Essential Guide to PeopleSoft Development and Customization

Module One

- In Depth Programming in PeopleCode
- Using Application Engine in PeopleSoft
- PeopleTools Development
- Application Designer Techniques

Tony DeLia
Galina Landres
Isidor Rivera
Prakash Sankaran

MANNING

Essential Guide to PeopleSoft Development and Customization

Essential Guide to PeopleSoft Development and Customization

TONY DELIA
GALINA LANDRES
ISIDOR RIVERA
PRAKASH SANKARAN

MANNING

Greenwich
(74° w. long.)

For electronic browsing and ordering of this and other Manning books, visit http://www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact:

Special Sales Department
Manning Publications Co.
32 Lafayette Place Fax: (203) 661-9018
Greenwich, CT 06830 email: orders@manning.com

.

Manning Publications Co.
32 Lafayette Place
Greenwich, CT 06830

Copyeditor: Adrianne Harun
Typesetter: Dottie Marsico
Cover designer: Leslie Haimes

Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – VH – 03 02 01 00

brief contents

contents

about this book

The *Essential Guide to PeopleSoft Development and Customization* is an exhaustive, as well as practical, guide covering PeopleSoft 7.5 and many new features in release 8.0. Both novice and experienced programmers will benefit from the detailed coverage of topics ranging from the basics of Application Designer to the proper use of PeopleCode within the Application Processor. The book serves as both a reference and a tutorial and covers advanced topics that other books avoid. The reader can gain valuable expertise by following the exercises and building sample applications and utilities. Extensive coverage of PeopleCode, including scroll and function library examples, can be found as well as the methodology behind customization and upgrades. Discover how to effectively utilize SQR and Process Scheduler. Master various levels of PeopleSoft security. Most developers won't touch PeopleSoft COBOL programs with a ten foot pole. Expand your horizons by uncovering the secrets of PeopleSoft COBOL and the PTPSQLRT module and even walk through a sample customization. Application Engine is a powerful PeopleTool—but one of the least understood. Through a series of simple but effective exercises, the reader can learn Application Engine concepts such as dynamic SQL, decision logic, and dynamic sections. A useful Application Engine utility is produced which will enhance the delivered Process Scheduler panels. This book takes a soup-to-nuts approach leading the reader through the full cycle of application development.

The four authors provide the reader with the skills necessary to compete in the PeopleSoft marketplace for years to come. Special sections are included which provide detailed information on new features in PeopleSoft release 8. The reader should gain valuable insight into the next generation of PeopleTools. Exciting new features such as the new PeopleCode Debugger and PeopleCode dot notation, using a new series of object classes, are revealed. Also covered are Application Designer enhancements and improved Process Scheduler design and SQR support. See firsthand how Application Engine has been turbo-charged with a new line of meta-constructs, PeopleCode actions, and file handling capability, as well as a new integrated design. The authors' primary goal was not to be the first book on the market ... it was to be the best.

INTENDED AUDIENCE

This book is intended for both beginner and experienced PeopleSoft support personnel including:

- technical developers and consultants who develop, customize, or support PeopleSoft applications
- functional consultants and users who would like greater insight into the realm of PeopleSoft application development
- project leaders and managers responsible for PeopleSoft implementations, customizations, and upgrades.
- database administrators, network technicians, programmers, and all other technical personnel involved in PeopleSoft implementations and support
- computer specialists who would like to learn how to use the PeopleSoft development toolset through self-study.
- those who enjoyed *SQR in PeopleSoft and Other Applications* and would like to pick up where the book left off

HOW THIS BOOK IS ORGANIZED

There are seven major parts to this book. The reader will find it to be most effective by following the examples and exercises provided. Special chapters appear in some sections describing new features in PeopleSoft release 8. These include details on Application Designer, PeopleCode, SQR, Process Scheduler, and Application Engine.

Part 1 gives an overview of PeopleSoft architecture including comparisons between two-tier, three-tier, and web based architecture and their functions. Also included is an introduction to the PeopleSoft environment. Part 1 also describes development and administrative tools such as Application Designer, Data Mover, Security Administrator, and Tree Manager. The reader will become familiar with records, fields, panels, and menus as well as PeopleCode, projects, and upgrades.

Application Development comprises part 2 of the book. The reader will gain valuable insight while building a sample Problem Tracking application. Critical design elements and components are discussed and incorporated into your application. We apply enhancements using search records, derived work fields, PeopleCode, and push buttons. We also discover panel design features such as scroll bars, subpanels, secondary panels, and grids. We learn the difference between PeopleSoft and database objects. We also cover the application processor as it performs during search processing, data retrieval, and PeopleCode events. New Application Designer features found in PeopleSoft release 8 are presented.

Part 3 is an in-depth look at PeopleCode. Here the reader can find a basic overview along with detailed descriptions of the PeopleCode language and related components. Follow the examples which reveal the proper technique for accessing panel buffer fields, working with scroll bars and effective dates, using embedded SQL, and performing error handling procedures. Additional topics such as Security, Meta-SQL, and function libraries are discussed along with debugging techniques. New PeopleCode features found in PeopleSoft release 8 are presented.

Part 4 deals with customizing PeopleSoft-delivered applications. In this section, we determine when to customize and what impact customization has on future upgrades. Tips on performing an upgrade are given along with a discussion on the proper use of projects. Walk through several sample customizations and the steps required to successfully implement your derived modifications.

Part 5 discusses the use of SQR in PeopleSoft applications. Here we see how SQR programs are set up and run in the PeopleSoft environment. We find comprehensive coverage when using run control records and communicating with Process Scheduler and Process Monitor. Unearth the secrets behind implementing security levels and scheduling recurring jobs. New SQR and Process Scheduler features found in PeopleSoft release 8 are also presented.

Part 6 is an explanation of PeopleSoft COBOL and it's unique structure. Differences between conventional COBOL programming and PeopleSoft's particular flavor is discussed. Learn the fundamentals behind the PTPSQLRT module which is the driving force behind PeopleSoft COBOL. Individual PTPSQLRT actions are examined along with the required parameters for each. A realistic modification is performed demonstrating the concepts behind PeopleSoft COBOL. Additional facets such as Process Scheduler API, Configuration Manager and using trace files are discussed.

Part 7 serves as both a reference and tutorial into the world of Application Engines. After an overview describing the basic components and functionality of an Application Engine, the section offers the reader a series of exercises designed to demonstrate each A/E concept. During these simple exercises we cover decision logic and loop control, as well as how to access cache records, effectively utilize dynamic sections, and more. Additional topics are presented describing trace files, restart capability, and analysis of Application Engine programs. New Application Engine features found in PeopleSoft release 8 are also presented.

The appendices found in the book consist of descriptions of the Problem Tracking application (appendix A), Locations by Operator Class application (appendix B), a listing of PeopleSoft system tables (appendix C), Application Engine examples (appendix D), a list of commonly used built-in PeopleCode functions (appendix E), and a list of Application Engine functions (appendix F).

CODE DOWNLOAD

All source code presented in this book is available from the Manning website. The URL www.manning.com/delia includes a link to the source code files.

about the authors

TONY DELIA has over fifteen years' experience working with mainframe, client/server, and relational database applications, including PeopleSoft HR, Payroll, and Financial applications. He specializes in custom application development. Tony enjoys roller hockey, weight lifting, and most other physical activities. He seldom travels far without a sketchbook and crayons. Some of his artwork and technical creations can be seen on his website http://www.sqrtools.com.

GALINA LANDRES has been working in the field of computer science for more than twenty years. Galina has been involved in the development and customization of PeopleSoft applications since PeopleSoft's release 3.0. She is a co-author of *SQR in PeopleSoft and Other Applications* (Greenwich, CT: Manning Publications, 1999). Galina is a founder of SQRLand (www.sqrland.com), a consulting company specializing in PeopleSoft, SQR, and relational database applications.

ISIDOR RIVERA has been in the field of software development for twenty years. His background includes Mainframe and Client/Server applications. He has worked on systems in areas such as Financial Modeling, Accounting, and more recently, Human Resources and Payroll applications. Isidor has much experience converting legacy data to PeopleSoft for distinct business units of a major corporation and has in-depth knowledge of the globalization of PeopleSoft applications.

PRAKASH SANKARAN has been working with client/server applications for the past twelve years. During that time, he has been involved in implementing PeopleSoft applications for the past ten years. He has been working with the PeopleSoft application since release 1. Prakash has extensive experience in converting legacy systems to PeopleSoft as well as upgrading existing PeopleSoft applications to newer releases. Some clients which contributed to Prakash's development and growth in the PeopleSoft field are the International Monetary Fund, Wakefern Food Corporation, Best Foods, St. Francis and Bristol Hospitals, Seagram, and SPX Corporation.

The authors share several common threads. Besides having a considerable amount of PeopleSoft experience, each of them has consistently exhibited a willingness to share this experience with others. It is this spirit of helping others which has served as the motivating factor in producing this book.

acknowledgments

Many people deserve special recognition for their part in the making of this book. First and foremost, our appreciation goes to the people at Manning Publications who made this book a reality. Not only were they professional and supportive, but very patient and understanding as well. Our special thanks goes to Marjan Bace, Ted Kennedy, Mary Piergies, Adrianne Harun, Dottie Marsico, Leslie Haimes, and Sharon Mullins.

We would like to thank the following people who participated in the technical review of this book: Ahmet Emre, Peter Choi, Andrew Gatti, Buddy MacDonald, Cary Cloud, Celia Hyman, Cindy Finnigan, David and Lisa Hill, David Hardacker, Del Iglesia, Doug Cha, JR Growney, Peter Choi, Richard Reid, Steve Britt, and Steve Gill

A special note of gratitude goes to Chris Heller, the Director of PeopleTools Product Strategy, for supplying release 8.0 information and also reviewing much of our material. His contributions to this book have been greatly appreciated.

Tony DeLia would like to thank his family for their support and acceptance of occasional neglect. Additionally he'd like to thank his wife Tanya, who has been an inspiration and a beacon, carefully guiding the direction of his career. She has also given him the greatest gift imaginable, his daughter Katie. His dog Devon deserves some praise for quietly waiting to be let out while Tony finished some of these chapters. Tony would also like to thank Galina, Isidor, and Prakash for the opportunity to be a part of this book.

Galina Landres would like to thank her husband Vlad for his enormous help in the review process. This whole project was his idea and Vlad helped tremendously to bring it to life. Many thanks go to her son Gene and her daughter Inna for their love and support as well as their help in the book's creation. Galina thanks her dear parents for being very understanding and patient. Special thanks go to Irina, Arkady, Ester, and Leon for their continuous love and support. Many thanks to the entire team at Seagrams (her best and favorite client). Huge appreciation goes to her fellow co-authors Tony, Isidor, and Prakash for their excellent work and great friendship.

Isidor Rivera writes: This is for the memory of my father Isidro, Sr. Your illness and subsequent passing in the spring of '99 was very unexpected. We had so many plans early last year, just as this book project was beginning. In the months following your passing, it became so difficult to come home and work on this project. The weekends and late evenings were spent thinking about all the wonderful things you did for Sonia and me. How we marvel at your work and miss

you deeply. Thanks to the co-authors, Galina, Tony, and Prakash for helping me find the strength and will to continue.

Prakash would like also to thank his book colleagues, Galina, Tony, and Isidor for putting up with his work schedule and his endless (not anymore!) delays with his part of the book. He also thanks his primary clients in the past five years—Wakefern Food Corporation, Best Foods, St. Francis and Bristol Hospitals, Seagram, and SPX—for giving him an opportunity to acquire the experience he needed to write this book. Finally, he would like to thank his father for always supporting him in whatever he has done in his life.

about the cover illustration

The cover illustration of this book is from the 1805 edition of Sylvain Maréchal's four-volume compendium of regional dress customs. This book was first published in Paris in 1788, one year before the French Revolution. Its title alone required no fewer than 30 words:

> *Costumes Civils actuels de tous les peuples connus dessinés d'après nature gravés et coloriés, accompagnés d'une notice historique sur leurs coutumes, moeurs, religions, etc., etc., redigés par M. Sylvain Maréchal*

The four volumes include an annotation on the illustrations: "gravé à la manière noire par Mixelle d'après Desrais et colorié." Clearly, the engraver and illustrator deserved no more than to be listed by their last names—after all they were mere technicians. The workers who colored each illustration by hand remain nameless.

The colorful variety of this collection reminds us vividly of how culturally apart the world's towns and regions were just 200 years ago. Dress codes have changed everywhere and the diversity by region, so rich at the time, has faded away. It is now hard to tell the inhabitant of one continent from another. Perhaps we have traded cultural diversity for a more varied personal life—certainly a more varied and exciting technological environment. At a time when it is hard to tell one computer book from another, Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional life of two centuries ago, brought back to life by Maréchal's pictures. Just think, Maréchal's was a world so different from ours people would take the time to read a book title 30 words long.

PART 1

An introduction to PeopleSoft and PeopleTools

PeopleSoft has been very successful in the ERP marketplace for many years. An obvious reason for this has been PeopleSoft's ability to provide solid packaged solutions for a wide range of business functions. Equally important is PeopleSoft's commitment to incorporating the latest advances in technology into its software. Each release of PeopleSoft has kept stride with the best that technology has to offer. For instance, PeopleSoft's architecture has evolved from a traditional two-tier to three-tier in release 7.0 to web-based architecture in the current 7.5 release and in the not-too-distant future, *n*-tier architecture will arrive in the upcoming 8.0 release. In addition, PeopleSoft provides an extensive toolset called PeopleTools, which allows customers to easily modify existing applications or develop new ones. Third-party software such as SQR and Crystal Reports are bundled into the PeopleSoft package for increased functionality.

In the pages ahead we take a look at the evolution of client/server architecture. We consider its components and how client/server architecture specifically relates to PeopleSoft. We then perform a general walkthrough of the PeopleSoft software, discussing some of its basic capabilities. You'll find a description of Configuration Manager and learn how it is used to control your PeopleSoft environment. An overview of many tools used within PeopleSoft—including Application Designer, Data Mover, Security Administrator, and Tree Manager are also presented. As you proceed through the chapters, you'll see that PeopleTools is very robust yet easy to use—and, yes, fun!

CHAPTER 1

PeopleSoft fundamentals

PeopleSoft has evolved from the traditional client/server architecture to the multi-tier architecture in PeopleSoft 7.0. In this chapter we will discuss the components that are part of this evolution and describe the functions of the individual tiers that form this architecture.

1.1 PEOPLESOFT ARCHITECTURE

1.1.1 Two-tier architecture

Traditional client/server installations are defined as two-tier architecture, which means that two components exist in a two-tier structure, Client and Server. Client refers to the workstation used to access the application; Server refers to either a database server or some other type application server. In PeopleSoft, Server, in client/server architecture, always implies the database server which hosts the application database.

The following illustration can help us understand the two components and how they communicate between each other.

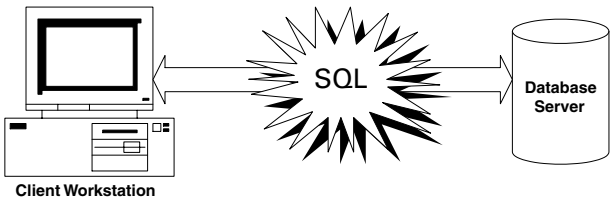


Figure 1.1 Two-tier architecture

The client workstation converts the client request into SQL statements and communicates to the database server using database connectivity tools. The client workstation processes all user requests and transmits them across the network to the database server. Some advantages of a two-tier architecture are as follows:

- simple architecture
- easier administration
- cost reduction

A two-tier architecture is ideal when the client is connected to the server on a local area network, but SQL transmissions are voluminous and, as a result, efficient transmissions are not possible across the wide area network. PeopleSoft and other client/server software applications wanted to overcome this challenge and make it possible for client workstations across wide area networks to have faster access to the application.

1.1.2 Three-tier architecture

Citrix Systems introduced an application server, which runs processes that would otherwise run on a client workstation. Users log in to the application server and sessions are run on the remote application server. Users transmit keystrokes and mouse-clicks to the application server which then transmits images back to the client workstation.

The application server acts as the third tier. The key advantage in configuring a third tier is that it is physically located near the database server. Communication between the database server and the application server is within a local area network or better. Data transmitted between the client and the application server is less voluminous than sending SQL requests across the wide area network directly to the database server. This concept reduces the size of data transmitted on the wide area network. Application servers provide a central point of administration as well. They are usually configured to have more processing power and memory. Multiple application servers can be configured to share loads from numerous clients accessing applications across the wide area network. Citrix servers are physical application servers, and there is always a cost involved in maintaining hardware.

