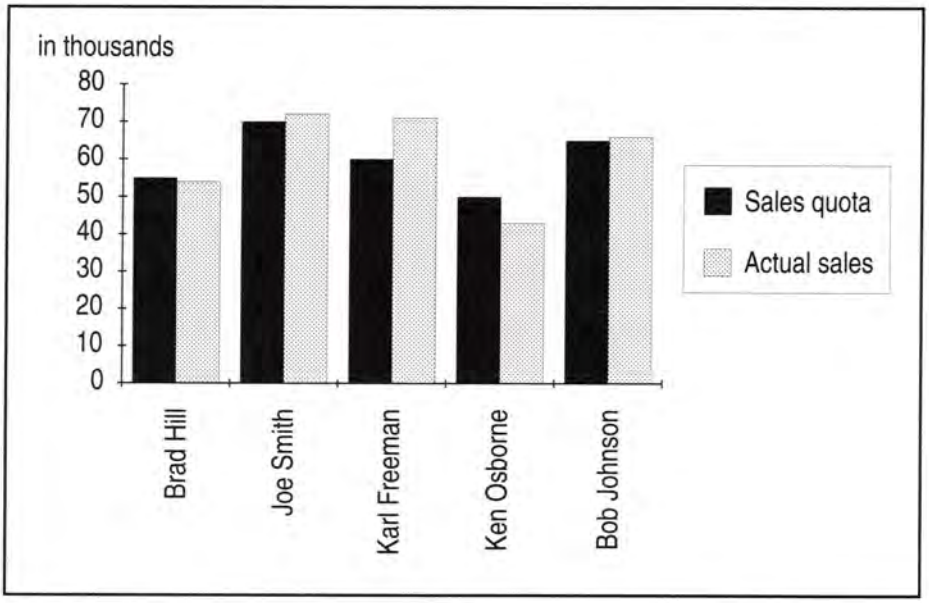
OLE technology is ideal for creating links between application objects. DDE provides a similar kind of automatic linking, but allows you to achieve a finer degree of control over how the data is updated and displayed. This section provides a few examples of how DDE can be used to integrate various applications in the Windows environment.

Integrating Microsoft Excel and Microsoft Word with DDE

In the previous section, an example was given on using OLE to integrate Microsoft Excel and Microsoft Word. To help compare OLE with DDE, the following example of integrating Microsoft Excel and Microsoft Word is provided, this time using DDE.

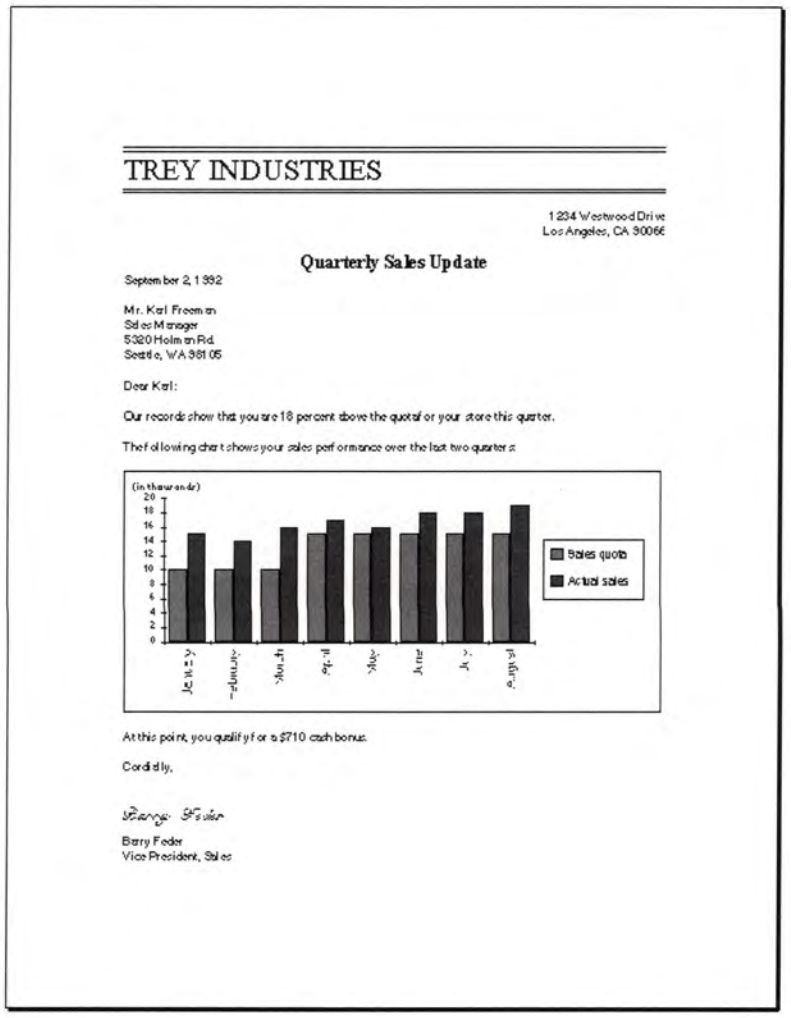
Suppose we have the following Microsoft Excel table and chart, showing the performance of five store managers in the Western region.

Employee ID	110	120	130	140	150
Manager name	Brad Hill	Joe Smith	Karl Freeman	Ken Osborne	Bob Johnson
Store	Aurora	Redmond	Greenwood	Ballard	Highlands
Sales quota	55	70	60	50	65
Actual sales	54	72	71	43	66
Percent of quota	-2%	3%	18%	-14%	2%



In our example EIS, the sales figures (both quota and actual) in the Microsoft Excel table are updated on a regular basis, which automatically updates the chart.

Among the features of this EIS is the capability to automatically generate sales reports to the store managers as shown in the following illustration.



TREY INDUSTRIES

1234 Westwood Drive
Los Angeles, CA 90066

Quarterly Sales Update

September 2, 1992

Mr. Karl Freeman
Sales Manager
5320 Holman Rd.
Seattle, WA 98105

Dear Karl:

Our records show that you are 18 percent above the quota for your store this quarter.

The following chart shows your sales performance over the last two quarters:

At this point, you qualify for a $710 cash bonus.

Cordially,

Barry Feder
Vice President, Sales

To generate the letter, the following Microsoft Excel macro was written:

Names	Formulas	Comments
	CreateLetters	
DialogOK	=DIALOG.BOX(A37:G45)	Display dialog
	=IF(DialogOK,"then continue, else", GOTO(End))	Exit macro if Cancel
ID	=(MATCH(FirstName&" "&LastName, 'C:\EIS\SALES.XLS'!B2:K2, 0) + 10)*10	Get employee ID#
SalesQuota	=INDEX('C:\EIS\SALES.XLS'!B4:K4, ID)	Get quota
ActualSales	=INDEX('C:\EIS\SALES.XLS'!B5:K5, ID)	Get actual sales
Percent	=INT((ActualSales/SalesQuota)*100) - 100	Figure % of quota
	=OPEN("Emp"&ID&".xls",,TRUE)	Open employee chart
	=SELECT.SPECIAL(13)	Select chart
	=COPY.PICTURE()	Copy chart
	=CLOSE()	Close file
DocChan	=INITIATE("Winword","\EIS\LETTER.DOC")	Start Word; open file
	=APP.ACTIVATE("Microsoft Word - LETTER.DOC",TRUE)	
	=EXECUTE(DocChan, "%tag"&LastName&FirstName&"~~")	Insert address
	=EXECUTE(DocChan,"%eg"&"PercentageAboveBelow~")	Move to perf. figure
AboveBelow	=IF(Percent>0, "above", "below")	Is quota < or > 10%?
	=EXECUTE(DocChan, ABS(Percent)&" percent "&AboveBelow)	Write perf. figure
	=EXECUTE(DocChan,"%eg"&"Bonus~")	Move to bonus position
BonusText	=IF(Percent>=10, GetsBonus, NoBonus)	Bonus for employee?
GetsBonus	="At this point, you qualify for a $"&(10*ActualSales)&" bonus."	
NoBonus	="You do not qualify for a bonus at this time."	
	=EXECUTE(DocChan, BonusText)	Enter bonus text
	=EXECUTE(DocChan,"%eg"&"Chart~")	Move to chart area
	=EXECUTE(DocChan,"%ep")	Embed chart in letter
	=APP.ACTIVATE("Microsoft Excel - SALES.XLS", FALSE)	Return to Excel
	=TERMINATE(SysChan)	Close system channel
	=TERMINATE(DocChan)	Close doc channel
End	=RETURN()	

Note This macro was written using the English language version of Microsoft Excel. Program statements may differ in certain localized versions of the software. If you are using a different version of Microsoft Excel, you can use this macro as a general model for how the program could work in localized versions.

The preceding macro is a relatively simple example of how to use DDE functions to control another application. It is called when the EIS user clicks a Create Letter button on the Microsoft Excel worksheet containing the sales table. The macro first brings up a dialog box that asks the user to enter the name of the employee for whom the letter will be created. It then looks up pertinent sales information about the employee on the spreadsheet. Next, it opens a file containing more detailed information on the selected salesperson and copies the chart found there onto the clipboard for later embedding into the letter.

When the macro is finished with the Microsoft Excel files, it opens a DDE channel to Word using the INITIATE function, and begins manipulating the Word letter with the EXECUTE function. EXECUTE provides an easy way to manipulate another application by sending keystrokes such as the "%tag" sequence, which calls the Format / InsertAddress command, using the letter's Glossary to find an address.

The Microsoft Word letter is preformatted with bookmark fields in the places where sales information will be added. Since the letter is based on the Microsoft Word standard blockletter template, the Microsoft Excel macro uses the Microsoft Word InsertAddress function included with the template to automatically address the letter to the appropriate employee. It then adds information about the salesperson's current situation and attaches a graph of the salesperson's progress during the last two quarters. When this is finished, the EIS user can edit the letter further to add a personal touch, and then print it out or send it through electronic mail.

Integrating Microsoft Excel and SQL Server with DDE

Microsoft Excel is excellent for quick and easy manipulation of character and numeric arrays in a "flat-file" (nonrelational) format. On the other hand, databases satisfy the need for relational operations among larger sets of data.

These two applications are therefore complementary, with the weakness of one being the strength of the other. Because of this, it makes sense to allow these applications to take advantage of the unique features of the other, allowing database information to be "downloaded" into the rows and columns of a spreadsheet. This enables quick "what-if" analyses on database data without the need to invest time and money in custom database applications.

To achieve this merging of database and spreadsheet, some versions of Microsoft Excel come with a special DDE capability that allows it to connect with a wide variety of databases over a network. This capability is offered through the Q+E facility, which was introduced in Chapter 7. With Q+E, you can write Microsoft Excel macros that perform data analysis and produce reports using database data that is either local or on a network server.

Q+E accepts DDE messages from Microsoft Excel macros, reformats them as SQL queries, and sends them to a database engine for processing. When interacting with Microsoft Excel macros, Q+E acts as a "backstage" DDE communications facilitator that users don't see.