PeopleSoft joined with BEA Systems to introduce a transaction-based application server called Tuxedo. Tuxedo application server is a collection of server processes that communicate to the database server. On the server side, workstation listeners are listening to client Tuxedo requests and sending them to the appropriate server process. These server processes request individual services, which can handle jobs such as SQL calls, panel group build, panel group save, and so forth.

Some advantages in a physical three-tier architecture include:

- remote session capabilities near the database server
- reduction of network traffic by transmitting only keystrokes, mouse-clicks, and images across the wide area network
- single point of administration and monitoring per application server
- single point of installation per application server
- load balancing using multiple application servers

Some advantages in a Tuxedo-based three-tier architecture include:

- the ability to transmit more requests using Tuxedo services than using SQL on a network
- the ability to process Tuxedo requests close to where data resides
- the ability to reduce bottleneck in the database server because Tuxedo requests are queued in the application server and transmitted to the database using Tuxedo services as they become available
- the ability to achieve a minimum installation of clientside software, thereby resulting in thin clients
- load balancing by installing many application servers that process data requests from clients
- the ability to encrypt data transmitted from the database server to the client
- the scalability of Tuxedo application servers, which support a wide range of operating systems, databases, and hardware platforms
- the ability to install application software in a single server, thus resulting in easier software maintenance and upgrades

In both Citrix and Tuxedo implementations, more than one application server can be installed depending on the number of clients accessing the database. Some implementations use Tuxedo clients on Citrix Metaframe systems which take advantage of both systems. Users access the three-tier client software on Citrix application servers using remote sessions. PeopleSoft application software is installed/replicated on one or more Citrix application servers.

Note, that a Tuxedo application server can either be a logical or physical configuration.

Logical Application Server In a logical application server configuration, Tuxedo application software is installed on the same physical machine as the database server. In this case, only two physical machines exist in the whole configuration: the client workstation and the database server, which hosts both the database and Tuxedo.

Physical Application Server In a physical application server configuration, Tuxedo application software is installed on a separate physical machine, one different from the database server. In this case, three physical machines exist in the configuration: the client workstation, the application server, and the database server. The physical application server and the database server are either connected on the same network backbone or within the local area network.

The illustrations in figures 1.2 and 1.3 can help us understand both the physical and logical three-tier architectures in PeopleSoft.

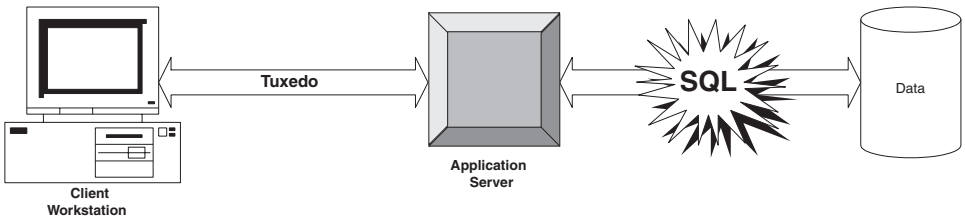


Figure 1.2 Physical three-tier architecture

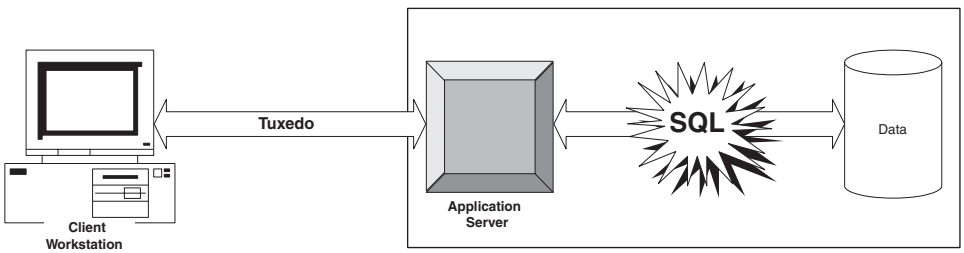


Figure 1.3 Logical three-tier architecture

Notice that the only difference between these three-tier architectures is the number of physical machines configured.

With the arrival of electronic commerce and the recognition of the advantages of accessing applications on the Internet, the next step toward expanding the PeopleSoft architecture was to create a web client.

1.1.3 Web architecture

PeopleSoft introduced a web client which can access a web server that hosts PeopleSoft HTML files and Java applets. The web server communicates with the application server using a BEA System product, called Jolt, that supports web client connections. Jolt interprets HTML and Java applet requests from the web server to the application server. Jolt acts as the translator of Java and HTML codes into C++ codes.

The web server architecture can be either a physical or logical architecture. In a physical architecture installation, Jolt Internet Relay (JRLY) and Jolt Relay Adapter (JRAD) software are required. JRLY and JRAD are products supplied by BEA Systems to secure transactions transmitted on an Internet or Intranet connection. JRLY is installed on the web server placed outside the firewall, and JRAD is installed on the same machine as the application server. Figure 1.4 illustrates the web architecture in a PeopleSoft 7.5 installation.

In figure 1.4, we notice that the web server is an additional tier to a logical three-tier architecture. The web client uses a web browser to access the application. Java applets and HTML files loaded on the web server are used to access the application. The web server connects to the application server with the help of Jolt.

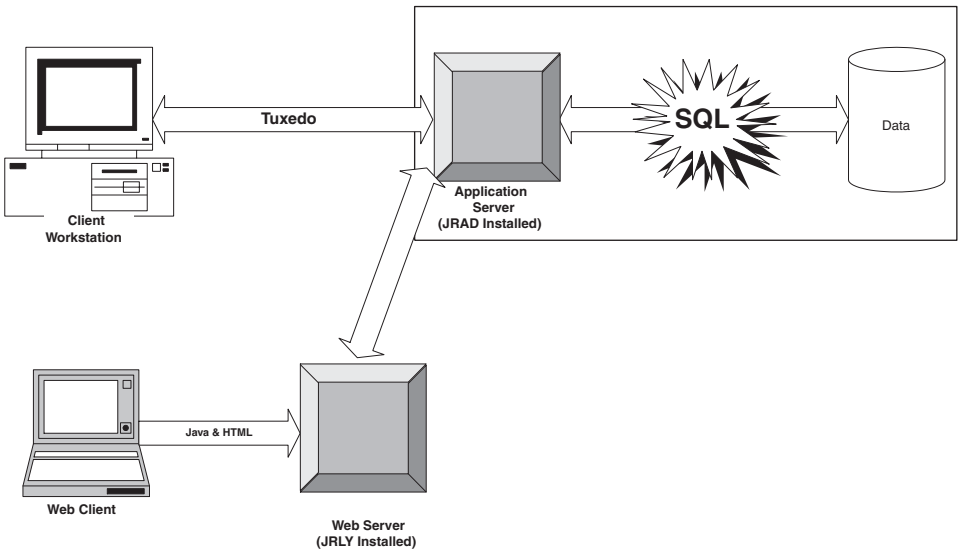


Figure 1.4 Web architecture

1.1.4 *n*-tier architecture

So far, we have discussed the evolution of PeopleSoft architecture in the previous sections. We should realize by now endless possibilities exist for expanding this architecture to service various types of clients. PeopleSoft 7.5 serviced Windows clients, three-tier clients and web clients. PeopleSoft 8 will be released with the deployment of a number of tiers which service many types of clients. The number of tiers that can be deployed is expandable, hence it is called the *n*-tier Architecture.

PeopleSoft 8 also introduces the Internet client which can access the PeopleSoft application using the HTTP protocol via a web browser. PeopleSoft 8 will still support the Windows client and the three-tier client. Additional features such as the Directory Server, Application Messaging and PSWebDeploy will be available with PeopleSoft 8. Directory Server provides a single point of user ID and password administration for users logging into multiple databases. Integration with third party applications using XML/HTTP based messaging will be available using Application Messaging and Publish/Subscribe concepts. PSWebDeploy helps a large number of users access the PeopleSoft application with minimum installation required on their workstations. The client workstation is installed with an executable called PSLaunch that accesses the PeopleSoft application server and downloads the components required to access a PeopleSoft application panel. The PSLaunch software can be launched using a link from a web browser. PSLaunch software can also be launched from an email attachment or from a file stored on the user workstation.

The *N*-tier architecture is advantageous in that it:

- introduces the Internet client that can access the PeopleSoft application using a web browser
- offers minimum workstation installation and thin client
- reduces network traffic by accessing the application using Tuxedo requests that are smaller than an SQL request
- deploys and adds components to this architecture to service more types of clients than before

Figure 1.5 provides an insight into the *n*-tier architecture, illustrating the types of clients that this architecture can service.

As you can see in figure 1.5, multiple application servers and web servers are deployed accessing the same database. In the same installation, Windows clients, Tuxedo clients, and Internet clients are deployed. Windows clients can access the PeopleSoft application using traditional two-tier access.

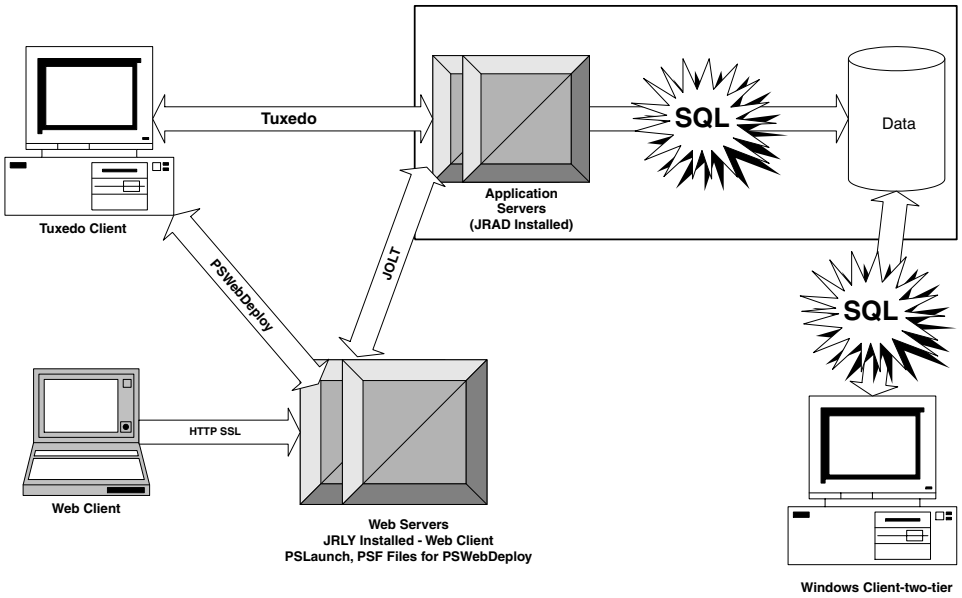


Figure 1.5 *n*-tier architecture

1.1.5 Tiers and their functions

The following important tiers are pertinent only to the PeopleSoft architecture.

Client A client is the tier which facilitates the user to access the server. A client can be a user workstation which runs a Windows 95 or Windows NT operating system. The client can access PeopleSoft using a traditional two-tier connection using SQL, a three-tier connection using Tuxedo, or a four-tier connection using a web browser.

Database server The database server hosts the PeopleSoft application database. The application can be hosted on a variety of relational database platforms such as Oracle, Microsoft SQL Server, Sybase, DB2, and so forth. The database server can also be hosted on a variety of operating systems such as Windows NT, Unix, MVS, and so forth. In addition, the database server hosts the SQL connectivity software which communicates with the client and the application server.

Application server The application server, an intermediate tier which connects the client, the web server, and the database server, can be run on a Unix or Windows NT operating system. The application server runs the Tuxedo and Jolt middle-ware applications and can either be a logical server—which resides on the same machine as the database server—or a physical server—which resides on a different machine than the database server. The application server can also be a Citrix server which runs remote

client sessions. Some installations deploy both Citrix servers and Tuxedo application servers which help create remote client access for tools and database access for data.

Web server Web servers service web clients which access the PeopleSoft application using a web browser. The web server contains HTML and Java applets which the web clients access using a web browser. The web server can either be a physical or logical server. When the web server is installed on a separate physical machine, it also contains the Jolt Internet Relay, used for communicating to the application server.

1.2 A USER'S VIEW TO PEOPLESOFT

PeopleSoft Applications offer an array of functionality, tools, and reporting features. The current technology is based on relational database and client/server architecture. The back end or database can reside on many platforms. Platform/database combinations such as UNIX/Oracle, MVS/DB2, Windows/SQLBase, and NT/SQL Server are just some of the many platforms and databases supported by PeopleSoft Applications.

As the new millennium progresses, the move toward web-based technology is becoming ever more pronounced. PeopleSoft e-Business and Enterprise Performance Management applications are also gaining momentum using web technology.

1.2.1 Signing onto PeopleSoft

Our journey into PeopleSoft begins with a sign-on into the application. The example in figure 1.6 illustrates the PeopleSoft sign-on panel for a two-tier connection. Figure 1.7 illustrates a three-tier sign-on.

The sign-on panel identifies several items:

Connection type PeopleSoft applications support connectivity on both two-tier and three-tier client/server configurations. When a client signs onto the application using a two-tier connection, the client connects directly to the database server. In a

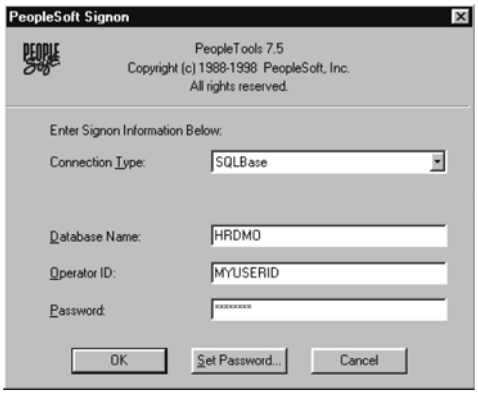


Figure 1.6
PeopleSoft sign-on (two-tier)

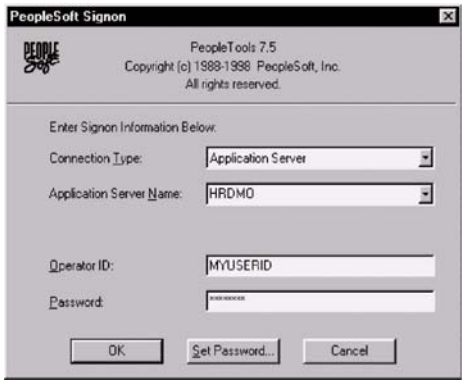


Figure 1.7
PeopleSoft sign-on (three-tier)

three-tier connection, the client is connected to an application server that maintains connections to the database server

Database name The database name is simply the name of the database to which we are connecting.

Operator ID The operator ID is the ID used to enter the PeopleSoft application. The ID is generally set up by the security administrator. PeopleSoft IDs are linked to an operator class which has specific functionality privileges allocated to it. In addition to English, the operator ID can have other languages such as Dutch, French, German, Japanese, Portuguese, and Spanish linked to it. Two users who have unique IDs but are linked to the same operator class can view the same panels in their own language.

Password The password is initially established by the security administrator, but can subsequently be changed by the end user. This can be accomplished by clicking the Set Password button. Some items on the sign-on panel do not have to be keyed in every time we logon. Unless we are transitioning from one platform and/or database type to another, or we support applications on varying platforms and databases, the connection type parameter can be set one time. The database name, application server name, and ID can also be set. As a result, these parameters do not have to be entered at each login. The Startup tab of the Configuration Manager panel enables defaults and overrides during the PeopleSoft login process. An example of the Configuration Manager panel is shown in figure 1.8.

1.2.2 Configuration Manager

The Configuration Manager (figure 1.8) enables PeopleSoft settings to be administered from a central site. These settings, however, are based on how the application is implemented. Registry settings impact a workstation setup and can be shared by an entire group. Therefore, a change to one setting may impact all users in the group. This is common in a Citrix environment or when users share a common file server

which contains the executables and runtime Dynamic Link Libraries (DLLs). The Configuration Manager contains tab settings which can be used to tailor specific environmental conditions. Configuration Manager can be entered using the Edit → Preferences →Configuration menu navigation from either the various applications or from a PeopleTool such as Utilities, Process Scheduler, Application Engine, Mass Change, or Translate.