Virtually all functions that can normally be performed on a SQL Server database (including record insertion, data rollback, and other standard features) can be accomplished through Microsoft Excel macros, allowing the quick construction of complete database user interfaces.

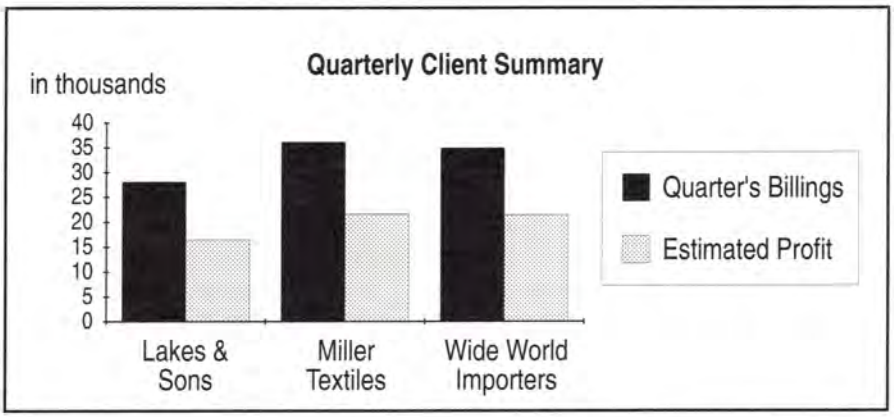
The following code sample is a simple Microsoft Excel macro that establishes a DDE link with Q+E to query a Microsoft SQL Server database. The macro queries the Clients database for billing and expense information on jobs completed for three clients. It then creates a chart showing the profit earned from each client during the last quarter.

Names	Formulas	Comments
	ClientSummary	
Client1	="Best Hardware"	Client IDs
Client2	="Glassware Inc."	
Client3	="Spectrum Novelties"	
	=FORMULA("Client ID", QBSumm.XLS!$A$1)	Set up table headings
	=FORMULA("Quarter's Billings", QBSumm.XLS!$B$1)	
	=FORMULA("Estimated Profit",QBSumm.XLS!$C$1)	
	=FOR("I",1,3)	Loop once for each client
ClientID	=INDEX(B4:B6, I)	
chan	=INITIATE("QE", "SELECT HOURLY_RATE, HOURS_BILLED, EXPENSES FROM SQLServer I Clients WHERE CLIENT_ID='" & ClientID & "'")	Initiate a DDE channel with Q+E. The select statement defines the query.
Billings	=SUM(REQUEST(chan, "R1C1:R4C1") * REQUEST(chan, "R1C2:R4C2"))	Billings = Sum (Rate * Hours_Billed)
Profit	=Billings - SUM(REQUEST(chan, "R1C3:R4C3"))	Profit = Billings - Expenses
	=FORMULA(ClientID, OFFSET(QBSumm.XLS!$A$1,I,0))	Fill in the appropriate values in the spreadsheet table
	=FORMULA(Billings, OFFSET(QBSumm.XLS!$B$1,I,0))	
	=FORMULA(Profit, OFFSET(QBSumm.XLS!$C$1,I,0))	
	=TERMINATE(chan)	
	=NEXT()	
	=FORMULA.GOTO(QBSumm.XLS!$A$1:$C$4)	
	=FORMAT.AUTO()	Format the table
	=FORMULA.GOTO(QBSumm.XLS!$A$6:$F$21)	
	=ACTIVATE("QBSUMM.XLS")	
	=COPY()	
	=CREATE.OBJECT(5,"R6C1",0,0,"R22C7",0,0,1,TRUE)	Make a chart based on the table
	=CHART.WIZARD(TRUE,"QBSUMM.XLS!R1C1:R4C3", 3,1,2,1,1,1,"Quarterly Client Summary","","","")	
	=RETURN()	

In this macro, the INITIATE command loads Q+E into memory and opens a query window in which the appropriate database information is placed. The DDE channel number is stored in the variable Chan, which is used by subsequent REQUEST commands to retrieve the HOURLY_RATE, HOURS_BILLED, and EXPENSES fields for the client named in the WHERE clause of the SELECT command.

The Clients database (which is accessed by Q+E) has fields for the CLIENT_ID, HOURLY_RATE, the MONTH in which the work was done, HOURS_BILLED that month, EXPENSES generated by the work, and extra NOTES. However, the macro only uses four of these fields, which Q+E allows you to choose using the SQL SELECT statement.

During each iteration of the FOR loop, a different query window is opened and the total quarter's billings and profit associated with the appropriate client are calculated and entered on a new spreadsheet. Once the loop is finished, the macro takes this newly created table and makes the following bar chart from it.



All the commands used in the macro are a standard part of Microsoft Excel's macro language. They are the macro user's interface to DDE and enable sophisticated application integration without the extra work of low-level programming in a language such as C.

Much more sophisticated applications can be built, and even some of these can be fairly easy to implement. For instance, several DDE channels can be opened at the same time, and information can be sent from different databases into multiple spreadsheets. This data can be processed by spreadsheet macros and written back into the database as updated records with the POKE macro command.

Integrating Microsoft Project and Word with DDE

Typically, project management packages such as Microsoft Project are used as a component of the data source layer in an EIS application. This means that Microsoft Project provides data to the EIS, but users do not interact with it directly.

The following macro demonstrates how Microsoft Project can be integrated into an EIS as a data source. In this example, Microsoft Project is controlled externally by Microsoft Word using DDE within a WordBasic macro.

The macro begins by running Microsoft Project and opening the project file PRODUCT.MPP, which contains the relevant project data. It then displays a custom resource allocation view of the project data in the file.

Next, using one of the Microsoft Project macro commands, a message box is displayed, allowing the user to return to Word and terminate the DDE link between the two applications. With a macro such as this, the EIS user can access data created by Microsoft Project without requiring any knowledge of Microsoft Project itself.

```
Sub MAIN
ProjChan = DDEInitiate("Winproj", "c:\winproj\library\product.mpp")
AppActivate "Microsoft Project - product.mpp"
DDEExecute ProjChan, "View [Resource Allocation 2]"
DDEExecute ProjChan, "EditGoto .ID=[1] .DATE=[6/17/92]"
DDEExecute ProjChan, "Message .Message=" + Chr$(34) + "Click OK when you are done
viewing" +  Chr$(34) + ".Type=0"
AppRestore
AppMaximize
DDETerminate ProjChan
End Sub
```

CHAPTER 9

Accessing Enterprise-wide Data

For smaller companies, corporate data is usually spread over a few servers or minicomputers connected to the same network. For medium and large-scale corporations, corporate data can exist on many different types of computing platforms across a range of local and wide-area networks.

Since it is impractical to centralize corporate data in a single location and under a single application, your EIS must be able to access data no matter where (or how) it is stored. Furthermore, this data access should be completely transparent to the user. People simply want information without having to worry or think about where it is stored or how it is formatted.

A major part of building a successful EIS is therefore understanding which technologies can facilitate data access on an enterprise-wide basis, and help make the mechanics of this access transparent to the user. This chapter describes a few options for accessing enterprise data.

The Spectrum of Enterprise Data Access Tools

Chapters 7 and 8 discuss various data access technologies and applications, including DDE, OLE, ODBC, Microsoft Access, and Q+E. Each of these offer unique advantages for making data available to your EIS.

In addition to these technologies and applications, a variety of third-party products can be used to connect your EIS to the corporate data they need. Enterprises typically have one or more of these products (such as communications servers and database servers) that are already installed to facilitate the exchange of data between heterogeneous corporate systems and personal computers.

Many of these third-party products can work cooperatively with Microsoft applications, and can be used to supply enterprise-wide data to EIS applications built with Microsoft products. Consult your third-party product documentation for additional information.

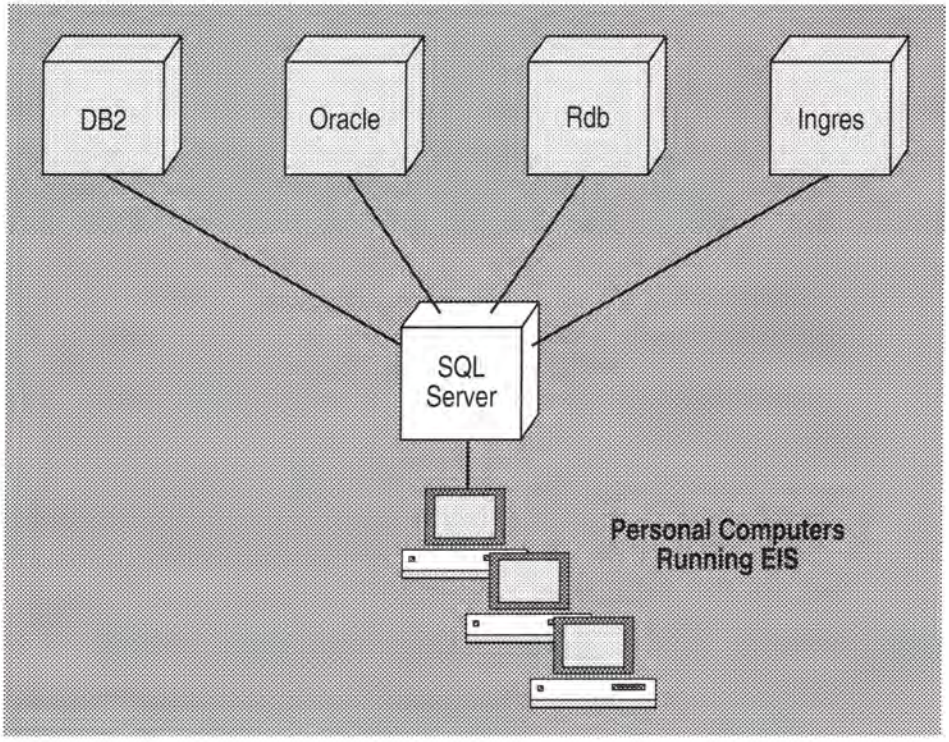
Microsoft offers several powerful products and technologies that enable personal computers to access a wide range of data sources located virtually anywhere in an enterprise. These products can supplement existing third-party data access tools, or provide a comprehensive data access solution if no enterprise data access capability has been installed.

Equally important, Microsoft's enterprise data access solutions are designed to work with Microsoft desktop applications. This enables your data access solution to be a tightly integrated component of your EIS. In the following sections, the following three Microsoft data access products will be discussed:

- Microsoft SQL Server.
- Microsoft Open Data Services.
- DCA®/Microsoft Communications Server.

Microsoft SQL Server — The Enterprise Connection

Microsoft SQL Server is a high-performance network database management system designed to help companies integrate, manage, and access data anywhere in an enterprise. In Microsoft's Open EIS approach, SQL Server can act as a central "hub" through which enterprise data is transferred to the desktop. Users can therefore work with corporate data using the same graphical applications they use to work with personal data.



Connection to the many environments shown in the preceding illustration can be through facilities provided with SQL Server, or through a wide variety of third-party gateway solutions.

What Is Client-Server Architecture?

SQL Server offers a client-server architecture that supports the most sophisticated database features while providing an enterprise-wide data connectivity strategy that delivers information directly to the desktops of users.

Client-server architecture splits an application into a front end client component (such as Microsoft Excel) and a back-end server component, such as SQL Server. The client component presents and manipulates data on the workstation, and the server stores, retrieves, and protects data. This architecture makes it possible for multiple clients—such as spreadsheets, personal computer databases, and custom applications—to access the same SQL Server database at the same time and share data.

Client applications communicate with SQL Server using Structured Query Language (SQL), which is the standard language for relational databases and is compatible with most database products. SQL Server can communicate with any remote client (such as Windows and MS-DOS® applications) that uses SQL.

What SQL Server Offers

SQL Server runs on OS/2®-based systems and can operate with LAN Manager or any other network that is 100% compatible with OS/2. Versions also exist for a variety of other platforms, such as UNIX and VMS®, which have LAN Manager network implementations available.

One of the greatest benefits of the SQL Server design is that it supports distributed data management, making truly distributed applications possible. The SQL Server distributed update capability allows databases on multiple SQL Server installations to be updated with a single transaction. This ability to scale a system in response to data server requirements provides substantial flexibility, permitting access to geographically or functionally dispersed data.

In addition, SQL Server offers the following features.

- Advanced software technology gives it reliability, performance, and data processing capabilities equal to or better than those found in production-oriented, minicomputer/mainframe database management systems.
- Transaction processing ensures that SQL Server databases are consistent and can be recovered in case of system failure—whatever the cause.
- High-performance architecture makes optimum use of current-generation hardware, maintaining high throughput levels as more users are added to the LAN.
- Advanced administration and security features ensure a secure, easy-to-manage system.
- Referential integrity maintains consistency between multiple tables of a database. Critical business policies, such as "don't delete customers with open orders," can be enforced easily across all applications and users.
- A special type of stored procedure, known as a *trigger*, can be associated with particular tables, and automatically initiated whenever attempts are made to modify these tables. Triggers can protect data, take actions based on changes to a table, keep summary data up-to-date, and enforce business rules.
- Comprehensive user-level security protections are implemented for database objects and commands. Special restrictions allow the system administrator to restrict particular columns, tables, and procedures to specific users.

Connecting to Enterprise Data

SQL Server connects to relational and nonrelational data anywhere in an enterprise, using gateways to access remote systems. This capability allows SQL Server to provide connectivity to users who want to use their existing workstations while tying their LANs into a variety of minicomputer and mainframe data sources. For example, Database Gateway for DB2 from Micro Decisionware® allows SQL Server applications to access data stored in DB2, IBM's strategic mainframe relational database management system.

Wide Variety of Front-end Tools

SQL Server databases can be integrated into an EIS through a broad range of front-end tools that allow custom user interfaces to be built. You can use Microsoft Excel with Q+E, Microsoft Access, Visual Basic with the Visual Basic Toolkit for SQL Server, or a number of other
third-party interface tools to build graphical front ends through which EIS users can access corporate data.

Microsoft Open Data Services

One of the most important technologies for accessing enterprise-wide data from client personal computers is Microsoft Open Data Services (ODS). ODS is a specification that allows gateways to be built between SQL Server and a wide range of other databases on MVS, UNIX, VAX/VMS, and AS/400® systems.

You can build ODS-compliant gateways yourself using the ODS developer's toolkit provided with SQL Server, or you can purchase an ODS gateway from a variety of third-party vendors. These gateways are independent of the network and allow personal computers running the Windows environment to access data in DB2, Oracle, Rdb, SQL/400®, and other databases.

The Open Data Services developer writes code (using the Microsoft C/C++ professional development system) to take specific actions and to access particular data sources based on incoming requests from a client or server. The code handles general classes of requests called *events*, such as connect, language, and disconnect events. When a client requests data, ODS passes the request to these custom routines, and then routes the reply back to the client application over a network. To the client, the data looks as if it were coming from SQL Server.

DCA/Microsoft Communications Server

Although LAN Manager can work with a wide range of gateway products, not all of these gateways have been optimized to allow an efficient link between client-server applications and mainframes.

In cooperation with Digital Communications Associates (DCA), Microsoft has developed an advanced communications gateway that is optimized to work with LAN Manager and employs a full client-server architecture.

The DCA/Microsoft Communications Server (Comm Server) runs on top of LAN Manager on the server and needs only a small application layer on each client (MS-DOS or OS/2). In addition, a communications card (typically an SDLC or token-ring card) must be installed in the server.

Comm Server provides 3270 emulation, 3287 printer emulation, and support for IND$FILE, LU6.2, HLLAPI, SRPI X.25, QLLC, CSV, and other protocols. This broad range of connectivity allows your EIS application on the user's desktop to connect to virtually any mainframe in your organization and download data from DB2 databases and other sources.

CHAPTER 10

Implementing the Data Source Layer

So far, you've seen some of the details of building a user interface for an EIS, and how to access corporate data. You've also seen how spreadsheets, documents, databases, charts, text files, and other information "containers" can be thought of as potential data sources for an EIS.

In this chapter, the data source layer will be examined more closely and you'll see why organizing and optimizing data storage is a critical part of any EIS.

What Is a Data Source?

In Chapter 3, a data source was defined as any source that can provide data to your EIS. Under this very broad definition, a data source can include:

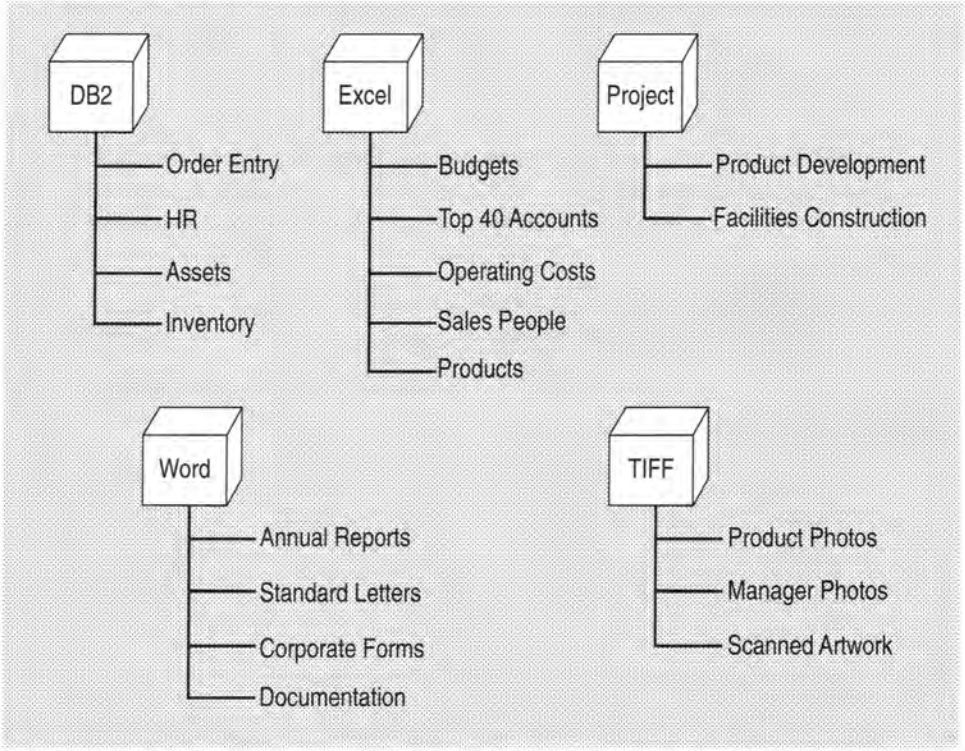
- Any application in the Windows environment that can provide data to an EIS. This includes applications such as Microsoft Excel, Microsoft Word, Microsoft Project, and others.
- Server-based applications such as SQL Server.
- A minicomputer- or mainframe-based application such as Rdb or DB2.
- Files in a variety of formats that aren't necessarily associated with any particular applications. These include "neutral" format files such as tagged image file format (TIFF) and rich text format (RTF) files that are readable by a large number of applications. Even simple ASCII text files fall into this category.

The definition of data source needs to be broad because this is typically how data exists in a corporation—in a wide variety of containers, formats, and locations. Although virtually anything is a potential data source, some are more useful than others. For example, applications that support DDE and OLE can easily be integrated directly into the EIS and become part of the solution.

In one sense, the data source layer of your EIS is already "implemented" in some form because data exists in various sources throughout a company. However, this data is not necessarily stored in the most efficient manner for an EIS. Therefore, implementing the data source layer means taking existing data sources and optimizing them to work more efficiently with an EIS.

Some Preliminary Steps

Identifying the various data sources in your organization is the first step toward implementing an effective data source layer. You can prepare a simple list of these, or you can sketch a diagram of all the data sources you think might play a role in your EIS. This allows you to visualize how data is currently organized in your company and will offer clues as to how you can organize it more effectively. For example, the following illustration shows what such a diagram could look like.



In the preceding illustration, some of the data sources listed will play a more important role than others. Additionally, there are probably relationships or dependencies that exist between the major data sources. For example, some of your project-related information may contain budget data from your spreadsheets. If possible, show these relationships and dependencies in your diagram.

For each type of data in your organization, there are three major considerations:

- **Organizing** How the data will be logically structured within its host application
- **Staging** How the data is strategically located in the enterprise
- **Formatting** How the data is physically structured within its various files

Each of these has a direct impact on your EIS development effort, as well as on how your EIS will perform once it is implemented.

When you consider a particular data source such as a group of spreadsheets or a database, you should ask the following questions:

- Will the organization, staging, and format of the data simplify the development of the EIS features that use it?
- Will the organization, staging, and format of the data make its retrieval by an EIS fast and efficient?

Data sources for which at least one of the answers to the above questions is "no" are potential problem areas that should be addressed early in the design phase of an EIS development effort.

Organizing, Staging, and Formatting Your Data

There is often a feeling among corporate developers and systems consultants that a corporation's data architecture is "sacred" and can't be changed. As a result, it is sometimes assumed that nothing can be done with the existing architecture to make it more efficient.

It is true that changing corporate data to suit the needs of an EIS can be disruptive or lead to unwanted results. But there are a number of things you can do to optimize the way an EIS uses corporate data that requires no significant changes to the existing data architecture. In this section, three such optimizations will be considered: organizing, staging, and formatting.

Organizing Data

Your data's organization is how the data is logically structured within the application. For example, a single database can consist of various tables whose fields are relationally linked to other tables in the database. There are many different ways of structuring these tables, and the overall design of these tables defines the data's organization.

As another example, consider a Word document that contains various charts and pictures. Some of these charts and pictures might be linked objects, while others are embedded. This also defines a unique organization of data.