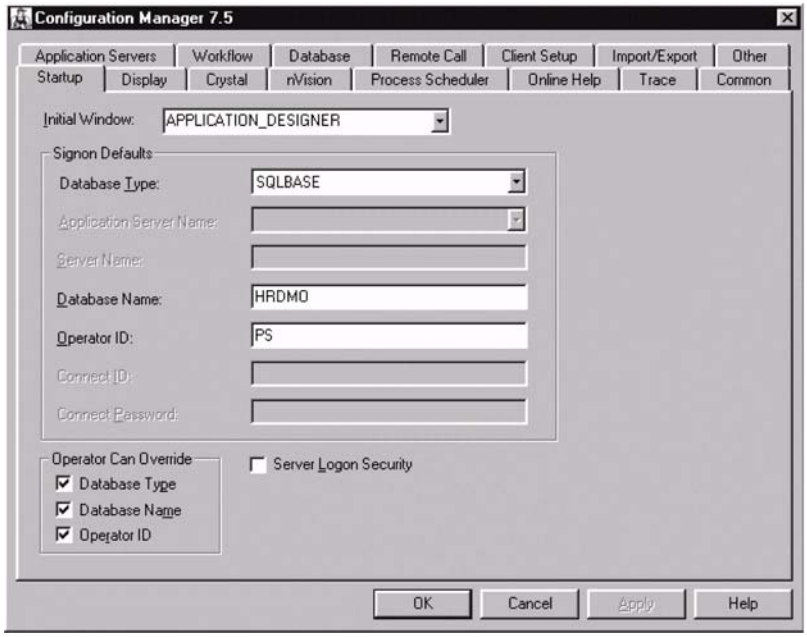


Figure 1.8 Configuration Manager

Startup The Startup tab allows for the entry of default values for database type, database name, and operator ID. In the startup tab, additional parameters exist which can be defaulted but they are more platform-specific. These parameters include the current ID/password for DB2 and Informix. MS SQL Server, and Oracle platforms do not use these options. The lower left portion of the panel enables us to override the database type, name, and operator ID. These parameters can then be modified during sign-on.

Display The Display tab enables the modification of the PeopleSoft application panels to be adjusted, based on desktop needs. These modifications include changes to panel height and width.

The Navigator display options can be set so that the navigator is displayed once during PeopleSoft startup each time a menu group is opened or not displayed.

Crystal The Crystal tab provides the Crystal executable path and default location of Crystal Reports. Additional Crystal options include using trace during execution as well as the subsequent logging to a trace file.

nVision The *n*Vision settings are linked to PeopleSoft queries which are sent to an Excel spreadsheet. The number of blank Excel columns between output data on a spreadsheet can be specified. If no blank separator columns are required, the "Space Between Query Columns" parameter can be set to zero.

Process Scheduler The Process Scheduler settings enable us to specify the directory search path for SQR programs and COBOL executables. This tab setting also identifies any PeopleTools and MS Word executable directory.

Online help Any online documentation associated with Windows help or PeopleBooks can be defined based on function keys and PeopleBooks search order.

Trace Several types of settings can be used during an online session. Trace can include PeopleCode trace, SQL trace, and message agent trace. The default online trace file is DBG1.tmp, which can be overridden by specifying an online trace file.

Common This tab specifies the language setting used on panels and related objects. The Cache file directory can also be specified on this tab. Data Mover, which is a database administration tool used to migrate application data and objects, requires an input, output, and log file. The directories for Data Mover files can be specified on this tab.

Application servers Any configured application server to which a client is allowed to connect to can be specified on this tab. Additional parameters are Application Server Name, Machine Name or IP Address, Port Number, Tuxedo Connect String. The Set and Delete buttons enable the entry and removal of Application Server Names.

Workflow Under the Workflow tab you specify the options and locations related to the Workflow implementation at your site.

PeopleSoft Workflow allows tasks to be automated into flexible business processes. From a technical perspective, the options required to use Workflow can be identified on this tab. Some items, which can be specified on this tab, include Message Agent, Forms, and Mail Protocol.

Database Databases such as DB2 and Sybase may require additional settings that can be used to improve or monitor the system operations. Some parameters include DB2 message size, Sybase packet size, and Application Designer image conversion which enables the conversion of images to a new format during upgrades.

Remote Call The options on the Remote Call tab are related to the Tuxedo Remote Call. Transactions that require intensive memory and CPU resources can be run on a remote server. The parameters include the timeout, the debugging options, and the appearance on the desktop of a child COBOL process.

Client Setup The Client Setup tab identifies the options which impact workstations as well as invoke the Client Setup process. The settings include Shortcut Links, 3-Tier Minimal Install, and ODBC Setup. When checked, the Install Workstation option connects the Client Setup function.

Import/Export The environment settings established can be exported to or imported from a file using the information specified on the Import/Export tab.

Other These settings impact the environment used to run the PeopleSoft quality server for manufacturing applications. These settings require two parameters, which identify the local data directory and SQR Output.

1.2.3 Navigation in PeopleSoft

Once you logon to PeopleSoft, the panels you see displayed depend on several factors:

- PeopleSoft products installed on your system
- the initial window default in the Configuration Manager
- the security profile of your operator ID, which allows you to view only certain panels

For example, a typical Human Resources user would have access to the Administer Workforce panels, illustrated in figure 1.9.

The panels and menus used in the delivered PeopleSoft applications can be easily navigated with proper knowledge of function keys and toolbar icons. Custom applications should also use the same convention as standard PeopleSoft applications. Throughout this book, you'll see figures which reference two small custom applications used to present the topics discussed. One application is used for problem tracking and the other links operator classes to office locations.

Let's discuss the objects found on a PeopleSoft application panel.

Menus are used to group functionally-related panels and panel groups. A typical menu in the HRMS system is illustrated in figure 1.10. The menu identifies the bar items contained in the menu. The menu navigation required to display the panel group illustrated in figure 1.9 can be written as

Navigation: Go →Administer Workforce (GBL) →Use →Hire

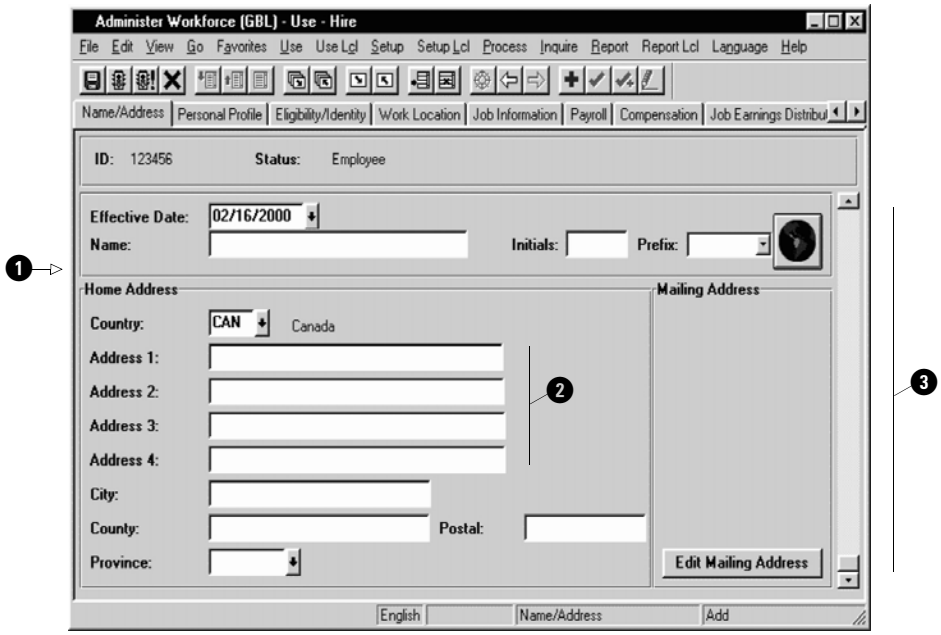


Figure 1.9 Administer Workforce panels

1. Panel
2. Record Fields
3. Scroll Bar

Navigation: Administer Workforce (GBL) →Use →Hire

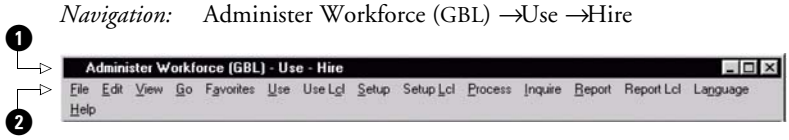


Figure 1.10 Menu portion of toolbar

1. Menu Label
2. Menu Bar Label

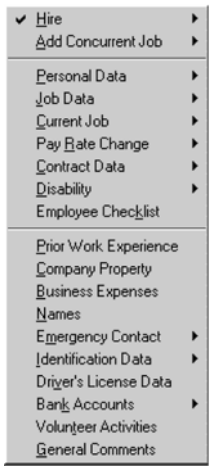


Figure 1.11 Menu items under the Use menu bar label

After the menu is displayed, each menu bar item may contain one or more menu items to which the operator has access. The menu items associated with the Use menu bar label are shown in figure 1.11.

Features, such as the toolbar, are common to PeopleSoft Applications. Specific panel functionality allows us to use the toolbar for saving data, submitting a process or canceling out of a panel. When a list is present, several list toolbar buttons can be used to display a list or move up and down the list. Additional toolbar buttons can be used to navigate from one panel to another or to insert/delete rows from a scroll bar. Figure 1.12 illustrates a standard toolbar available with most PeopleSoft applications using release 7.xx.

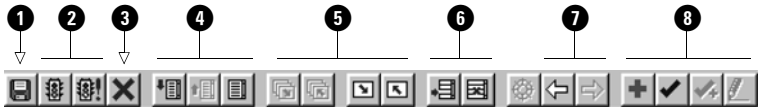


Figure 1.12 Typical PeopleSoft application toolbar

1. Save button
2. Run, Run with Defaults
3. This cancels a panel
4. Next in List, Previous in List, List
5. Next Panel in Group, Previous Panel in Group, Next Panel, Previous Panel
6. Insert Row (F7) Delete Row (F8)
7. Change Window content, Back ⇦ Forward ⇨
8. Add, Update/Display, Update/Display All, Correction

If you are familiar with Windows applications you know that the Save button is enabled after a change has been made to a panel. A save operation on a panel or panel group can also be performed by pressing the ENTER key.

Run and Run with Defaults are used to submit processes to the Process Scheduler. The Process Scheduler can submit a process such as an Application Engine, an SQR, or a COBOL program to be run on a client workstation or server.

The Cancel button is used to cancel activity on a panel.

Next in List, Previous in List, and List are activated when a partial key value is entered into a search dialog and the results of the search return more than one record. These buttons enable a list to be displayed and navigated upward or downward.

Next Panel in Group, Previous Panel in Group, Next Panel, and Previous Panel allow for movement between panels or panel groups.

Records which contain effective-dated rows or are part of a parent/child key hierarchy can be presented on panels that utilize scroll bars. The use of scroll bars enables us to work effectively with multiple record hierarchies. At the same time, however, scroll bars present unique challenges to both the developer and end user. (In subsequent chapters throughout this book, scroll bars will be discussed and explained in more detail.) Figure 1.9 is an example of a panel that contains one scroll bar. A panel with scroll bars may use the Insert Row and Delete Row icons. These icons are used to insert or delete effective-dated rows. Function key F7 can be used to insert data, while F8 deletes data.

The Change Window Content button is used to toggle on and off the business process maps. The Back and Forward buttons are used when navigation display is used with menus.

Additional buttons such as Add, Update/Display, Update/Display All, and Correction are used when data consisting of new keys are added or when data are updated.

Figure 1.13 Tabs in a panel group

Panel tab labels identify the panels in a panel group and are used to move from one panel to another. An example of panel tab labels is shown in figure 1.13.

KEY POINTS

1. PeopleSoft has evolved from a traditional two-tier client server architecture in the earlier years, to a web based *n*-tier architecture in PeopleSoft.
2. PeopleSoft *n*-tier architecture enables various types users to access PeopleSoft applications. Application owners, end users, employees, and vendors can access PeopleSoft applications using various architectures and tools.
3. PeopleSoft provides applications that are robust, full of tools, and functionality.

KEY POINTS (CONTINUED)

4. The PeopleSoft sign-on process requires a connection type, database name or application server, operator ID, and password. These options can be set once using the Configuration Manager.
5. Menus, toolbars, and panels vary, based on the functionality assigned to an operator class.
6. Toolbar buttons as well as menu actions and function keys enable the end-user to save, cancel, insert, or delete data on a panel.

CHAPTER 2

Development tools

In this chapter, we cover some basic concepts surrounding the primary development tools used in PeopleSoft. The Application Designer is a conglomeration of the primary development tools that design a PeopleSoft application. Prior to release 7 of PeopleSoft, objects were built using individual tool menus for each object. For example PeopleSoft 6 has Data Designer to develop record definitions, translate values, and so on; Panel Designer was used to build panels; and Menu Designer was used to build menus. Starting from release 7, however, PeopleSoft integrated all the development tools into an integrated menu, the Application Designer.

2.1 FIELDS

Fields are at the lowest level in the totem pole of objects in PeopleSoft. Fields are individual objects defined in a PeopleSoft system. One or more fields are assembled to form a record definition, and fields can be shared across record definitions. In other words, the same field can be used in more than one record definition. The basic attribute of a field is the same across all these record definitions. For example, a field defined as character field is always a character field across all record definitions.

The following field attributes are shared across all record definitions in the system.

- field type
- field length
- field decimal places
- translate values (optional)
- long and short names
- field formats

The field type, length, decimals (if any), long names, short names and field formats are the same across all record definitions. Translate values, although attached to a field, can be used optionally in record definitions. Fields on record definitions can be defined not to use the translate values attached to fields.

Field types allowed in PeopleSoft include the following:

- *Character* fields are usually used for codes in PeopleSoft.
- *Long Character* fields are used to store comments in PeopleSoft.
- *Number* fields can hold positive integers and decimal numbers.
- *Signed Number* fields can hold negative integers and decimals.
- *Date* fields can hold dates and is always represented in a "MM/DD/YYYY" format online regardless of how they are stored in the database.
- *Time* fields can hold time and are represented in a "HH:MI:SS.999999" format. Seconds and subseconds are optional and can be suppressed.
- *DateTime* fields can hold both date and time in one field. They are represented in the same format as date and time fields put together. They can be used to store date and time stamps in PeopleSoft.
- *Image* fields are used to store pictures in PeopleSoft. Some formats in which images can be stored are bitmaps, JPEG, Postscript, and so on. Employee photos are stored using Image fields in a PeopleSoft HRMS system.

Fields attributes are stored in a database table called PSDBFIELD, the catalog table, which is populated as fields are created and changed in a PeopleSoft system. All field definitions in a PeopleSoft system can be listed from this catalog table. During an application upgrade, this catalog table is used to compare and list out differences in fields across databases.

PSDBFIELDLANG This language-related catalog table stores field long and short descriptions in alternate languages. The long and short descriptions can be used when a user, who has a default language other than English, accesses the field online.

When fields are used in record definitions, they are stored in a catalog table called PSRECFIELD. By querying this table we can find out all the records that use a certain field in the system. (See chapter 4 for more about creating fields in PeopleSoft.)

2.2 RECORDS

A record definition is a collection of fields. A record definition in PeopleSoft can be an SQL table, an SQL view, a Sub record, a Derived/Work record, a query view or a dynamic view. SQL tables and SQL views also exist in the database. Other types of record definitions are stored only in the PeopleSoft system.

Record definitions may possess a variety of attributes, but they can be categorized into three properties. The list of properties for a record definition include General properties, Use properties, and Type properties.

General properties contains a description of the record definition, the last date and time the record definition was updated, and the ID of the operator who updated it. Use properties defines the key fields, search fields, list fields, query security record, related language record, parent record, and the audit properties for the record definition. Type properties defines the type for the record definition. We define whether the record definition is an SQL table, an SQL view, and so on. under the Type properties. Let us look at some of the catalog tables that store properties of a record definition.

PSRECDEFN Record definitions in PeopleSoft are stored in a catalog table called PSRECDEFN. This table stores all the primary attributes for a record definition. It holds attributes such as record type, audit record name, related language record, parent record name, query security record, index count, field count, and others. This table also holds the table space name for record definitions defined as an SQL table. All record definitions that are SQL tables or views are stored with a prefix of 'PS_'; all PeopleTools record definitions are defined with a non-standard SQL table name. The SQLTABLENAME field is an override to the 'PS_' prefix. In the database, the table name is stored with the value entered in the SQLTABLENAME field without the 'PS_' prefix.

PSRECFIELD The PSRECFIELD table stores the fields that the record definition contains. Each field in the record can have its own edit properties. It can either have translate value edits, prompt table edits, or a Yes/No edit. These attributes are specific to the record definition field. Each field can also hold default values and PeopleCode events specific to the record definition. A field in this catalog table, called PROGCOUNT, contains the number of PeopleCode events for the record field. If this program count does not match the actual number of PeopleCode events for a given field, the record definition cannot be opened.

PSINDEXDEFN The PSINDEXDEFN table contains a row for each index for the record definition. This table is populated only if the record definition is an SQL table or an SQL view.

PSKEYDEFN The PSKEYDEFN table contains all the fields that the PSINDEXDEFN holds. All the record fields that compose the index are stored with the key sequence.

PSRECDDLPARM The PSRECDDLPARM table has all the DDL parameters for the record definition. This is stored only if the record definition is an SQL table or an SQL view.

As you can see, record definitions serve as building blocks with fields assembled in them. It is important to remember that record definitions defined as SQL tables or views are also stored in the back end database. Other types of record definitions are stored only in PeopleSoft and are not database objects. Record definitions that are database objects have to be built in the database as well. Chapter 8 describes the process of building database objects from PeopleSoft record definitions.

PSRECDEFNLANG This language-related catalog table stores long and short descriptions of record definitions. When a user has a default language other than English, the user can see record descriptions in his/her own default language. Developers can login to the system using alternate languages and enter descriptions in that language.

Other types of record definitions are Derived/Work records, query views, dynamic views, and subrecords. The use of Derived/Work records and subrecords are explained in chapter 6.

2.3 PANELS

Panels serve as user interface to the application. Panels are built using field and record definitions. Panels are a collection of record fields adhering to certain rules. Panels vary from simple panels, panels with scroll bars, panels with subpanels and secondary panels, and panel groups.

Let us look at all the panel field types used in PeopleSoft:

- *Frames* group fields logically. For example, fields used to hold an address can be grouped into a frame for clarity.
- *Group boxes* are similar to frames in the sense that they are also used to group fields. The difference is that group boxes are used to group the same field with translate values. Group boxes group the same record field with radio buttons representing different translate values. An example of a panel with group boxes is the PAYROLL_DATA2 panel in a PeopleSoft HRMS system.
- *Static Text* fields are used to hold free form text.
- *Static Image* fields store images that remain static and do not change at any point of time.
- *Checkboxes* store fields that either accept 'Yes' or 'No' for input. They can also be used to store fields that contain only two translate values. They can be turned on or off to store two different values in them.

- *Dropdown list boxes* are used for any field that has a prompt record. By clicking on the drop-down button, the user can invoke the prompt to produce a list of valid values.
- *Edit boxes* are very similar to drop-down list boxes except that they do not have a prompt list. Edit box is the most common panel field type in PeopleSoft. When a field that has a prompt record is defined as an edit box, the panel designer automatically changes the field into a drop down list box. The prompt button can be hidden using the Panel Field Properties screen.
- *Images* stores images in the database. Unlike static image fields, image fields can hold dynamic images. Images can be inserted into the panel field by either choosing the F5 key or by choosing "Edit/Insert Image" from the application menu.
- *Long Edit boxes* are used for long fields that are, in turn, used to enter comments. They can be sized to the desired height and length, and they scroll as more data are input into them.
- *Push Buttons* are used to invoke a command, bring up a secondary panel, or invoke a process. We will learn more about push buttons in chapter 6.
- *Radio Buttons* are fields that can contain a value from the translate table. They are used for record fields which have translate values in them. PeopleSoft has changed most of its radio button fields to drop-down list boxes in release 7. This makes a lot of sense because radio buttons restrict the values that can be entered into an application panel. Since radio buttons can hold only one translate value from a field, a radio button must exist for each translate value. In contrast, one drop-down list box shows all the translate values in a list.
- *Trees* represent data in a tree format. For example, departments can be entered using a tree format in a PeopleSoft HRMS system.
- *Grids* are used instead of scroll bars. They can replace single level scroll bars to represent data in a spreadsheet format. (See chapter 7 for more information relative to Grids.)
- *Scroll Bars* store multiple rows of data into the same record definition. Scroll bars can also be used to store effective dated rows in the system. (See chapter 7 for more information about scroll bars.)
- *Secondary Panels* are used to organize fields in a separate panel. When a panel has optional fields, those fields can be stored using a secondary panel. For example, in the PeopleSoft HRMS system, secondary panels are used to enter mailing addresses for employees. Secondary panels are invoked using push button fields. (See chapter 7 for more information about secondary panels.)
- *Subpanels* are used to organize fields from a subrecord. They are used as an input mechanism for repetitive fields. Addresses in any record definition contain standard fields such as street address, city, county, state, zip code, and country. A subpanel, which contains all these fields from a subrecord, can be built and used in more than one panel.

The following catalog tables are used to store panel definitions in PeopleSoft:

PSPNLDEFN The PSPNLDEFN catalog table stores panel descriptions, field counts, panel types, and grid definitions. Panels are built using the same panel name in different languages. LANGUAGE_CD is part of this catalog table and stores one row for each language, if needed.

PSPNLFIELD The PSPNLFIELD table stores the attributes of all fields in the panels. Some attributes stored in this table include the panel field type, the record name, the field name, the field labels, and so on. This catalog table also stores rows for panel fields in different languages, if necessary. Attributes such as Related Display and Control Display items are also stored in this catalog table. This catalog table has the most number of rows stored in the database among all PeopleSoft catalog tables because, for every panel field, one row exists in this table. Even if the same record field is used in ten different panels, ten rows are stored in the database.

2.4 PANEL GROUPS

Panel groups contain a series of panels that are either organized functionally or contain many record definitions. A single panel group can be used to store data into multiple records. Panel groups are ideal for records that have more than one child record. For example, in the PeopleSoft HRMS system PERSONAL_DATA is used to store personal information for an employee. JOB, EMPLOYMENT, and BEN_PROG_PARTIC are tables that store employment and benefit information for the employee. These tables are placed in a series of panels into a panel group called JOB_DATA.

Individual panels are attached to form a panel group. All the panels in a panel group use the same search record. Also, a search record can be defined when the ADD action is used. Panel groups can also contain work panels that can be hidden. In other words, the hidden panels will be invisible when the panel group is accessed online.

Panel groups were part of the menu object prior to PeopleSoft release 7. Starting from release 7, panel groups are objects themselves. Panel groups can be attached to more than one menu item. Panel groups are stored in their own catalog tables. Panel groups were part of the menu object prior to PeopleSoft release 7. Thus, as stated, starting from release 7, panel groups are objects themselves which means panel groups can be attached to more than one menu item.

PSPNLGRPDEFN The PSPNLGRPDEFN table stores the panel group description, search records, processing location, and market definitions. Markets are used to store the same panel group using different market locations. For example, suppose we want to build different conditional logic for the same panel group for different user regions. The market field makes this possible.

Let's say that we want to use one panel group with two different market definitions for users in United Kingdom and the U.S. We want to have descriptions written using U.K. English conventions or U.S. English conventions depending on the market definition defined in the panel group. In this case, we first build the panel group with 'USA' as the market definition, then clone it by changing the market definition to

GBR. Logic can be built by using the system variable %MARKET%, available during panel processing.

PSPNLGROUP The PSPNLGROUP table stores all the panels contained in the panel group. Each panel can have its own name and label. Item Name uniquely identifies the panel, and Item Label shows in the application menu. Panels can also be hidden within a panel group. All these definitions are stored in this catalog table.

PSPNLGDEFNLANG This catalog table stores language-related descriptions for a panel group.

PSPNLGROUPLANG This catalog table stores panel item names and labels in languages other than English.

2.5 MENUS

Menus serve as gateways to the application. Menus store panel groups that in turn hold application panels which help the user access data from the database. Menu items are individual items which hold a panel group and provide access to an application panel. Users have to be given access to a menu item to access an application panel. Panel level security can be managed using the Security Administrator menu in PeopleSoft.

Menus can be either "standard" or "pop-up" menus. Standard menus, used to create application panels, come with pre-defined menu items. File, Edit, View, Go, Favorites, and Help are the bar items that come predefined in a standard menu. Standard menus are included in a menu group, a collection of like menus grouped by function. When we create a Menu definition, we can specify the menu group for the menu. We can also specify the sequence for the menu within the menu group. We can likewise specify the sequence for the menu group, which can either be sorted numerically or alphabetically.

Pop-up menus are attached to panel fields and function as context sensitive menus. When the user right-clicks on a panel field, pop-up menus are activated. Pop-up menus are defined as a panel field attribute in the panel definition and do not have any predefined menu items. Pop-up menus, useful in bringing up help when the user needs it, can also be used to facilitate look-ups of related information for panels in a standard menu. Panel Fields can be highlighted to indicate the existence of an associated pop-up menu for the panel field by activating the Highlight Pop-up Menu Fields checkbox under the Display tab in the Configuration Manager. The following catalog tables store menu definitions:

PSMENUDEFN This catalog table stores the menu definition. It contains the menu group name, the menu label, the menu sequence, the menu group sequence, and the menu type.

PSMENUITEM This catalog table holds the individual items in a standard or pop-up menu. It contains the panel group, the bar name, the item name, the item label, and the override search record name.

PSMENUDEFNLANG The PSMENUDEFNLANG table stores menu descriptions and labels in languages other than English.

PSMENUITEMLANG The PSMENUITEMLANG table stores menu item names, labels, and bar names in languages other than English.

Menus, when migrated to other databases, are migrated as a whole. Individual menu items cannot be marked for migration. For this reason, when menus are being developed by more than one developer, caution has to be exercised in moving these menus to other databases.

2.6 PEOPLECODE

Activated at different points of panel processing, PeopleCode events are used to control logic during panel processing. Some PeopleCode events are activated before a panel is brought up online; other PeopleCode events are activated during save time. The following PeopeCode events are available in PeopleSoft:

- `FieldDefault` PeopleCode is used to default values into a panel field.
- `FieldEdit` is used to edit values entered into a panel field.
- `FieldChange` is used to perform actions upon entry into a panel field. Other fields can be populated, depending on values entered into a panel field. Functions held in `Field Formula` PeopleCode can be called from `Field Change` PeopleCode.
- `FieldFormula` usually holds functions called from other PeopleCode events. `Field Formula` can also be used to perform logic based on values entered on several fields in the panel.
- `RowInit` is used to initialize fields in a panel before they are displayed.
- `RowInsert` is used to perform logic when rows are inserted on a scroll bar.
- `RowDelete` is used to perform logic when rows are deleted from a scroll bar.
- `RowSelect` PeopleCode is used to drop records on a scroll bar before they are displayed on the panel.
- `SaveEdit` is used to edit fields in the panel at save time. `Save Edit` can also be used to edit several fields in the panel.
- `SavePreChg` is performed before the data are stored in the database. It can be used to change values in panel fields just before they are saved in the database.
- `SavePostChange` is performed after the data are stored in the database. It can be used to insert values into other tables after save time.
- `SearchInit` PeopleCode is used to populate values into fields used as input fields for the application panel. Fields in search records which appear in the input dialog box contain the `Search Init` PeopleCode event.
- `SearchSave` is used to edit values entered in the input dialog box.

- `WorkFlow` events are used to create work flow to other users based on functions performed by the current user.
- `PrePopup` PeopleCode is activated before a panel is brought up on a pop-up menu.

The following catalog tables store PeopleCode events in PeopleSoft:

PSPROGNAME This catalog table stores one row per PeopleCode event. It contains the record name and the field name that holds the PeopleCode event as well as the type for the PeopleCode event.

PSPCMNAME This table stores references to other fields in PeopleCode events.

PSPCMPROG This catalog table stores the actual PeopleCode text. It also holds the count of number of references made to other fields from the PeopleCode event.

2.7 PROJECTS AND UPGRADES

Projects are a collection of objects developed to build an application. Let's say we are developing a time and attendance system. We can include all fields, records, panels, panel groups, menus and PeopleCode used to develop this application. Projects are useful when you want to organize objects and migrate them to other databases. Projects can also be used for change control purposes.

Objects can be inserted into a project by either pressing the F7 key or by choosing "Insert/Objects into Project" from the Application Designer menu. Projects are stored in the following catalog tables:

PSPROJECTDEFN This catalog table stores the project definition and also holds attributes such as commit levels, copy options, target database name, operator ID necessary to sign on to the target database, and report filter options.

PSPROJECTITEM This catalog table stores all the objects included in the project as well as the object type, the action to be taken on the object during upgrade, and the copy status.

PSPROJECTMSG The PSPROJECTMSG table holds error/status messages when projects are upgraded from the source to the target database.

2.7.1 Upgrades

Projects can be upgraded by accessing the Upgrade option from the Tools menu. Objects in projects are pushed from the source database to the target database. Prior to release 7, objects were pulled from the source database into the target database. In order to migrate objects, it was necessary to log into the target database. Starting from release 7, however, objects are pushed from the source database. This makes a whole lot of sense, especially in the implementation stage of a project. Usually, test and production databases exist in every implementation. As soon as development is finished, projects can be pushed to the test database for testing. Once the application is tested, the project can be moved to the production database.

2.8 CROSS-REFERENCE UTILITIES

Utilities, built inside the Application Designer and Utilities menus, are helpful during development in PeopleSoft. Let's take a look at these utilities and consider how we can use them.

2.8.1 Find object references

In the Application Designer, we can find where a particular object is being used by choosing "Edit/Find Object References" from the Application Designer menu. The object should be open first in order to find its object references. The result is then shown in the output window in the bottom of the Application Designer screen. Double-clicking on any of the resulting reference objects will open that particular object in the screen (figure 2.1).

Notice how the five objects which refer to MY_APPLICATION_ID field are shown on the output window in the bottom of the screen. We can find references for any object in the Application Designer by following the same procedure.

Navigation: Open →Field →MY_APPLICATION_ID →Edit →Find Object References

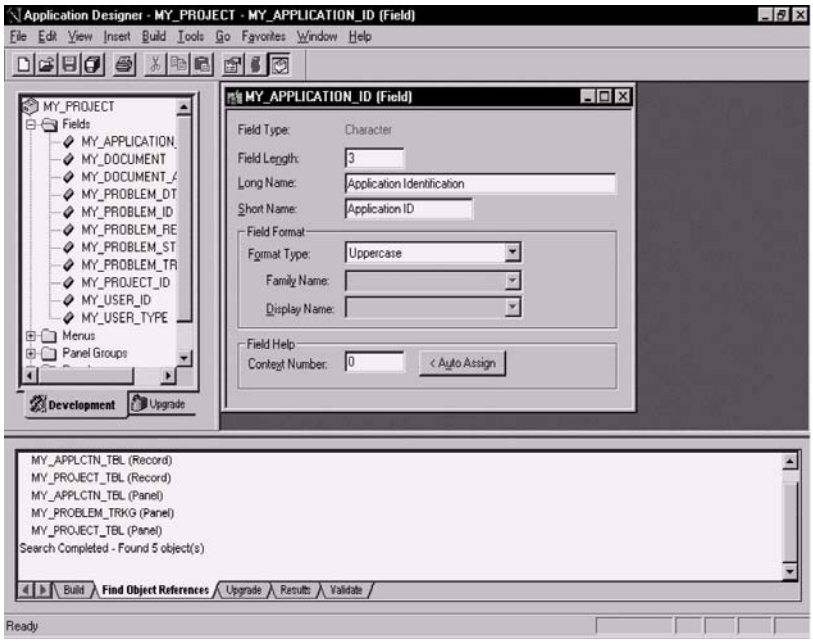


Figure 2.1 Find object references using the Application Designer

2.8.2 Find string In PeopleCode

Navigation: Edit →Find in PeopleCode

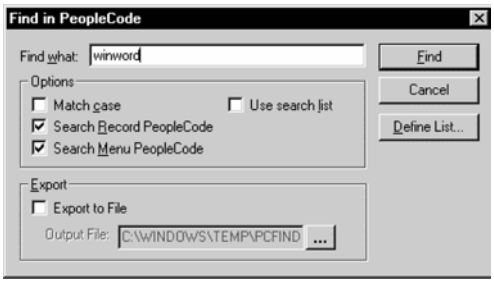


Figure 2.2 Find string in PeopleCode

Navigation: Define List (Find in PeopleCode)

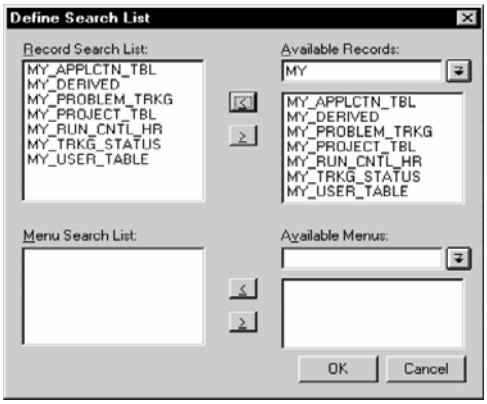


Figure 2.3 Define list of object to search

We can also search PeopleCode text in PeopleSoft to search for a string. This feature is useful for finding customized PeopleCode in the system. We should follow a convention in developing custom PeopleCode. We mark custom PeopleCode with a standard comment and use this comment as the string to search for all occurrences of custom PeopleCode. To search the PeopleCode for text, choose "Edit/Find in PeopleCode" from the Application Designer screen (figure 2.2).