Data is typically organized within an application to best suit the way it is currently used. Unfortunately, this organization is not necessarily optimal for an EIS. For example, a database's design may be optimized for a transaction processing application such as an inventory system. In this case, individual product inventory records in the database are updated frequently, but this design is usually not very efficient at processing queries for inventory summary reports from EIS users.

While designing your EIS, it is important to think about how the organization of your data relates to the features you are trying to implement. Later in this chapter this will be illustrated with an example.

Staging Data

When accessing corporate data, two specific performance problems are commonly encountered by EIS users:

- The time needed to search for data within its source and process it for use by the EIS can be longer than desired.
- Accessing remote data over slow transmission media (such as phone lines) can be time-consuming.

A good example of these problems is accessing mainframe data. Since mainframe databases are commonly used to store corporate data, they are a major source of information for an EIS. However, querying large corporate databases to obtain summary information (which is typically what EIS users want) can take a significant amount of time. Furthermore, although some personal computers enjoy "direct" connection to mainframes over a variety of fast transmission channels, some EIS clients must access mainframe data using a relatively slow-speed modem.

A common technique that is used to solve these two problems is called *staging*. Staging refers to the process of reorganizing, relocating, and/or processing the data your EIS needs to access —before it is requested. In this manner, data can be accessed faster by EIS users.

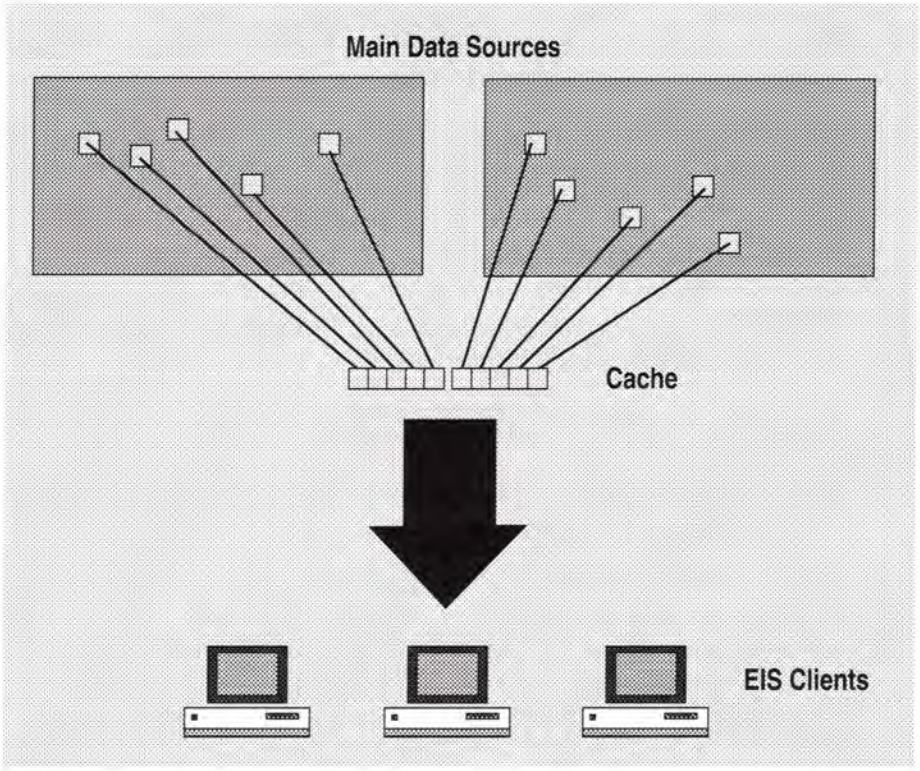
A significant advantage of staging is that you don't need to restructure the data in the original data source, or make duplicate copies of it. Instead, staging involves the creation of a high-performance data "cache" that contains preprocessed information your EIS is likely to need the most frequently.

Why Caches Are Effective

Caches can significantly reduce data query time and data transmission time by keeping the most frequently accessed information in a format and location optimized for fast access by EIS users. The concept of an EIS cache is similar to that of a RAM cache in a personal computer. It is based on the fact that about 90 percent of the requests for data can be satisfied using only about 10 percent of the total available data.

Only a small percentage of the total data is actually used by an EIS because EIS applications tend to focus on presenting summary information, rather than all the available details. Keeping summary information in a cache therefore satisfies most of the requests for data. For example, a company might have a mainframe database that contains product sales information that is updated on a daily basis. However, in most cases, users will not be concerned with daily figures, but rather with monthly sales volumes. The monthly totals constitute only a small fraction of the total sales data.

Without staging, it would be necessary to constantly query the original source of information and perform all the calculations required to summarize the data each time a user wanted monthly sales results. This forces the mainframe to use its resources to service frequent EIS requests, which is unnecessary and it could take a long time for results to appear.



Cache Types

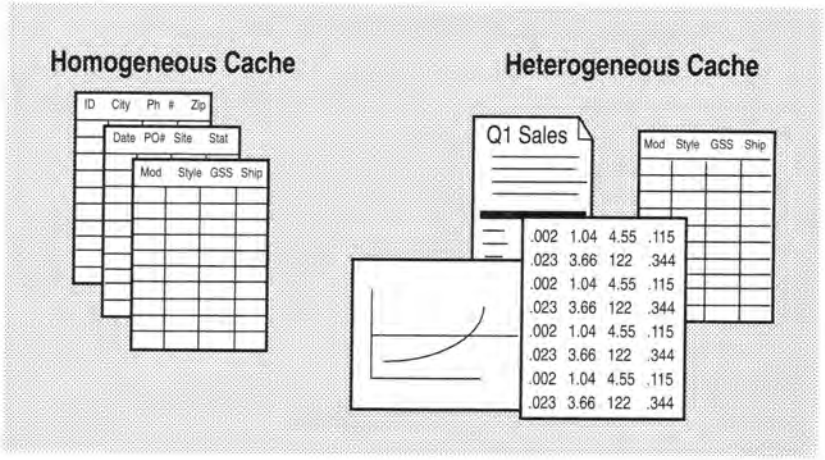
There are two major types of cache:

- Homogeneous
- Heterogenous

A *homogeneous cache* contains information structured in a consistent manner under the control of a single application. The most common type of homogeneous cache is a database (such as Microsoft SQL Server or Microsoft Access) where information is stored in a series of tables and records. For example, a database that contains summary sales data from a mainframe and summary inventory information from a minicomputer is a homogeneous cache. A group of Microsoft Excel worksheets containing summarized budget data is another example of a homogeneous cache.

In contrast, a *heterogeneous cache* contains different types of data formatted in different ways by different applications. For example, a heterogeneous cache might contain a database of sales information, frequently needed documents (such as report templates), budget spreadsheets, and Gantt charts for key projects. The physical location of the cache is usually a single directory on a server, or a series of directories. A heterogeneous cache can therefore be thought of as an aggregate of homogeneous caches.

As with the homogeneous cache, a heterogeneous cache should contain preprocessed, preformatted, or summary information that makes accessing the data faster. For example, a heterogeneous cache could contain a spreadsheet consisting of budget totals for every department in a company without listing the line item details.



Organizing Cache Data

Staging data is most effective when the information stored in the cache is structured to match the way EIS users request data. One way to determine the contents of your cache is to consider the natural hierarchies in your data. For example, suppose your EIS needs to highlight sales data by geography, but the actual data in the main data source is stored by postal code. Since most EIS users want to see the data at a higher level of detail, you can take advantage of the fact that a natural hierarchy for your data is Country, State, Province, City.

You can therefore place country, state, province, and city data in the cache, and give EIS users the ability to view sales by each region. The total number of records for the country, state, province and city data will be far less than the total number for all postal codes, so this is an appropriate use of caching.

Ideally, your system should give users the option of viewing raw transaction data if needed. For example, if summary data in the cache shows that product sales are below normal for the month, an EIS user could query the transaction data and display sales for all zip codes in that city. This may help pinpoint the source of the problem. Since this data is not located in the cache, the query would involve connecting directly to the main data source. An EIS may therefore require dynamic access to a small fraction of the entire set of source records, but this operation will occur infrequently and without a major impact on the computer that stores the source data.

Synchronizing Cache Data

Another requirement of staging is ensuring that data in the cache is up-to-date with the main data source. Updating information in the cache is done in a period of time that makes sense for how your EIS displays data. For example, if your EIS displays sales data by quarter and month, sales data in the cache should be updated at the end of each month. Other data may have to be updated much more frequently. For example, performance and trend data for a stock portfolio may have to be updated once a day, or even more frequently, depending on how the information is being used.

Although sales and stock data are usually updated regularly, some data may require irregular updates. For example, if your cache contains summary budget information, you may have to update this data whenever budgeting has been completed.

These data update periods should be considered while designing your cache. In some cases, an update can be triggered automatically by dates and times. In other cases, an EIS administrator may have to manually trigger the update. If possible, it is usually better to have updates occur automatically.

Implementing a Staged Data Cache

Caches often require some tuning or refining before they work with maximum efficiency. This tuning should be done during the cache implementation process, rather than after the system is installed. To accomplish this, it is often best to implement a data cache using a series of gradual steps as follows:

1. **Create a static cache** Create a small, static "dummy" cache by performing a set of one-time queries on the main data source and building tables in the cache using the results of the query.
2. **Connect it** Using data access tools discussed in previous chapters, connect the cache to the user interface of your EIS. The goal of this step is to ensure that you can access all the cache data from the EIS.
3. **Test it** Test the cache with the user interface of your EIS to make sure it meets the needs of the application. Although the data is never updated, it is good enough to test the user interface and determine if it meets performance criteria. It is usually a good idea to involve actual EIS users in the test.
4. **Create a dynamic cache** Write the code that allows the cache to be dynamically updated by periodically connecting to the main data source and downloading data. This mechanism needs to update only cache data that has changed, and will usually be activated when certain events occur, such as the start of a new month. Different categories of cache data usually have different update periods. For example, product market-share data might require monthly updating, stock exchange information might require hourly updating, and departmental budgeting data would be updated every time a new budgeting cycle occurs.
5. **Test it Again** Perform additional testing with the dynamic version of the cache to verify that it meets the needs of the application. Again, try to involve actual EIS users.
6. **Document it** Produce documentation containing all the procedures and assumptions needed to maintain the system. This should include documenting the rules used to collect and store data in the cache, describing the process used to update cache data, and explaining how cache data is related to the user interface features of the EIS.

Formatting Data

Each application has a "native" format for its data. For example, Microsoft Word has a file format for its documents that is different from the file formats of other word processing programs. Native formats are optimized for a particular application and are usually not directly usable by other applications.

Often, existing sources of data that you want your EIS to access will have formats that differ from the native formats used by your EIS applications. An important design consideration is therefore how your EIS will deal with non-native formats.

For example, a department in your company may use a spreadsheet program that is different than the one your EIS is based on, or there may be a large number of documents that were created by a word processing program that is no longer used.

It is a good idea to have a strategy for dealing with these different formats. You might want to convert all foreign formats to your EIS application's native format, or you could convert these formats each time the data is accessed. Usually, the former method is the best because formatting data repetitively can impact performance.