We enter text as a search string in the box labeled "Find what." We also define a list of record definitions for search by clicking on the Define List push button. In figure 2.3 we define the list of record definitions and push them to the left-hand side of the screen. We click on "OK" to complete the list for search, and the output is displayed in the output windows of the Application Designer screen. When a list is not defined, all PeopleCode events in the database are searched for the occurrence of the text string.

The results of the PeopleCode search can also be exported into a file by choosing the Export to File option and specifying an output destination. The Match Case checkbox will match the string exactly for upper and lowercase characters.

2.8.3 Record Cross References

Navigation: Go →Utilities →Use →Record Cross Reference

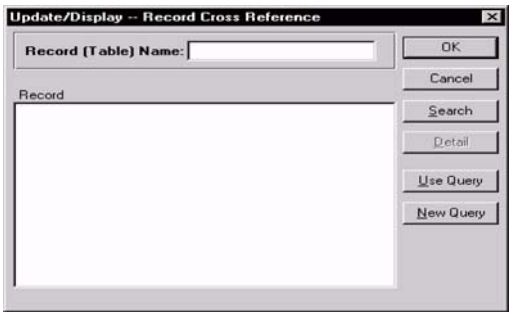


Figure 2.4 Record Cross Reference

We can find the Cross References for a record definition across all objects in the database by choosing "Use/ Record Cross Reference" from the Utilities menu.

By choosing the record name in the input dialog box in Figure 2.4 we can find all the objects in the database that refer to the record definition.

In figure 2.5, two tabs exist which show information on objects that refer to the MY_PROJECT_TBL record definition. The first tab shows all the panels, views, and menu items that use the record definition. The second tab shows prompt definitions, field defaults, and PeopleCode events that use the record definition. The Record Cross Reference is a great tool to use to determine where a particular record definition is being used.

Navigation: Use →Record Cross Reference (from the UTILITIES menu)

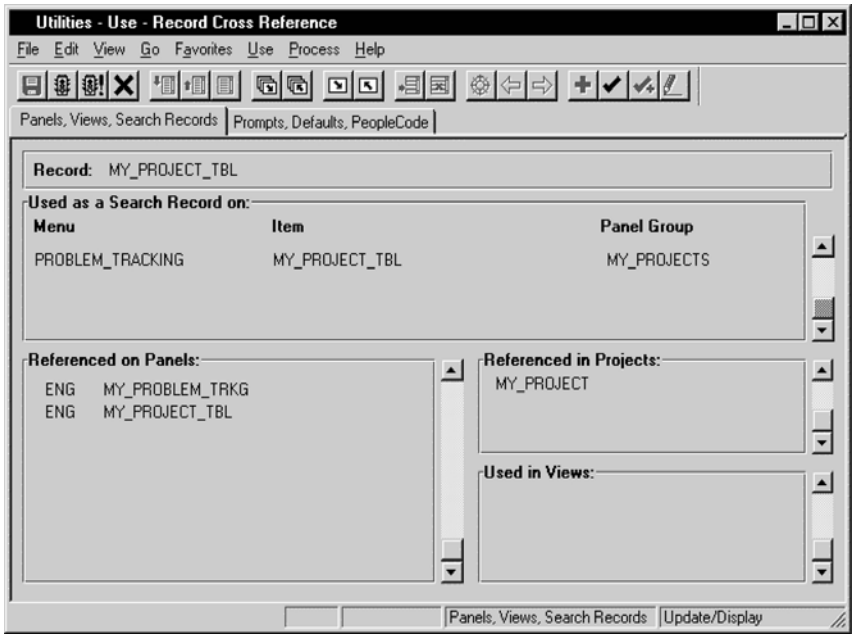


Figure 2.5 Record Cross Reference

All these utilities work using SQL views delivered with the PeopleSoft system. These views refer to the catalog tables that we have been visiting throughout this chapter. Some SQR programs, which come delivered with the system, also serve as cross reference utilities. These SQR programs start with an XRF prefix. For example, XRFFLPN.SQR shows references to fields from panels.

KEY POINTS

1. Application Designer is an integrated tool for application development in PeopleSoft.
2. Fields are the lowest level objects used in PeopleSoft application development.
3. Records are a collection of fields and can be defined as SQL tables, SQL views, Derived/Work records, Sub records, Dynamic views or Query views.
4. Panels are designed by assembling record fields on an empty panel.
5. Panel groups are mandatory objects that attach panels to menu items. Panel groups can also be a collection of panels assembled together.
6. Menu Items serve as user interface to an application panel. Menu Items are objects that are used in implementing panel security.
7. PeopleCode events help in creating logic during panel processing.
8. Cross-Reference Utilities come delivered with the PeopleSoft system and are very useful in application development.

CHAPTER 3

Administration tools

In addition to the tools discussed in chapter 2, PeopleTools provides mechanisms to load and unload data across database platforms. Simple prompt tables can be loaded using the Import Manager tool.

The Security Administrator provides a tool to manage PeopleSoft operator IDs. The ability to group many users into an operator class enables the individual(s) administering security to grant or revoke access to panels and processes much faster when the IDs are linked to a particular class.

Tree Manager is another tool that can be used for reporting security. This chapter presents an overview of these tools and will complement the information obtained in the chapters ahead.

3.1 DATA MOVER

Data Mover is a PeopleTool utility that enables the developer to move data from one database to another. Data Mover can also be used to move tables from one PeopleSoft platform to another. Most organizations have multiple databases such as production, QA, and development. Consequently, the need to move or unload data from one environment to another is a necessity.

3.1.1 Data Mover overview

Data Mover uses commands which can be entered ad-hoc or from predefined script files.

The scripts generated by Application Designer can be used in Data Mover to execute SQL statements against database tables without regard to database platform. Data Mover also uses scripts to load and unload data and to perform table manipulation. These scripts can be made up of Data Mover commands, SQL statements, or a combination of both.

Let's define a simple Data Mover script which unloads data from one environment and then loads the data into another.

Our first step is to sign onto the database. The sign-on process is similar to the PeopleSoft application sign-on described in chapter 1. After sign-on, the Data Mover window is displayed. This window is the mechanism used to process Data Mover scripts.

NOTE Data Mover scripts do not necessarily have to be defined in the Data Mover window. Scripts are generated by Application Designer during create and alter table actions. Scripts can also be developed using a text editor.

In figure 3.1 the assumption is made that the script is defined in the Data Mover window. After Data Mover is launched, a new script can be defined.

The Data Mover window consists of an input window and an output window. Statements are entered into the input window and processed by the Data Mover PeopleTool. The data are captured on a file that can be supplied to another Data Mover script during a migration of data to another database or platform. Data Mover is extensively used during upgrades and specific system updates such as patches and fixes. Additional applications of Data Mover can be used for backups and restores or the movement of setup data. In the PeopleSoft HRMS application, the use of setup data such as Company, Department, and Locations can be moved to another database on a regular basis. This is very useful during testing or when it is necessary to have an environment similar to production. Reusable scripts can be developed to unload data from one database and load it into another.

Navigation: File →New

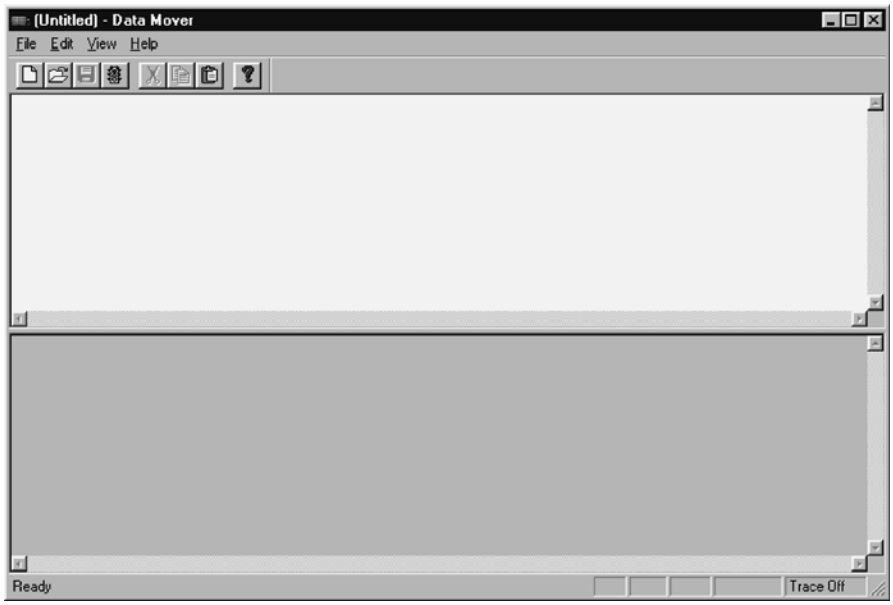


Figure 3.1 Data Mover window

NOTE The use of Data Mover may not always be feasible when very large tables are moved. For large tables, database specific utilities such as Oracle Export/Import or DB2 SQL loader may be used. The disadvantage is that those tools have specific platform and database dependencies.

We've developed an application called Problem Tracking, to illustrate the tools and concepts used in PeopleSoft. In figure 3.2, Data Mover scripts are used to export several tables from the Problem Tracking system.

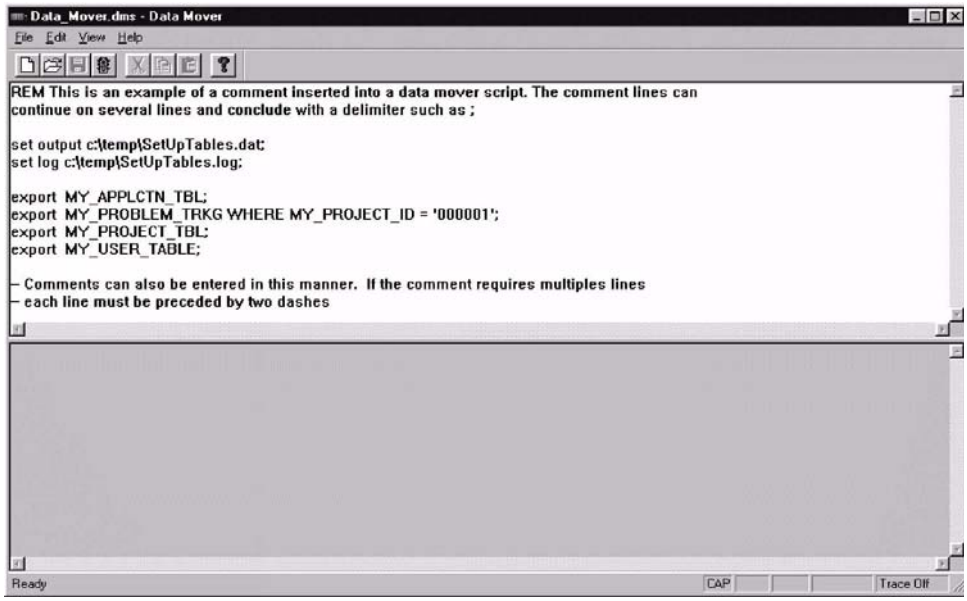


Figure 3.2 Data Mover commands

3.1.2 Examining the Data Mover script

Let's examine our Data Mover script. The first statement is a remark. In Data Mover, a remark can be entered by coding `REMARK`, `REM` or `--`. The next two statements are `Set` commands, which are used to identify the environmental settings with which Data Mover will work. In the example, the first `SET OUTPUT` statement identifies an output file. When tables are unloaded, the specified output file will contain the exported data. The next `Set` statement represents a log file which records Data Mover activity. The activity is also displayed in the output window.

The `Export` statements are used to write the specified records to the file identified by the `SET OUTPUT` statement. The `Export` statement can accept an optional `WHERE` block, which can be used to limit the selected data to specific values. The contents of the `WHERE` block can be any valid SQL statements. In the example, Data Mover will only export entries from MY_PROJECT_TBL which have a MY_PROJECT_ID field containing `'000001'`. The other tables are exported in their entirety. Data Mover commands are delimited using a semicolon (;).

To run the script, click on the traffic light or select File →Run Script (figure 3.3).

The log file contains statistics such as run date, data mover release, and number of records exported for each table. The output file SetUpTables.dat contains the exported table entries and data. In a real world example (and depending on the number and size of tables exported), a file such as SetUpTables.dat can become very large.

Navigation: File →Run Script

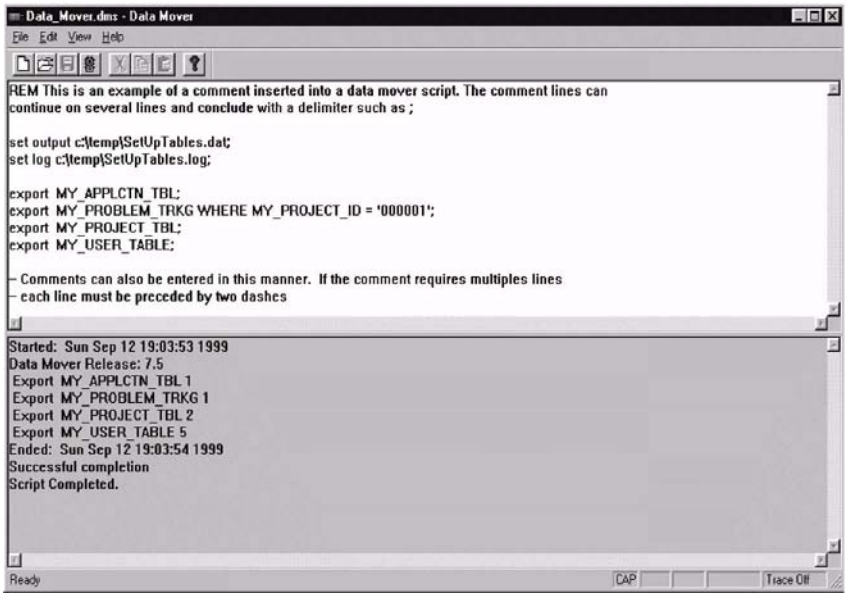


Figure 3.3 Data Mover windows after execution

This data can be saved for backup purposes or can be migrated to another database. Let's continue with the example and assume that the data will be loaded onto another database. Our next task is to load the data into another database. The statements used to accomplish this task are illustrated in figure 3.4.

The `SET INPUT` statement identifies the name of the input file for this script. In this example, the input file is the output file used in figure 3.3. The `SET LOG` statement refers to the file used in the export operation. The log file is written over and contains the results of the data load.

The next statement in the script is `REPLACE_ALL`. This statement drops the specified table and any associated indexes. The table and indexes are then created using the characteristics that appear in the input file. The data are then loaded into the specified table.

This example contains an embedded SQL statement. When the export operation is run as shown in figure 3.3, the `Where` statement is used to select only those rows in MY_PROBLEM_TRKG that contain a `'000001'` in the MY_PROJECT_ID field. This is done in order to preserve any existing data for MY_PROJECT_ID values other than `'000001'` on the target record. The `Delete` statement removes any `'000001'` values before the subsequent `Import` statement is processed. The `Import` statement does not alter existing table characteristics or data. For an existing

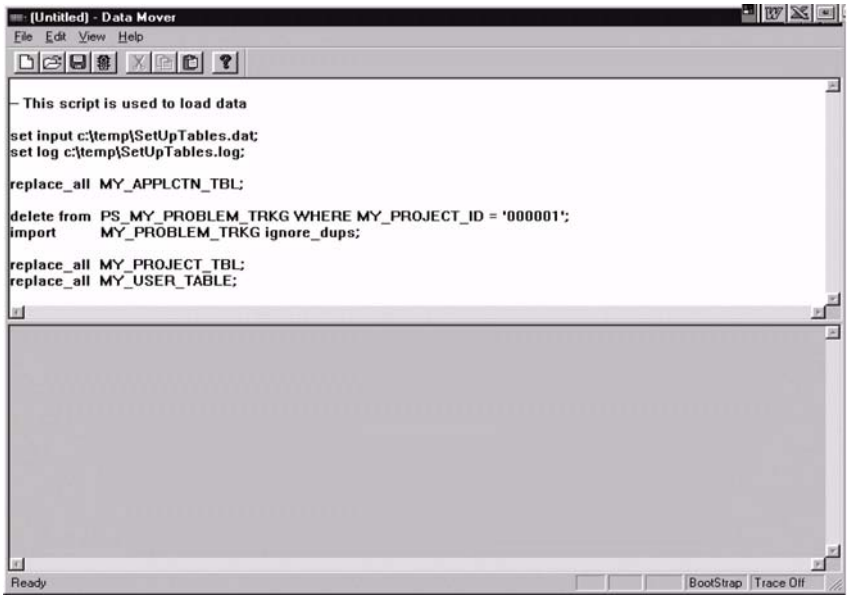


Figure 3.4 Data Mover script to load

Figure 3.5 Data Mover duplicate data

table, `Import` inserts non-duplicate rows only. Duplicate entries generate an error message similar to the one illustrated in figure 3.5. Duplicate entries can be ignored using the `IGNORE_DUPS` parameter. `IGNORE_DUPS` permits duplicate row messages without abnormally terminating the import operation. In the example, the `IGNORE_DUPS` parameter is redundant because of the preceding `Delete` statement. The parameter should be used with caution because there may be instances where duplicate rows might indicate design errors in the export and import process. `REPLACE_DATA` is a version of the `Import` statement. The difference is that `REPLACE_DATA` first deletes data from the table and then inserts the corresponding data referenced on the input file.

3.2 IMPORT MANAGER

Import Manager is another tool that can be used to load Application data. A popular use of Import Manager is the conversion of data from one system to another or from one set of codes to a format compatible with PeopleTools tables. One unique feature of Import Manager is that, while data are loading, system edits are being performed as if the data were entered from a PeopleSoft application panel. Edits can also include

PeopleCode programs if necessary, and PeopleCode programs can contain code to execute when they are run during the Import Manager process only.

Import Manager works very closely with PeopleTools record definitions. The fields on the record definition are mapped to data on the Import Manager upload file. A great way to learn Import Manager is by example, so let's begin.

3.2.1 Defining an import definition

The following example loads data obtained from an existing legacy application to a soon-to-be implemented PeopleSoft HRMS system. The data are loaded into the Department table, and our objective is to build as much of the record information as possible. After the departments are loaded, the end user can complete the record with additional information.

The input file exists as an ASCII file with a fixed length format. (These are file requirements for Import Manager.)

Let's begin the Import Manager session:

Navigation: PeopleTools →Import Manager →File →New

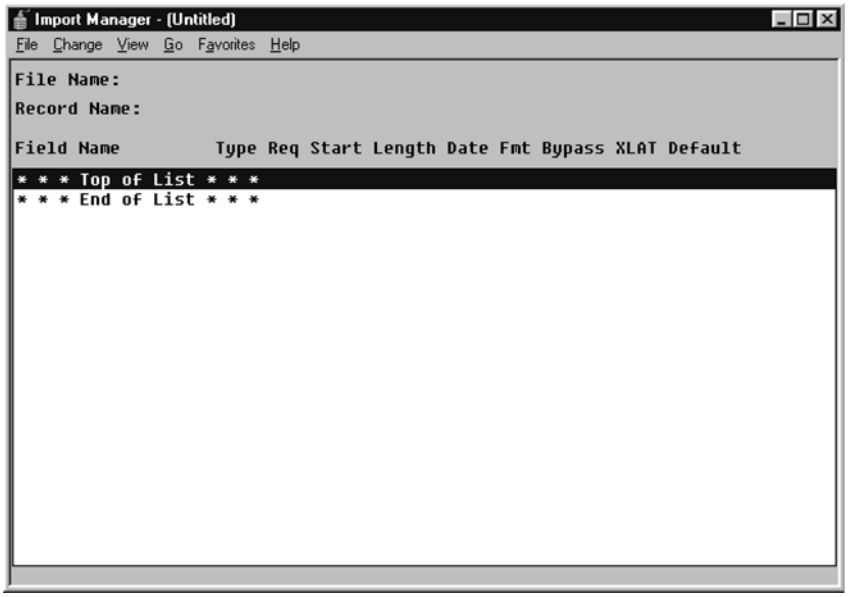


Figure 3.6 Import Manager window

Our next step is to assign a file name and record name to the Import Manager definition.

Navigation: Change →Header

Figure 3.7 Header menu option

The Import Header information is displayed (figure 3.7). The record name is selected from a list box containing all available record names as shown in figure 3.8.

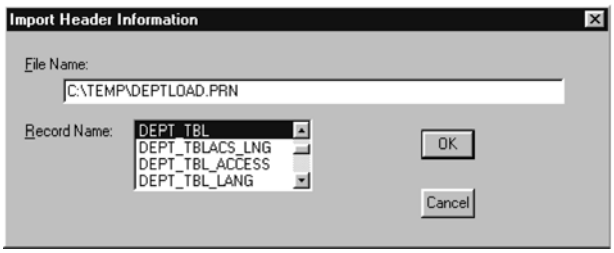


Figure 3.8
Defining header information

After the Import Header information is entered, we click OK and are presented with a list of field names from the DEPT_TBL record definition. The record field names are then available in the Import Manager window (figure 3.9).

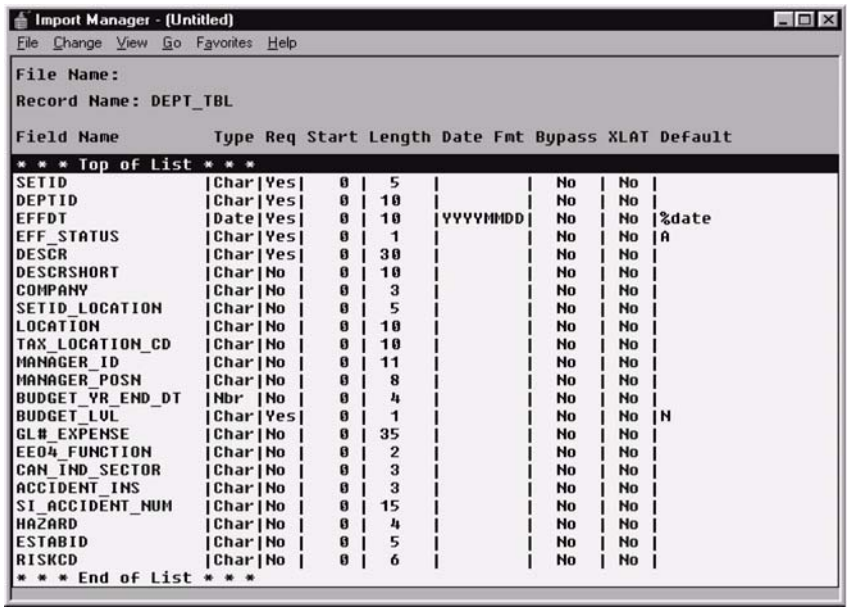


Figure 3.9 Record Definition window

In this example, the Import Manager upload file contains basic information. To load a "stripped down" version of the department table record, three data elements in the input file are all that are required. By combining default values and one PeopleCode program, data can be loaded into the department table using the small input file illustrated in figure 3.10.

The mapping process can now begin. During this process we have several options. We can:

- point fields to columns on the input file,
- default fields by supplying default values in the Import Field definition, or
- provide no specific reference to fields. (The assumption is that this category of fields can either be populated later or will not require a value at all.)

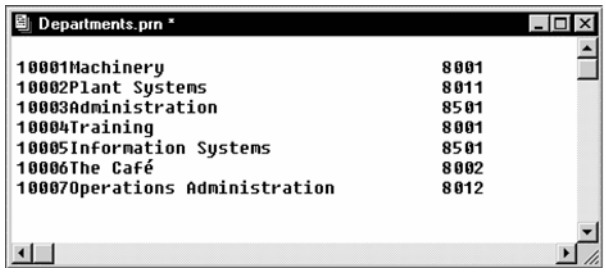


Figure 3.10
Import Manager data file

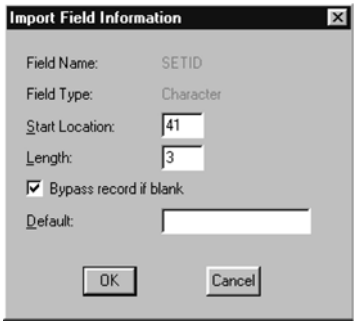


Figure 3.11 Field information window

The first field on the record is SETID. We double-click and enter the starting location and length on the corresponding input record. The dialog box to enter this information is illustrated in figure 3.11.

The default starting location is zero, and the default length is obtained from the record. In the example, SETID and COMPANY use the same value. This enables us to map these two fields to one specific column on the upload file.

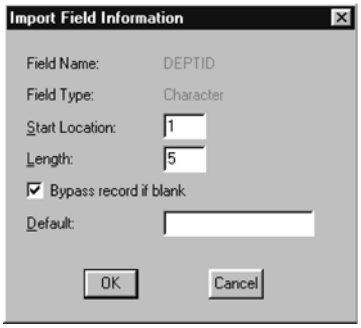


Figure 3.12 Assign department ID

The next field is DEPTID. DEPTID is the field which appears in column 1 on the upload file. The assignment for DEPTID is shown in figure 3.12.

Figure 3.13 Date formats allowed

The next field is the Effective date. EFFDT is defaulted to the current date (figure 3.13).

Import Manager default values are taken from the default value setting in the Application Designer Record Field properties. When an actual date value is loaded, Import Manager allows date formats to be specified.

Another feature of Import Manager is that field contents can be translated from one value to another. The translation process occurs before the record is inserted into the database. This is useful when data are being converted from a legacy application. In the example, column 39 on the upload file contains a comparable effective status value as it exists on the legacy application. We know that, in PeopleSoft, the EFF_STATUS field Xlat values are `A` and `I`, which represent `Active` and `Inactive` respectively. The legacy application equivalents are `1` and `2`. The EFF_STATUS field information appears in figure 3.14.

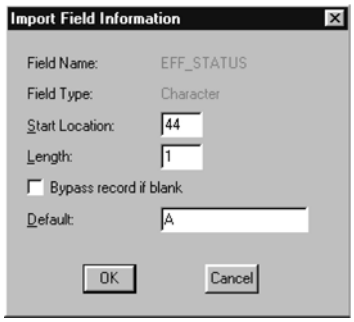


Figure 3.14 EFF_STATUS

When loading the effective status, blank values are not bypassed; they are defaulted to A. To set up translate values for EFF_STATUS, the field is highlighted in the Import Manager window followed by the corresponding menu action.

The Translate Values dialog box (figure 3.15) enables the entry of old and new values. Figure 3.16 illustrates how old and new values are entered into the dialog box.

Navigation: Change →Translate

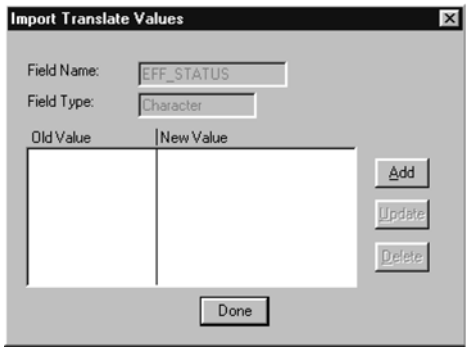


Figure 3.15 Translate values dialog

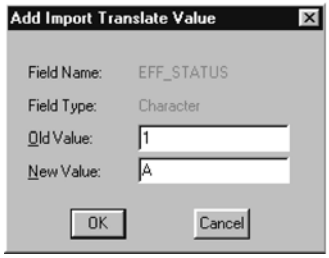


Figure 3.16 Entering translate values

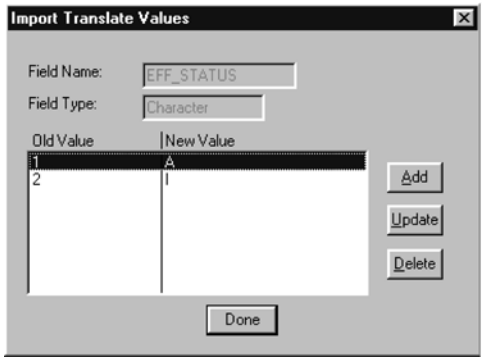


Figure 3.17 Completed translate values

The completed translate values are identified in figure 3.17.

Figure 3.18 Department description

The next two fields, DESCR and DESCRSHORT, contain the full department description as well as a short description. On the upload file, the description appears in column 6. Because the next field on the upload file begins in column 36, we allow a 30 character length for DESCR and 10 for DESCRSHORT. The field information for DESCR is illustrated in figure 3.18.

The start and end columns of data on the upload file must be tracked carefully. It is possible to load the contents of one field into an incorrect column on the record. This is particularly true for fields that do not contain edits.

The "Bypass record if blank" box is checked for COMPANY, SETID, and DESCR. For EFF_STATUS, blank fields are allowed and are subsequently defaulted to A. Fields considered important, such as descriptions, key fields, or codes, should not be passed as blank whenever possible. In our example, we are loading "skeleton" records onto the department table which contains the significant data elements (SETID, DEPTID, DESCR). The EFF_STATUS is defaulted to `Active` if the data are not available. Some end-user involvement may exist after the process is completed. For this example, identifying departments as active or inactive can be one of these tasks.

The next field, COMPANY, shares the same position on the upload file as SETID. All other remaining fields are either defaulted based on the record definition or are left blank.

Navigation: Import Manager →File →Save

Figure 3.19 Import Manager save dialog

After these tasks are completed, our next step is to save the import definition (figure 3.19).

The Import Manager definition is now completed (figure 3.20). To view the definition based on the input order on the file, use the view menu option.

Navigation: View →Input Order

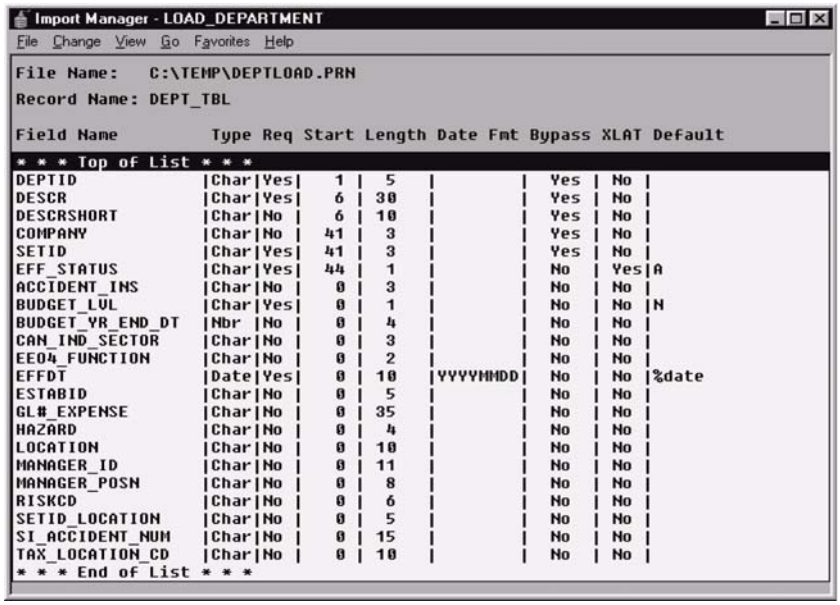


Figure 3.20 Completed Import Manager definition

When viewing the import definition fields we can see that Import Manager has several menu options available:

Navigation: View →Record Order	This is the default. Fields are displayed as they appear on the record definition.
Navigation: View →Input Order	Fields are displayed based on starting column positions of the upload file.
Navigation: View →Alpha Order	Record is displayed based on alphabetical field names.