If you stage your data using a cache, you can reformat it upon loading it into the cache, which doesn't change the format of the original source data but still offers all the benefits of a format that is tailored for EIS querying.

Two Scenarios

Since every organization is different, it is difficult to give specific advice. However, a few examples will help clarify the concepts discussed in this chapter.

Scenario 1: A Large Mail-Order Business

Consider a mail-order catalog business that has a custom order entry system that utilizes a relational database managemant system (RDBMS) on a mainframe. As orders are received, they are logged by customer service representatives who type in the date of the sale, customer ID, product number, purchase order number, quantity ordered, product price, and other items that describe the sale. A typical order volume for this company is 7000 orders per month, and all sales orders are kept online in the database for 12 months before being archived.

Also suppose that this company keeps its inventory data in a different RDBMS on a minicomputer located at the main warehouse.

The company's marketing staff has requested that an EIS be built to perform the following functions:

- Review daily inventory data to ensure that manufacturing ships enough product to the warehouse to prevent backlogs.
- Track weekly order volume to assess the number of customer service representatives needed to handle calls.
- Track monthly sales revenue by product, city, and customer to improve the ability of the marketing staff to make decisions regarding where the company's marketing budget should be allocated.

To implement this, one solution is to build an EIS that allows users to connect to the mainframe and download sales results to Microsoft Excel where they can be displayed graphically and manipulated.

But this solution has the following problems:

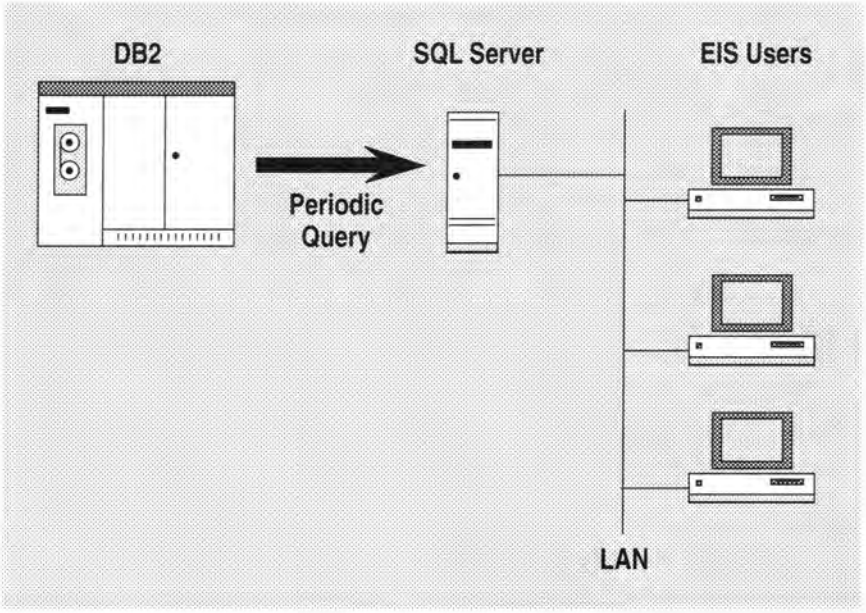
- There are nearly 100,000 records in the order entry database. Each time an EIS user wants to view sales data, the mainframe must process a large, CPU-intensive query that consumes a significant percentage of the mainframe's resources.
- Only a small fraction of the total information stored in both databases will be needed by EIS users. Querying for this small amount of data from such large sources is inefficient for the EIS, and may result in performance problems.
- EIS users will probably not find performance satisfactory because each query takes a long time to process. Users who are not directly connected to the mainframe or minicomputer will experience even worse performance because data must be transmitted over telephone lines at slower speeds.

Clearly, the two main data sources may be fine for order entry and inventory control, but they are not optimized for EIS. Part of the problem is the way data is organized in the databases. For example, the order entry database is divided into only two relational tables, one for customer data and the other for purchase order data. This makes it inefficient for the EIS to extract revenue-related information because the sales data is tied up with nonrevenue data such as purchase order numbers, order quantities, and product prices.

A Possible Solution

To solve our mail order company's problems, we can stage all the data needed by the EIS in a SQL Server database. See the preceding section for a description of staging data.

The solution is shown in the following illustration.



In this solution, weekly order volume and monthly sales revenue are updated appropriately, while inventory data is queried daily. The results of the query are stored in separate SQL Server database tables on a fast server personal computer that can be accessed by EIS users over a local area network.

This solution has many benefits:

- Since sales data is queried once a week (at most) instead of on-demand, the mainframe is off-loaded from the task of handling frequent queries of its database. Similarly, the minicomputer is queried once per day (possibly after normal work hours). Mainframe and minicomputer performance are therefore not degraded, and EIS performance is optimized.
- The results returned from the periodic queries contain only data immediately relevant to the EIS. All of the extraneous order entry information and inventory information is not included, making handling of the data by the EIS faster and more efficient.
- EIS users get their data faster because they can access it over a fast local area network instead of a telephone line.
- The solution is scalable. As more groups of EIS users are added on different networks, relatively inexpensive servers can be added, each of which contains the staged data. As a result, performance does not degrade when new users are added.
- There are a large number of front-end tools that you can use to access data on SQL Server from a personal computer running Windows, including Microsoft Excel with Q+E, Microsoft Access, and Visual Basic.

In this solution, SQL Server offers an alternative data source that requires no modification to the existing software (or data) of the existing order entry system and inventory systems.

Scenario 2: A Medium-Sized Law Firm

For the second scenario, consider a law firm consisting of 60 lawyers in three cities. Currently, the firm stores all its operations data (including client billings, operating costs, and human resources information) in Microsoft Excel spreadsheets.

The other major source of data is the large quantity of documents, including letters to clients, contracts, and invoices. The firm is replacing its word processing program with Microsoft Word, but almost all of the original documents are still in the native format of their old program. Although Word is capable of reading the older format, the original word processing program cannot read or write the new Word format.

The firm is growing quickly, with plans to hire another 20 lawyers over the next 12 months. Because of this growth, the managing partner has asked his information systems team to build an EIS that will enable better visualization of trends in client billings, expenses, and profitability. The managing partner also wants the EIS to allow attorneys to quickly browse through abstracts of existing contracts so that previous work will not be repeated. Additionally, he wants the EIS to be able to automatically create new contracts from old ones when attorneys enter information describing what the new contract should include.

The following are a few problems the EIS design team is considering:

- As client billing records grow into the many thousands, it will probably be necessary to move them into a database program.
- Contract documents are spread over a number of servers and are not well-organized.
- There are no document tracking tools that will allow the attorneys to quickly find contracts based on subject matter.
- Although the firm has decided to standardize on Microsoft Word, some of the attorneys will continue to use the old word processing program and must read and write documents in a different format than that used by Word.

A Possible Solution

To tackle the financial aspect of the EIS first, a quick analysis reveals that most of the existing Microsoft Excel spreadsheets such as operating costs and human resources information will maintain their present size or grow only slightly. There is no need to change these, especially since several macros have been written to manipulate the data.

The problem spreadsheet is the one that lists all client billings. This is already outgrowing the capabilities of a spreadsheet, and will grow much more. An excellent solution is Microsoft Access, which can easily handle the thousands of billing records while providing all the tools necessary to build an effective front end.

Microsoft Access also offers a solution to the contract tracking features of the EIS. All existing contracts can be entered into a separate database and tracked in a table that includes the following fields:

CONTRACT ID	DATE	CLIENT	ATTORNEY	SUBJECT	ABSTRACT	LOCATION

A single copy of Microsoft Access on a server will be enough to handle the requirements of both the financial and document tracking aspects of the EIS. For simplicity, all contracts should be consolidated onto a single server with a high-capacity hard drive that can be accessed by attorneys over a network.

Since the firm has standardized on Word, the "automatic contract" feature of the EIS should be implemented using Word and WordBasic. This feature will use WordBasic macros to take input from attorneys such as whether it is a prenuptual agreement or a facilities maintenance contract; which general clauses it should include, such as indemnification or arbitration, and other information.

Once the basic aspects of the contract are entered by an attorney, the WordBasic macro will automatically create the contract. The problem is that all of the existing contracts are still in the native format of the older word processor, and some of the attorneys will continue to use the old word processor. To solve this dual format problem, the EIS team can consider converting all existing contracts to rich text format (RTF), a widely used "neutral" format that can be read by both Microsoft Word and the older word processing program.

CHAPTER 11

Implementing the Communications Layer

An effective EIS allows people to see things that normally would be missed. Trends, opportunities, warning signs, and many other things come to light through an EIS.

A natural requirement that arises from the ability to view important data is the need to discuss this information with other people and take action based on what is displayed. This is the purpose of the communications layer.

Implementing the communications layer can involve more than just allowing EIS users to exchange information. It can also be an important means of communicating EIS information to users who may not have the EIS system installed on their computer. For example, an executive might want to send an EIS chart showing the company's growth rate to a broader range of company employees than just EIS users. In the mail message containing the chart, the executive could include a note thanking these employees for their hard work. The employees need only have an electronic mail package installed on their system to receive the message.

There are many possibilities for implementing communications in your EIS, and this chapter describes a few of the most prevalent types.

The Communications Layer

Like the data provider layer, the definition of the communications layer is fairly broad because of the number of possibilities involved. Here, the word "communications" does not mean networks, cables, and protocols. Rather, in the context of an EIS, communications refers to communicating EIS-specific information with other people.

There are many different approaches to implementing the communications layer. Some of these approaches involve no programming, but others can require varying amounts of custom coding, depending on the functionality you are trying to achieve.

In previous chapters, a few examples of EIS communications were briefly discussed. For instance, you could place a "Send" button on a worksheet showing sales figures for the current fiscal quarter. Clicking this button might automatically send the chart to designated users, or it could display an electronic mail dialog box with fields that the user could complete with the names of all recipients, the subject of the message, and so forth.

Regardless of how you implement your EIS communications capabilities, the following two components can usually provide most of what you need:

- The messaging applications programming interface (MAPI).
- An electronic mail client such as Microsoft Mail or the Windows for Workgroups Mail client.

After an overview of each of these is given in the following sections, some possibilities for implementing communications within your EIS will be discussed.

An Overview of MAPI

MAPI provides a single, standard API that virtually any application can use to implement messaging features such as electronic mail. You can use this API within almost any programming system, including Microsoft Excel macros, WordBasic macros, Visual Basic, and C.

Just as the print subsystem in the Windows operating system allows all applications to take advantage of print services through a standard dialog box, MAPI allows applications to interface with the messaging subsystem in the same way. For example, both Microsoft Excel and Microsoft Word support MAPI, and can display a message dialog box that users can fill-in to send the currently open worksheet or document to another user.

Microsoft Mail uses MAPI to provide its basic messaging functionality and provides an implementation of MAPI that other applications can access directly.

MAPI also provides other functionality such as access to address books, which are customized lists of message recipients.

One of the most significant benefits of MAPI is its simplicity. MAPI functions are high-level (compared to most networking APIs), and allow you to implement sophisticated messaging features with just a small amount of code. You deal only with functions such as sending, receiving, and addressing messages. The underlying messaging system is completely transparent.

An Overview of Microsoft Mail

Microsoft Mail is an electronic mail system that lets users communicate with each other across major networks and with most electronic mail systems. Almost any information that can be created or manipulated in the Windows environment can be exchanged over a network using Microsoft Mail.

With Microsoft Mail, you can:

- Send or receive any kind of file or document to other Microsoft Mail users, or to users of other mail systems using gateways. Examples of things you can send and receive are word processing documents, spreadsheets, pictures, programs, text files, and anything else that can be stored as a file. You can send multiple documents in a single mail message.
- Cut and paste almost any information from other applications into mail messages that can be sent to other users. This includes many different types of objects that can be created in the Windows environment, such as charts, tables, images, sound clips, and other objects that conform to the OLE interface. Recipients of these messages can double-click on the objects to edit them if the application that created the object is present on their computer.
- Store, organize, and manage all correspondence with other users. You can organize all your ingoing and outgoing communications by subject matter using folders. Folders can contain other folders to create a hierarchical organization. You can also easily search for mail messages based on the sender's name, the subject matter of the message, date of the message, and many other criteria.
- Create custom address lists of recipients that can be referenced through a single name. For example, you can assign the names of 10 managers in the finance department to the name "financemgrs" and send a message to all 10 managers simply by addressing the mail message to "financemgrs."
- Read, respond to, and create messages while working in other applications.

Once installed, Microsoft Mail acts as more than a mail application. Since it supports Simple MAPI, applications can use Microsoft Mail to communicate with messaging systems. This integration with other applications is described later in this chapter.

Using a number of gateways, Microsoft Mail can exchange information with users of other mail systems. The following gateways are available for Microsoft Mail:

- SMTP (for UNIX)
- MHS
- X.400
- SNADS (for IBM DISSOS, OfficeVision™, and Vermation MEMO)
- PROFS® (for IBM PROFS).
- FAX
- Microsoft Mail for AppleTalk® Networks
- 3+Mail®
- MCI MAIL®

Microsoft Mail is independent of the underlying network architecture. It runs on most major personal computer networks, including Microsoft LAN Manager, Novell NetWare, Banyan VINES, and Lantastic. The Microsoft Mail client is also a component of Windows for Workgroups, which also supports Simple MAPI.

There are several different ways to integrate Microsoft Mail into your EIS for a range of sophisticated electronic mail capabilities. These are discussed later in the chapter.

Mail Transfer Formats

When implementing the communications layer, an important design consideration is how the data will be captured within the EIS user interface layer and made available to the communications layer. For example, when the user is viewing a table and wants to send the table to another user along with a brief text note, in what form will the information get from the EIS to the mail system for delivery to its recipients? Three typical formats are as follows:

- **Images** A *screen dump* of the data is captured and embedded into a mail note as a bitmap object. Screen dumps consist only of pixel information captured from the computer's video display and do not contain actual EIS data.
- **Files** A file containing the data is attached to a mail note.
- **Objects** The object is selected, copied, and pasted into a mail note as an embedded object.

Following is a description of these three formats in further detail.

Images

In the Windows environment, you can capture any image that appears on your screen. You can capture images with a number of image capture utilities, or you can use the PRINT SCREEN key on the keyboard to capture screen dumps and send them to the Windows clipboard. If you are not familiar with this capability, you can try the following steps to learn how it is done:

1. Start any application and resize its window to approximately half of the screen's size. You can also open a file from within the application to display some data, but this is optional. Resizing the window is not required; it only helps you see what is being captured in the next steps.
2. Start the Windows Clipboard Viewer (available only in Windows 3.1) located in the Accessories group and maximize its window.
3. Press ALT+TAB to switch back to the application you started in step 1.
4. Press the PRINT SCREEN key on the keyboard. PRINT SCREEN is sometimes labeled differently on different keyboards, such as "PRNT SCRN."
5. Press ALT+TAB to switch to the Clipboard Viewer. An image of the entire screen appears in the Clipboard Viewer window.
6. Repeat steps 3 through 5, but this time hold down the ALT key while you press the PRINT SCREEN key (in step 4). When you examine the contents of the Clipboard View, only the application window appears.

You can therefore use the built-in image capture capability of Windows to capture either the entire screen (PRINT SCREEN key) or the active application window (ALT+PRINT SCREEN key). Since the image is sent to the Clipboard when it is captured, you can switch to Microsoft Mail and paste the image into a mail message to send to other users.

Note You don't have to start the Clipboard Viewer accessory to copy information to the clipboard. This was done in the previous example only to show the information that was copied to the clipboard.

The primary advantage of sending images rather than files is that users don't need to have the application that created the image to view the data. A user can capture an entire screen of information from the EIS and send it as an image to other users. This image can be copied from the message by selecting the image, choosing the Copy command from the Edit menu of Microsoft Mail and pasting it into any application that accepts Windows bitmap images.

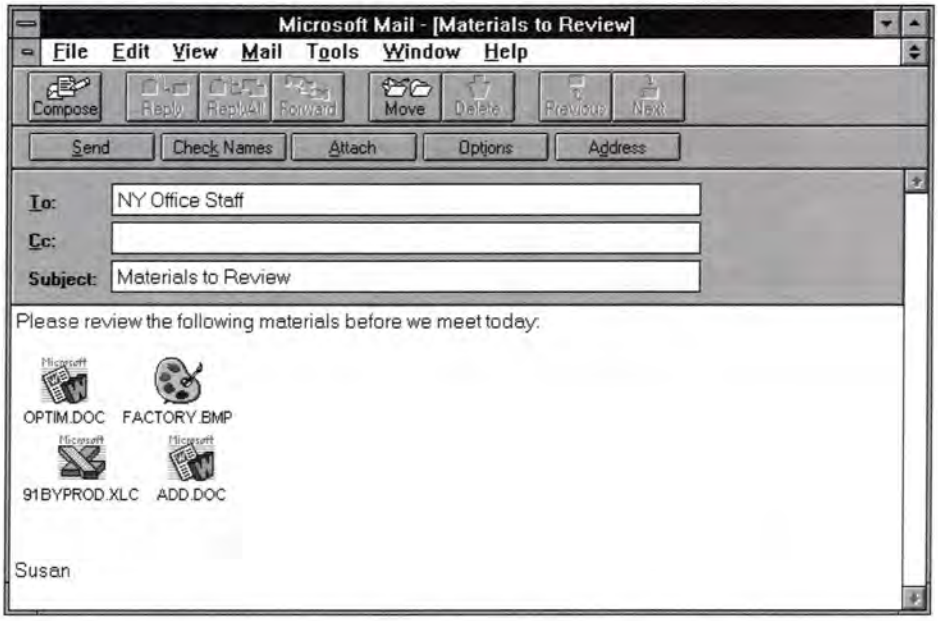
For example, a Microsoft Mail user can copy an image of a chart from their mail message and paste it into a report they are creating with Microsoft Word.

The two disadvantages of sending bitmap images through the mail system are:

- The EIS data displayed in the image is simply a picture of the data—not the actual data itself. The recipient therefore can only view the data and does not have a means to access or manipulate the actual EIS data values.
- Bitmap images are typically quite large, and sending them frequently through the mail system can add significantly to network traffic. Also, multiple bitmap images can consume a considerable amount of space on a mail server and on the user's local file system.

Files

With Microsoft Mail and Windows for Workgroups, you can attach one or more files to a mail note. The text of the message (if any), along with the files are then sent to the list of recipients. Each attached file appears within the mail note as an icon (as shown in the following illustration). By double-clicking an icon within the message, the application that created the file is automatically started, and the data can then be edited, copied, pasted, or manipulated as any data would normally be manipulated within that application.



In contrast with sending an image, the advantage of sending a file is that actual EIS data can be exchanged. Recipients can use this data as if it were created locally in their application. Sending a file to another user requires that the application that created the data be installed on the recipient's computer.

Objects

Object transfer involves embedding individual objects within mail messages. This is different from file transfer because you are sending an object, not a file. With Microsoft Mail, you can embed multiple objects in a single mail message.

In this context, objects can be of any format that conforms to the Windows object linking and embedding (OLE) specification, or a format that can be supported by the Windows clipboard. Examples of objects are charts, tables, sound clips, and bitmap images.

You can only embed objects within mail messages. Linking is not possible because once the objects are sent to another user, there is no way for the recipient to access the original information on the sender's personal computer.

The main disadvantage of object transfer is that different objects sometimes behave in different ways, depending on how the application handles objects. Some can be edited by double-clicking them in the mail message, but this is not always the case. This problem can be handled within an EIS by allowing the user to work only with certain types of objects whose behavior is known.

Using Mail and MAPI — The Possibilities

Using Microsoft Mail, Windows for Workgroups Mail, and/or MAPI, you can implement a broad range of communication capabilities in your EIS. These capabilities generally fall into three categories:

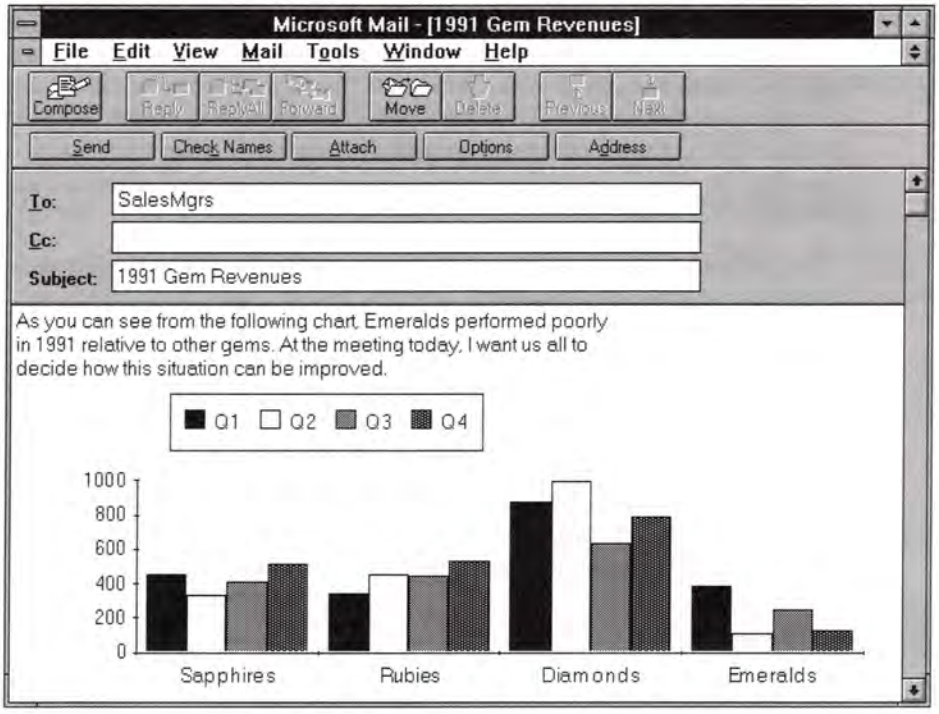
- **Standalone** A Microsoft Mail client is installed on each EIS user's computer, but is not integrated with other components of the EIS. You can give EIS users an electronic mail capability with no development work. However, it requires that users know how to get information out of the EIS system and send it through the mail system.
- **Standard Integration** You can integrate Microsoft Mail into your EIS by allowing users to invoke it directly from certain *messaging-aware* applications that are part of the EIS user interface. A messaging-aware application is one that can interact with a MAPI messaging system. For example, an EIS user could click a "Send" button on a Microsoft Excel report, which would display a Simple MAPI Send Note dialog box. The user could type in the names of people to receive the report and the subject of the message, and the report could be automatically embedded in the message. Standard integration involves less knowledge of electronic mail than the standalone option, but requires users to interact with the mail system.
- **Custom Integration** This usually involves a more "seamless" approach that involves a completely custom mail interface with functionality that reduces the amount of knowledge a user must have about electronic mail or the EIS. For example, instead of using a "Send" button in the manner described previously, the button could say "Send to Regional Managers," and could automatically send the report to all regional managers without requiring any interaction with the user interface of the mail system.