One additional item which can play a role during the Import Manager process is PeopleCode. When the import definition is run, the following PeopleCode events are executed:

- `RowInit`
- `FieldEdit`
- `FieldFormula`
- `SaveEdit`
- `SavePreChg`
- `WorkFlow`
- `SavePostChg`

Based on the PeopleCode events identified, let's insert a small piece of code that concatenates the company into the description field. This will help identify departments which were migrated from the legacy system. The code is inserted into the `SaveEdit` event:

```
If %Import = True Then
   DESCR = RTrim(DESCR, " ") | " (" | COMPANY | ")";
End-If;
```

The `%Import` system variable is verified for a return value of `True`. The system variable indicates this is an Import Manager session. During non-Import Manager sessions the variable returns `False`. As a result, any code in the context of the `If` statement is not executed.

NOTE PeopleCode is discussed in part 3 and appendix E also contains a selected list of PeopleCode built-in functions.

3.2.2 Running the Import Manager

At this point, we are now ready to run the import. The run import dialog box, shown in figure 3.21, contains a list of import definitions, run types, and report parameters.

Navigation: File →Run

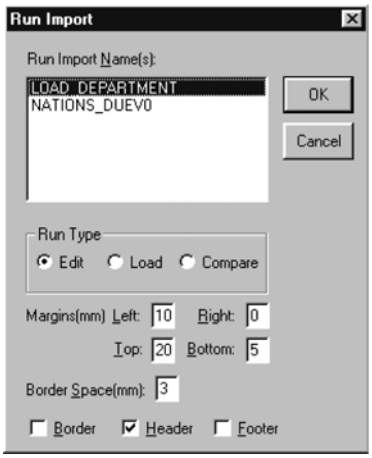


Figure 3.21 Run Import window

Import Manager can be run in one of three modes:

Edit	Generates a report based on the data in the upload file. No database inserts are performed.
Load	Attempts to write to the database. Edits are also performed against record keys, and translate and prompt tables. Effective date processing is implicitly performed.
Compare	Performs the same function as Edit. This is reserved for future use.

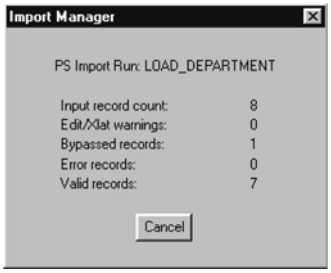


Figure 3.22 Import Manager run status

Import Manager displays a run status window (figure 3.22) that indicates the number of input records processed. For very large input files, we can identify the number of valid records or records that generate errors as Import Manager is processed.

Import Manager produces a report containing the record counts (figure 3.23).

```
Printed: 09/12/99  8:02:08PM   Database: HRDMO   PS Import Run: LOAD_DEPARTMENT   Version: 26   Page: 1

Record Name: DEPT_TBL
Run Type:    Edit

End of run.
    Input record count:  8
    Edit/Xlat warnings:  0
    Bypassed records:    1
    Error records:       0
    Valid records:       7
```

Figure 3.23 Import Manager report

Finally, let's look at the end results. The department profile within PeopleSoft is illustrated in figure 3.24. Note the embedded company code in the description field. This is accomplished using the small piece of PeopleCode inserted into the `SaveEdit` event.

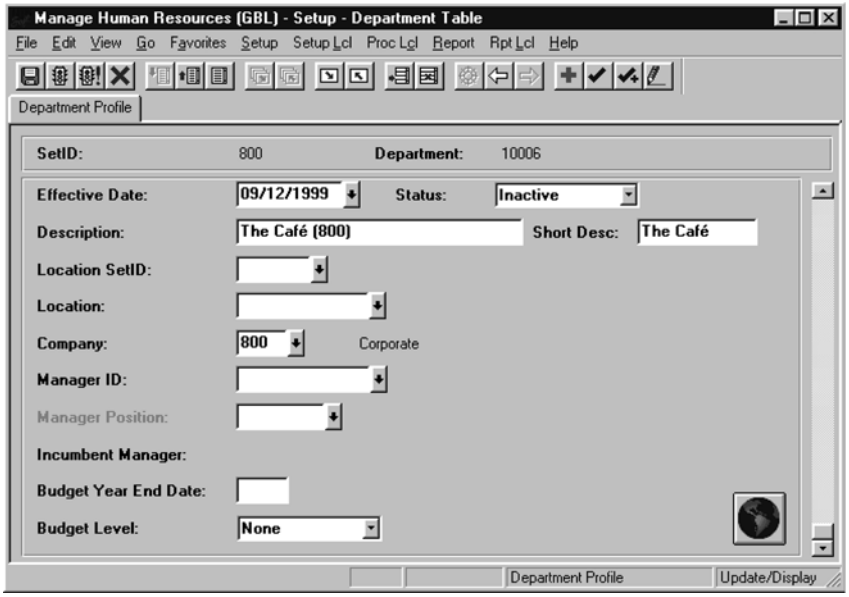


Figure 3.24 Results of Import Manager run

3.3 SECURITY ADMINISTRATOR

PeopleSoft security is comprised of operators and classes of operators. An operator is the individual ID assigned to a user. A class of operators is a profile that includes the menu items and type access each operator assigned to the class will have.

3.3.1 Defining an operator class

In this section we define a simple operator class and assign an operator ID to it.

The Security Administrator can be entered using the menu (figure 3.25).

Navigation: Go →PeopleTools →Security Administrator →File →New

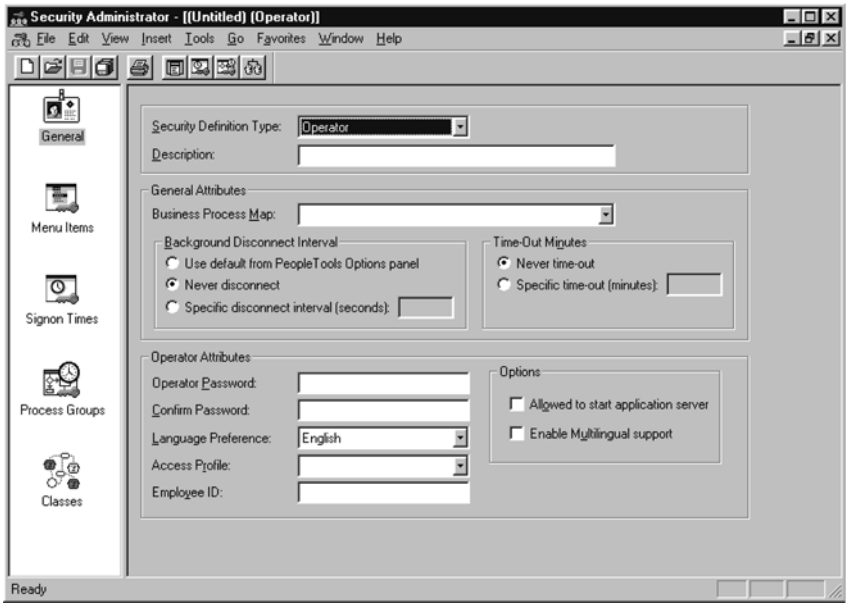


Figure 3.25 Security Administrator panel

Several options are available based on whether the security definition type is operator or class of operators:

- General
- Menu items
- Sign-on times
- Process groups
- Classes

General

First, we define a class of operators. To define a class, select "Class of Operators" from the Security Definition Type list box. An optional 30 character description can be entered. The General Attributes section contains several settings. Business Process Map identifies the path within the business process to which the operator class will have access to when using the Navigator. In HRMS, business processes are events such as New Hires and Terminations. These events are graphically represented in the business process.

The Background Disconnect Interval uses the default from the PeopleTools option panel. This feature enables a database connection to be released when the instance is moved to the background or is minimized as an icon. The connection is reestablished when a database call is required for SQL statements.

The Time-Out feature specifies the amount of inactivity allowed in minutes. The system signs off the ID after the threshold period has been exceeded. The General options are illustrated in figure 3.26.

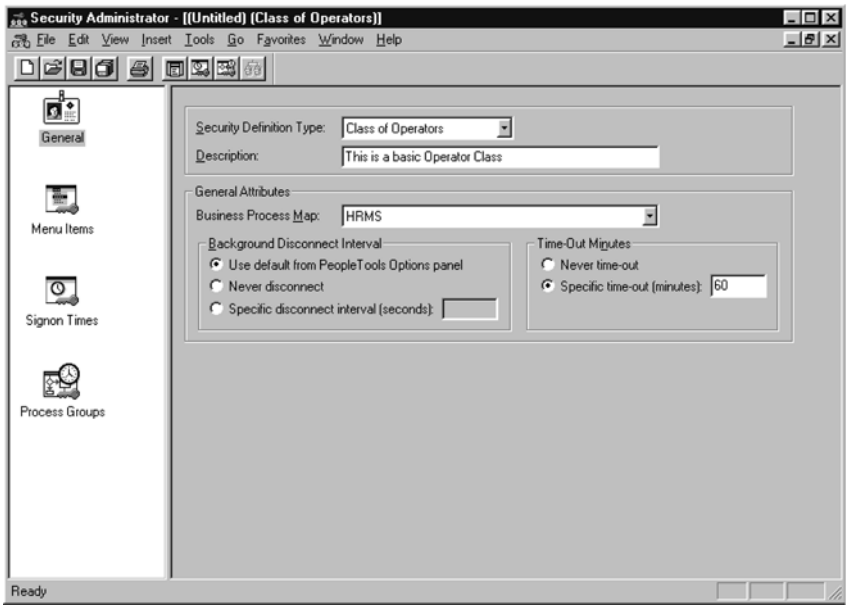


Figure 3.26 General Security options

At this point, we can save the panel by supplying an operator class name, then proceed to insert menu items.

Menu Items

Menu Items are the main components of an operator class. Here we select menus and make specific menu items accessible to each operator class. Click on the Menu Items icon within the Security Administrator window.

Our next step is to insert a menu (figure 3.27).

A list of available menus is provided. In the example, we select the ADMINISTER_WORKFORCE_(GBL) menu by double-clicking on its name. A list of associated menu items is displayed (figure 3.28).

When the Select Menu Items window is presented, we can either press the Select All button or select only those menu items the operator class is allowed. View Only security can be given by clicking on a menu item and then clicking the Change Display-Only button. The DispOnly column will then contain the value `Yes`. Display Only security

Navigation: Insert →Menu Name

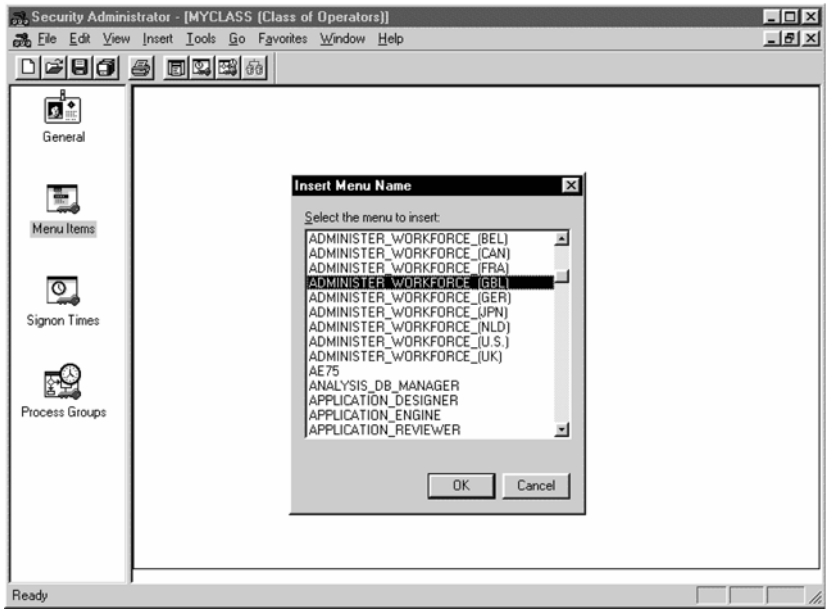


Figure 3.27 List of available menus

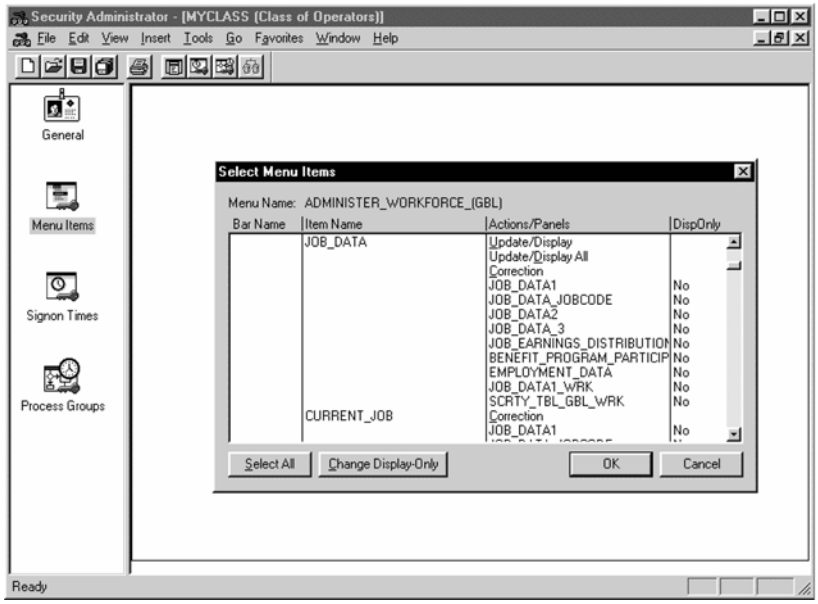


Figure 3.28 Select menu items window

permits the operator to view data, but does not allow changes. Figure 3.29 identifies the menu items selected for the ADMINISTER_WORKFORCE_(GBL) menu.

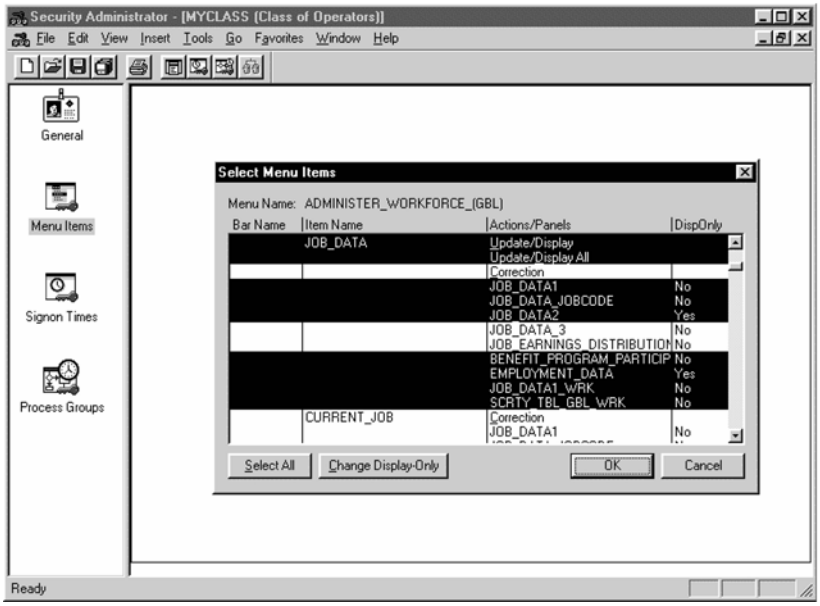


Figure 3.29 Menu Items selected. Note DispOnly option Yes

Note the DispOnly column for the JOB_DATA2 and EMPLOYMENT panels. Based on the menu items selected, any operator IDs assigned to this class have access to the JOB_DATA panels as follows:

- JOB_DATA1 Update/Display, Update/Display All
- JOB_DATA_JOBCODE Update/Display, Update/Display All
- JOB_DATA2 Display Only
- BENEFIT_PROGRAM Update/Display, Update/Display All
- EMPLOYMENT_DATA Display Only

To enable the operator class to run queries, the Query menu is inserted into the profile as well. The operator class now contains two menus (figure3.30).