Each of these solutions will now be described in greater detail.

Standalone Mail

To implement standalone electronic mail capabilities in your EIS, you only need to install Microsoft Mail or Windows for Workgroups Mail on an EIS user's workstation. In standalone mode, an EIS user will start Microsoft Mail separately from the EIS and then switch to it when the user wants to send a message, object, or document.

For example, suppose an EIS user is viewing a Microsoft Excel chart that summarizes human resources growth for the company over the past year. If this user wants to send the chart to another company employee with a text message attached to it, the user could select the chart, copy it to the Windows clipboard, switch to Microsoft Mail or Windows for Workgroups Mail, create a new message, and paste the chart into the message. The user would then fill out the various "To," "CC," and "Subject" fields of the message and choose the Send command. The following illustration shows a completed Microsoft Mail message, in which the user has copied a Microsoft Excel chart object.



Typically, users will capture data in either image format or object format before pasting it into a mail message. Although it is possible to attach actual files, EIS users typically do not know the names of the files that contain the EIS data.

Using a standalone mail package makes sense if you cannot invest the time to develop a custom mail solution or if the appropriate programming expertise does not exist. It is a simple means of providing communications capabilities to EIS users without having to write a single line of program code.

One strategy is to install a standalone mail capability as a temporary solution while you are developing custom e-mail. The first version of your EIS could therefore include a standalone mail package, while subsequent versions could contain standard mail integration or a fully custom approach.

The main disadvantage of the standalone approach is that users must know how to incorporate EIS information in a mail message. For example, if a user is looking at a quarterly report, the user would have to know how to select information in the report, copy it to the Windows clipboard, switch to the mail application, paste the information into a mail message and send the mail message to the appropriate people. For some users, this may require additional training.

Standard Integration

Implementing standard integration with Microsoft Mail is a fairly simple process in most cases. Standard integration relies on the fact that many applications such as Microsoft Word and Microsoft Excel support Simple MAPI. For example, when you install Microsoft Excel onto a computer that already has Microsoft Mail installed, the File menu of Microsoft Excel contains a Send command, which allows you to send the currently open file to other users.

Implementing standard integration with Simple MAPI in your EIS can therefore be as simple as retaining the Send command on the menus of messaging-aware applications that comprise the EIS. When the user is viewing data within the EIS and wants to send what is displayed to another user, the user can select the Send command. For example, suppose a user is viewing a Microsoft Excel chart and wants to send this chart to another user, along with a brief text message. By choosing the Send command on the appropriate menu, a standard mail form would be displayed, containing the chart as an icon.

The user simply fills in the fields of the form and chooses the Send button to send the message to the designated recipients.

Standard integration offers a more seamless interaction with the mail system than standalone integration because it doesn't require users to know how to copy and paste images, or switch between applications. However, standard integration requires that users know how to address messages to the correct recipients, and it also displays a separate application window.

Standard integration also requires a small amount of user interface design work. For example, which applications in your EIS should have the Send item enabled? If your EIS has a custom menu, which menu should contain the Send command?

Custom Integration

The two main goals of a fully custom mail capability are:

- Make the user's interaction with the mail system as transparent and effortless as possible. This includes automating many of the basic mail functions such as filling in address fields, attaching documents, and entering text messages.
- Offer additional functions or user interface features not already offered by the underlying mail application.

Custom integration is different from standalone or standard integration because it involves some programming. In some cases, this programming can be a simple macro attached to a button or it can involve direct access to the Simple MAPI programming interface through other programming languages such as Visual Basic or C.

The following example shows a Microsoft Excel macro that could be attached to a button on a worksheet. In this macro, the SEND.KEYS formula sends text and field control commands to a buffer. When the SEND.MAIL formula is executed, the "To" and "Subject" fields of the message are automatically completed using the text and field control commands stored in the buffer. The distribution list named "SalesMgrs" is entered in the "To" field, and the "Subject" field contains the text "Sales Report."

The field control commands (such as {TAB} and {HOME}) move the insertion point from one field to another or within fields. This macro even types a message about the chart and places it in the message text area.

For more information on using SEND.KEYS and SEND.MAIL, see the *Microsoft Excel Function Reference*.

Names	Formulas	Comments
	SendMail	
	=SEND.KEYS("SalesMgrs{TAB 2}+{HOME}Sales Report{TAB}The attached chart shows year-to-date revenue for the medical products division%S")	Automatically fills in all the fields of the mail message.
	=SEND.MAIL()	Sends the currently open worksheet
	=RETURN()	

Although custom integration requires programming, the possibilities for implementing custom integration are quite large. You can send data in image, file, or object format using a custom solution. In the next section, the primary means of implementing custom integration (Simple MAPI) will be discussed.

Simple MAPI — The Key to Custom Communications

There are two basic MAPI interfaces: Simple MAPI and Extended MAPI. Simple MAPI enables an application to send, address, and receive messages. These messages can include attached documents or even OLE objects.

Currently, only Simple MAPI is available through Microsoft Mail or Microsoft Windows for Workgroups. Extended MAPI offers a larger set of API functions, and will be available in the future.

In most cases, Simple MAPI will be all you'll need to implement a complete electronic mail system for your EIS. The following table shows the simple MAPI function names for the Visual Basic language.

Function	Description
MAPILogon	Begins a session with the messaging system
MAPILogoff	Ends a session with the messaging system
MAPIFindNext	Returns the ID of the next (or first) Mail message of a specified type
MAPIReadMail	Reads a mail message
MAPISaveMail	Saves a mail message
MAPIDeleteMail	Deletes a mail message
MAPISendMail	Sends a mail message
MAPISendDocuments	Sends a standard mail message
MAPIAddress	Addresses a mail message
MAPIResolveName	Displays a dialog box to resolve an ambiguous recipient name
MAPIDetails	Displays a recipient details dialog box

MAPI function names maintain their consistency throughout most programming environments. For example, the function names listed in the preceding table are exactly the same when programming with the C language. For more information about how to use MAPI within programming environments such as Visual Basic, see the *Microsoft Mail Technical Reference* or *Workgroup Programmer's Reference*.

Extended MAPI

In the future, extended MAPI will provide many additional API functions that provide even greater interaction with messaging systems. You can use Simple MAPI today to provide a wide range of communications capabilities in your EIS and migrate your applications to extended MAPI when it becomes available.

For More Information

If you want more information, you can call (800) 426-9400 to request a technical paper on MAPI. The part number of this technical paper is 098-35738.

CHAPTER 12

A Glimpse at the Future

Building an EIS application involves an investment of time and money. Depending on how sophisticated the application is, this investment can be substantial. Therefore, it is fair to ask what will happen to that investment in the future. Will the EIS solution that you build today last long enough to be profitable, or will it become obsolete as a result of new technologies?

In this section, questions such as these will be answered, and a few upcoming technologies will be considered.

Your Investment in EIS Is Important

From Microsoft's perspective, few things are as important as our customer's investment in software. With this in mind, we are designing future generations of applications and operating systems that not only act to preserve your investment in EIS software, but provide the capability to extend it far beyond its present level.

We will do this by offering new technologies that are compatible with their predecesors yet offer greater power, more features, and are easier to use.

Upcoming Technologies

Over the next several years, Microsoft will deliver a wide range of new software products that can be used to enhance the power of existing EIS applications. These products will consist of new versions of Microsoft's applications such as Microsoft Excel, Microsoft Word, Microsoft Project, Microsoft PowerPoint, and new versions of programming systems such as Visual Basic. Microsoft will also deliver major new additions to system-level technology.

In this chapter, we'll take a brief look at some upcoming technologies and explain their impact on EIS applications.

ODBC and MAPI

A major part of Microsoft's long-term vision for the Windows environment is the development of a set of standards that enable Windows applications to connect to a wide variety of resources in a standard way.

Known as the Windows open data services architecture (WOSA), this set of standards greatly simplifies the development of applications on personal computer desktops by specifying a single, well-defined API for a wide range of functions.

Two WOSA standards are of particular interest to developers of EIS applications in a client-server environment:

- Open database connectivity (ODBC).
- Messaging applications programming interface (MAPI).

ODBC enables client applications in the Windows environment to connect to many different databases directly as well as through *gateways*, which are specialized applications that "translate" database queries from one format to another. Based on the call-level-interface specification proposed by the SQL Access Group (a standards group made up of various database application vendors), ODBC was designed to work with a wide range of DBMS systems from different vendors.

Instead of using a different API to access each database, or developing a single API that taps only the "lowest common denominator" of a number of different databases, ODBC's robust API enables applications to take advantage of a database's full capabilities.

In the same way that ODBC provides standard access to databases through a single API, MAPI gives applications the ability to access a wide range of electronic messaging services. Users of EIS systems can use MAPI to communicate almost any kind of information to other application users anywhere in the enterprise or around the world.

EIS application developers can develop custom MAPI solutions or use a variety of off-the-shelf electronic mail and messaging products.

A growing number of vendors have publicly endorsed both ODBC and MAPI.

Vendors that have endorsed ODBC

Andyne	Information Builders	PowerSoft
Apple Computer, Inc.	Informix Software, Inc.	Progress Software
Brio Technology, Inc.	Ingres Corporation	Raima Corporation
Bull HN	IQ Software	Retix
CincomSystems, Inc.	Micro Database Systems, Inc.	Rochester Software Connection
Computer Corp. of America	Microrim, Inc.	Siemens
Coromandel	Microsoft Corporation	Software AG
Database Gateway	Must Software	Sybase, Inc.
DEC	NCR/Teradata	Tandem
EASEL	Neon Systems, Inc.	Uniface Corporation
Fairfield Software	Novell, Inc.	Unify Corporation
Fox Software	Oracle Corporation	Vertisoft Research, Inc.
Fulcrum Technologies, Inc.	PageAhead Software	Watcom Corporation
Hewlett-Packard Company	Pioneer Software	

Vendors that have endorsed MAPI

AT&T	DEC	Novell, Inc.
Banyan Systems, Inc.	ExMachina	Prometheus
British Telecom	Hewlett Packard Company	SkyTel
CompuServe, Inc.	Microsoft Corporation	SoftSwitch

OLE 2.0

OLE version 2.0 is a major extension to the OLE 1.0 architecture that paves the way for new computing environments that are easier to use, yet more powerful. OLE 2.0 includes a wide range of new features, including:

- **In-place activation** Directly activate objects within documents without switching to a different window. This includes operations such as editing, displaying, recording, and playing. In-place activation will have a significant impact on the way EIS applications work. It will enable users to edit a wide variety of EIS data (such as charts, tables, and text) from a single application window. This will make EIS user interfaces even more seamless and intuitive.
- **Nested object support** Directly manipulate objects nested within other objects, and establish links to nested objects.
- **Drag and drop** Drag objects from one application window to another, or drop objects inside other objects.
- **Storage-independent links** Allows links between embedded objects that are not stored as files on disk. This will enable embedded objects within the same or different documents to update one another's data regardless of whether they are recognized by the file system.
- **Programmability** Enables the creation of command sets that operate across applications, as well as within them. For example, a user could invoke a command from a word processing program that sorts a range of cells in a spreadsheet created by a different application.

For users, OLE 2.0 means a more productive way to work with applications and with the operating system. Applications that take advantage of OLE 2.0 features will have greater ability to interact together seamlessly, enableing users to focus more on creating and managing information rather than on remembering how to perform procedures.

For developers of EIS solutions, OLE 2.0 represents a much more powerful way to build an EIS using multiple applications, and many more features to implement within an EIS.

Significantly, since OLE 2.0 is an extension of the OLE 1.0 functionality, applications that comply with the OLE 1.0 specification will remain 100% compatible with OLE 2.0 applications. This means that EIS solutions that currently take advantage of OLE 1.0 will be able to interact with applications that support OLE 2.0. More importantly, you'll be able to upgrade your EIS to use new versions of Microsoft applications that support OLE 2.0 and have little or no development work to take advantage of OLE 2.0 features.

Microsoft Windows NT

Microsoft Windows NT ™ is the next generation of the Windows operating system. Its features provide a new level of computing power for Personal computer users. Microsoft Windows NT is not a general upgrade from the 3.*x* family of Microsoft Windows operating systems. Rather, it is a completely new version of Microsoft Windows that will run applications developed for the Microsoft Windows 3.*x* product family.

Microsoft Windows NT is a fully 32-bit operating system that contains a wide range of sophisticated features including:

- **Built-in networking** Create sophisticated client-server architectures that deliver enterprise-wide information to the user's desktop.
- **Full pre-emptive multitasking** Execute multiple applications in a highly efficient manner.
- **Symmetric multiprocessing** Delegate computing tasks among more than one CPU, resulting in a substantial increase in computing performance as additional CPUs are added.
- **Multiple threads of execution** Enables an application to split its various tasks into a series of subtasks that can be scheduled for execution. To the user, multithreading makes the application appear to be doing more than one thing at a time. For example, you can interact with a worksheet while simultaneously performing a recalculation and printing a document.
- **Platform-independence** Microsoft Windows NT is not tied to the Intel® family of microprocessors. It can run on other chips, including RISC designs such as the MIPS® processor family and DEC Alpha chip set.

What Will Microsoft Windows NT Mean to EIS?

Since Microsoft Windows NT runs all applications written for Microsoft Windows versions 3.0 and 3.1, all existing EIS applications can run on Microsoft Windows NT and take immediate advantage of its greater power. Future versions of Microsoft applications that support the advanced features of Microsoft Windows NT, such as multithreading, are already being readied.

These new features will enable more powerful EIS applications. For example, you can build a program that uses a thread to receive real-time data from a stock quoting service, another thread to perform periodic recalculations on the spreadsheet that contains the stock portfolio performance data, and a third thread to handle user input. If the computer has more than one CPU, the three threads can execute in parallel on different processors instead of being *time-sliced* by a single CPU. Even with a single CPU, Microsoft Windows NT will be able to multitask these threads, giving the user the impression that they are being processed in parallel.

Equally important, the built-in networking capabilities of Microsoft Windows NT facilitate much easier and consistent access to enterprise-wide information. You can build a sophisticated EIS architecture that takes advantage of the client-server model while having access to data in minicomputer and mainframe environments.

Beyond Microsoft Windows NT

OLE 2.0 and Microsoft Windows NT bring more than just a set of new features to computing. They are the foundation of a new model of computing that will appear in a future version of the Microsoft Windows operating system. This future version of Microsoft Windows will build upon both OLE 2.0 and Microsoft Windows NT to offer a more intuitive user interface and a distributed object computing model that extends the power of objects to networks while hiding the complexity of network environments.

The extension of object technology across networks means that users will have access to almost unlimited information, regardless of its type or how it was created. Furthermore, this access will require almost no knowledge of where the information may be located. Users will therefore have more control over their environment and will be able to achieve more work with less effort by manipulating objects on their local desktop system or across networks.

CHAPTER 13

Other Sources of Help and Information

In addition to this guide and the Microsoft Open EIS Pak, there are many other sources of information available. If you have problems or questions at any stage of designing or building your EIS, you can turn to these sources for help.

The following numbers and addresses are for the United States. For information on the availability of these resources outside the United States, please call your local Microsoft subsidiary.

Microsoft Developer Services

Microsoft Developer Services (MDS) is a resource that offers information on different Microsoft development tools and services. Call MDS at (800) 227-4679 between 6:30 A.M. and 5:30 P.M. Pacific time to receive additional literature on any of the development tools mentioned in this guide. Or, you can write to Microsoft Developer Services Team, Microsoft Corporation, One Microsoft Way, RWF, Redmond, WA 98052-6393.

Microsoft Press

Microsoft Press offers a wide range of books on Microsoft operating systems and languages. You can look for Microsoft Press books wherever books and software are sold. For a free technical books catalog, call Microsoft Press at (800) 888-3303 between 8:00 A.M. and 5:00 P.M. Pacific time. Choose extension 65608.

Microsoft University

Microsoft University is a U.S. training resource that offers courses to help support engineers, systems administrators, and network integrators improve their efficiency and productivity. Hands-on technical training courses help programmers minimize the learning curve and maximize their ability to develop applications that take full advantage of the latest Microsoft systems software.

MSU offers courses in many of the technologies discussed in this guide. For more information, call MSU at (206) 828-1507 between 8:00 A.M. and 5:00 P.M. Pacific time.

CompuServe Information Service

You can obtain 24-hour online access to the latest general and technical information about Microsoft products and services through CompuServe Information Service (CIS). Microsoft sponsors forums on CIS in the Microsoft Connection. Within these forums, you can exchange messages with experienced users of Microsoft products as well as Microsoft product support engineers.

Up-to-the-minute information is available on many topics, including using the Microsoft Windows operating system, developing Windows applications, languages, network products, and applications. You can download useful sample programs, utilities, drivers, files, and development tools from the forums and from the Microsoft Software Library.

Technical support is also available through the Microsoft Knowledge Base, a collection of technical articles about Microsoft products. You can use the product problem/suggestion report form to make suggestions or to report problems with a Microsoft product.

To connect to the Microsoft Connection, type **GO MICROSOFT** at any ! prompt. For more information on CIS, call CompuServe at (800) 848-8199 between 8 A.M. and 10 P.M. Eastern time and ask for operator 230.

Fee-Based Technical Support

Microsoft Product Support Services provides a multilevel, fee-based technical support program for those who need ongoing in-depth technical support and for those who need to quickly resolve specific technical problems. Each account includes the Microsoft OnLine for Windows, the technical support software that lets you use Service Requests to communicate quickly and privately with Microsoft support engineers.

Payment options range from a per service request fee to unlimited yearly subscriptions. You can even request that a team of support engineers be assigned to your account. Microsoft Knowledge Base, a database of technical information about each Microsoft product, is available through Microsoft OnLine for Windows.

For more information on fee-based technical support, dial (800) 227-4679 between 6:30 A.M. and 5:30 P.M. Pacific time.

Microsoft Authorized Network Specialists

To offer the high-quality local support and service required for enterprise computing, Microsoft has established a worldwide organization of Microsoft Authorized Network Specialists. Before being authorized as Network Specialists, resellers must undergo rigorous screening, training, and certification. Well-qualified to offer on-site support, training, and consulting services, these specialists can help you put Microsoft networking products to their best strategic use in your business. For the name of your nearest U. S. Microsoft Authorized Network Specialist, dial (800) 227-4679 between 6:30 A.M. and 5:30 P.M. Pacific time.

Microsoft Consulting Services

To help companies make the best strategic use of client-server and graphical user interface technologies, Microsoft Consulting Services (MCS) offers four primary services, each tailored to the unique needs of individual clients: evaluation and planning, systems design, development, and implementation support. MCS is a rapidly growing organization that allows you to build EIS systems across a number of countries.

MCS is staffed by consultants with a wide range of experience and expertise, including senior consultants in the field of information technology management and knowledgeable specialists in individual, strategic technologies. Clients also have access to in-house development tools, services, and Microsoft research and development facilities for performance testing of customized applications.

MCS consultants have built EIS solutions for many clients, both large and small. They understand the many technical issues that must be addressed when building an EIS application.

For more information about MCS, please dial (206) 882-8080 between 8:00 A.M. and 5:00 P.M. Pacific time.

Third-Party Value-Added Solutions

A number of independent software companies have incorporated Microsoft products into their own solutions. Some of these may be useful when building an EIS application. For more information, dial (800) 227-4679 between 6:30 A.M. and 5:30 P.M. Pacific time.

Microsoft Product Support Services

Microsoft offers a variety of comprehensive support plans that are designed to meet many different needs. You get telephone support from expert Microsoft support engineers, electronic Service Request support, convenient access to the Microsoft OnLine Knowledge Base, Software Library, electronic bulletin boards, and much more. Microsoft also offers incremental fee-based support options. For more information about Microsoft product support, dial (800) 227-4679 between 6:30 A.M. and 5:30 P.M. Pacific time.

You can contact Microsoft Product Support Services with a Telecommunications Device for the Deaf (TDD) by dialing (206) 635-4948 between 6:00 A.M. and 6:00 P.M. Pacific time, Monday through Friday.

Microsoft Consultant Relations Program

This program can refer you to an independent consultant in your area. These consultants are skilled in macro development and translation, database development, and custom interface design. For information about the consultants in your area, dial the Microsoft Consultant Relations Program at (800) 227-4679, extension 56042.

Microsoft Inside Sales

For further information on any of the Microsoft products mentioned in this guide, or to order any of these products, contact Microsoft Inside Sales at (800) 227-4679.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®

1192 Part No. 35340