Signon times

To display sign-on times, click on Signon Times or select View →Signon Times (figure 3.31):

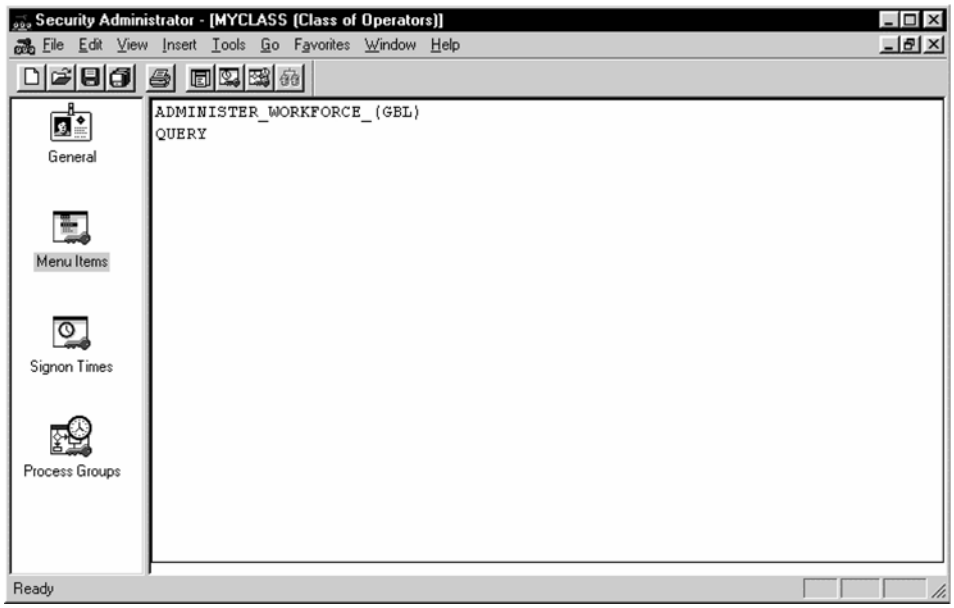


Figure 3.30 Menus allocated to operator class

Navigation: View →Signon Times

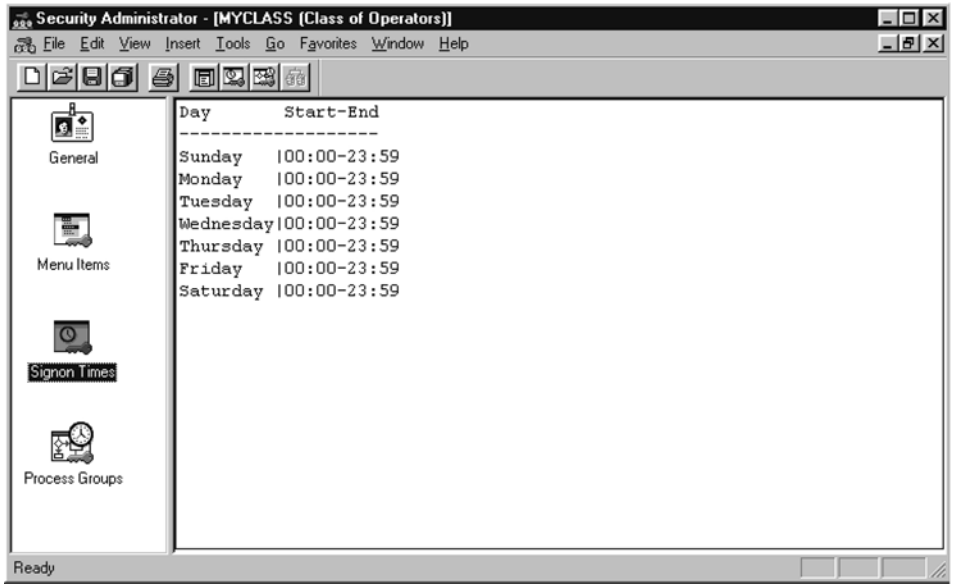


Figure 3.31 Authorized sign-on times

Sign-on times identify what days of the week (and hours within those days) during which a user is authorized to logon. In some organizations sign-on times are restricted to prevent updates during batch cycle runs such as Payroll. Occasions also exist when activities such as database reorganizations or data migrations occur, which may require limited access for specific periods of time.

The authorized sign-on times display consists of two columns, Day and Time, representing the allowed sign-on times for each day of the week. To change the sign-on times, double-click on the day of the week.

We can have multiple sign-on times during the day, provided the time ranges do not overlap. Let's assume, for example, that Saturday is the system maintenance window, and, with the exception of a few IDs, no one else should be on the system between 17:01 and 21:30. To enforce this, we can change the time range for Saturday and then insert an additional time period which allows for sign-on after the maintenance window.

To change the Saturday sign-on time, double-click on Saturday and change the time as shown in figure 3.32.

This change enables the user to be logged on between 00:00 and 17:00. To add additional time periods, select Insert →Signon Times (figure 3.33).

Modifying the sign-on time ensures that our users are not on the system during the Saturday time period 17:01–21:29. This information may help provide clues when a help desk report reads `User mysteriously disconnected from system`.

Process groups

The next available item in the Security Administrator window is process groups. When process definitions are set up using the Process Scheduler PeopleTool, the definitions are linked to one or multiple process groups. When a process is defined and attached to a process group, that process cannot be run by an operator until the class contains the process group as part of its definition. A process group can contain many process definitions.

To add a process group, select Insert →Process Group (figure 3.34).

NOTE Process definitions are discussed in part 5.

At this point we have an operator class with two menus. The class is disconnected after sixty minutes of inactivity. With the exception of the Saturday 17:01–21:29 time period, the operator class can be logged onto the system at all other times.

3.3.2 Linking operator IDs to an operator class

Now that an operator class has been defined, we need to define an operator ID and link it to the class. Before proceeding, do not forget to save the operator class settings.

To establish an operator ID, click on the General icon in the Security Administrator window or select the View, General menu item.

The next step is to change the security definition type to Operator and enter an optional description. The General Attributes section is required when an operator ID is not linked to an operator class.

The operator attributes are defined as follows:

- *Operator Password* The Operator Password is the password the operator will require during sign-on.
- *Confirm Password* This field is compared to the operator password and must match its value.
- *Language Preference* This option overrides the base language setting defined in the configuration manager. The option can be left blank to use the default setting. The language preference can be overridden using the Configuration Manager as discussed in part 1.
- *Access Profile* After the operator has been validated during logon, this parameter identifies the ID and password used to connect to the database. A typical access profile is SYSADM.
- *Employee ID* The Employee ID field contains the operator's PeopleSoft EMPLID, which is compared against the database and prevents users from changing their own data.

Additional options also exist. They include:

- *Start the application server* This option indicates whether or not the user is allowed to start the application server. Access to this option is commonly limited to system administrators.
- *Multilingual support* This option permits the user to work in multiple languages. The user does not have to log off or change his/her language preference setting.

The appropriate fields are entered and saved. The operator ID profile is illustrated in figure 3.35.

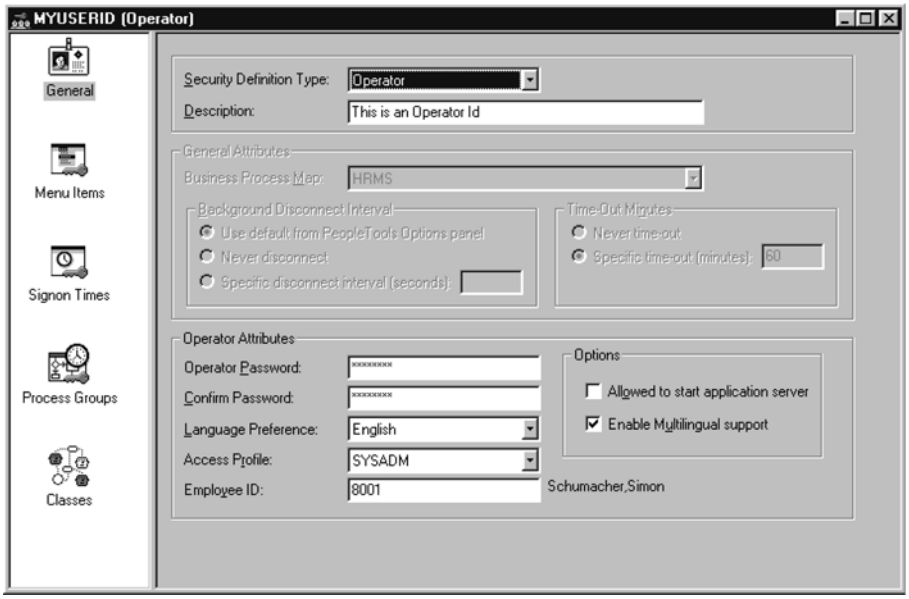


Figure 3.35 Operator ID attributes for MYUSERID

In this example, the operator ID is linked to the MYCLASS profile which implies that the menu items, sign-on times, and process groups will be defaulted from the class profile.

To link the operator ID to one or more classes, click on the Classes icon in the Security Administrator window or select View →Classes From the Menu (figure 3.36).

Navigation: View →Classes

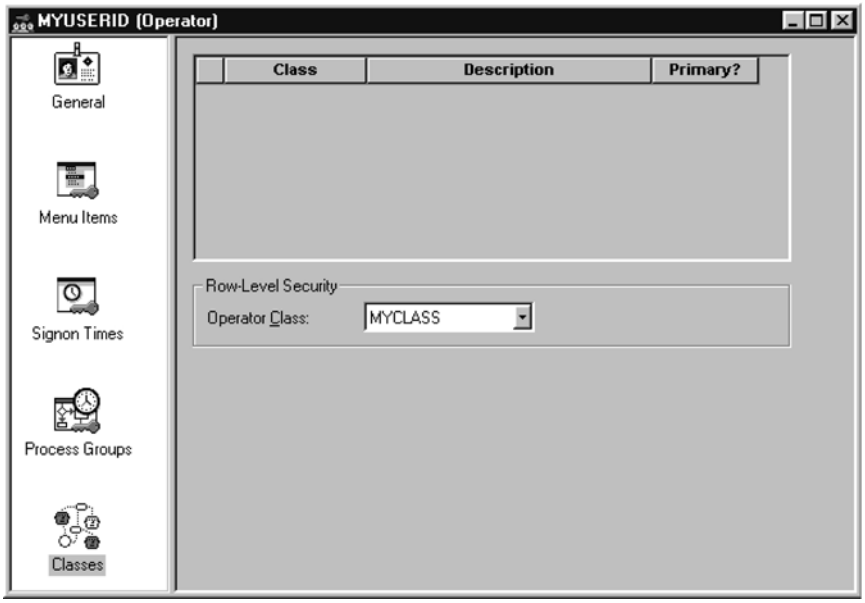


Figure 3.36 Viewing operator classes linked to an ID

Navigation: Insert →Class

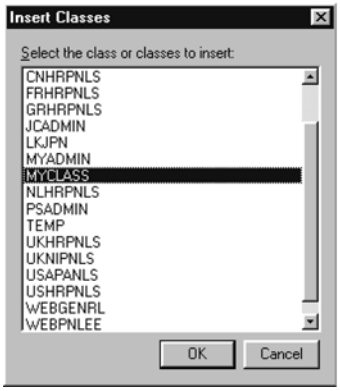


Figure 3.37 Attaching operator class

To attach one or more operator classes to an ID, we can select Insert →Class (figure 3.37).

The operator ID MYUSERID is now linked to the class MYCLASS and is authorized to use the menu items selected.

Operator classes serve an important function, particularly from a table space and system administration perspective. The System Administrator allows operator IDs to be defined without a link to an operator class. This may be useful for a handful of IDs, but when hundreds or thousands of IDs are necessary, quite a large amount of system resources and administration time are involved. The one-to-many ratio between a class and many operator IDs reduces the number of rows in the PSAUTHITEM system table. This tool table contains the authorized menu items allocated to an operator ID or class. Without classes, a system with one hundred users would require at least one hundred times the amount of rows in the PSAUTHITEM table. System resources are improved, for example, when ten distinct classes

of operators are allocated among the one hundred operator IDs. From an administration perspective, security is facilitated when IDs are grouped into classes.

When it becomes necessary to add additional operator classes, we use the Insert Classes menu and select the appropriate classes. An example of MYUSERID with additional classes allocated is illustrated in figure 3.38.

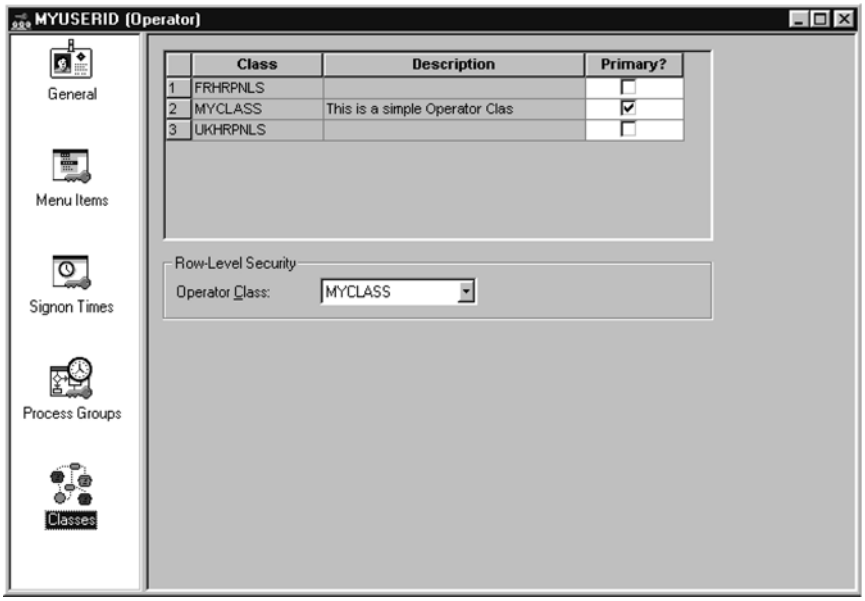


Figure 3.38 Allocating classes to an ID

The example in figure 3.38 identifies MYCLASS as the primary operator class. Any menu items associated with the two additional classes are made available to MYUSERID. Row level security is used to limit access to specific data linked to the security search views.

3.3.3 Restricting Application Designer Access

An IS organization can include developers and functional analysts supporting or implementing a PeopleSoft application. Sometimes a rift between developers and functional staff, and even among developers themselves, may arise when the issue of security is addressed. The question of who has access to specific tools is often a touchy subject. Some developers do not want undocumented changes made to objects. This is a justified argument, particularly when the changes cause production problems or do not appear until the next upgrade, during which time the culprits have either moved on or developed short-term amnesia.

When a group requires restricted access to an entire object type, such as panels and menus in the Application Designer, the access can be restricted using the Application Designer Access feature shown in figure 3.39.

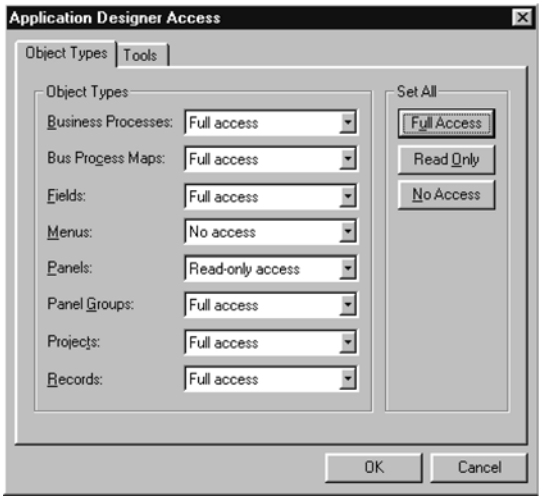


Figure 3.39
Restricting access to an entire object type

In release 7.xx of PeopleTools, object types are defined as:

- business processes
- business process maps
- fields
- menu definitions
- panel/panel group definitions
- project definitions
- record definitions

The example in figure 3.39 does not allow menu access in Application Designer and enables read-only access to panels. Now, let's assume we have someone who is assigned to work on specific functionality but requires access to update menus and panels. This can cause a dilemma. On the one hand, we want the person to make the necessary changes to a specific set of panels and menus. Alternatively, we have to change the settings to Full Access for menu and panel objects. How can this quandary be solved? The answer is Object Security, our next topic of discussion.

3.4 OBJECT SECURITY

Object security is an additional layer of security which can be used to restrict access to PeopleTools objects such as record definitions, panel definitions, menus, and others.

Object security is also used at the field level. When a modification to a field on a record is required, object security access is necessary. A change that involves the modification of a field label or field attribute will require access to every record that contains the field.

3.4.1 Object groups

Object groups are the entities defined using object security linked to security administrator profiles. The object group can be comprised of objects related to specific applications such as Human Resources or General Ledger. The object group can also include PeopleTool objects such as records, menus, and panels. Object security is applied to object groups only and not to individual objects.

Let's define a simple object and attach it to a security profile. The initial object security window is shown in figure 3.40.

Navigation: Go →PeopleTools →Object Security →File →New Group

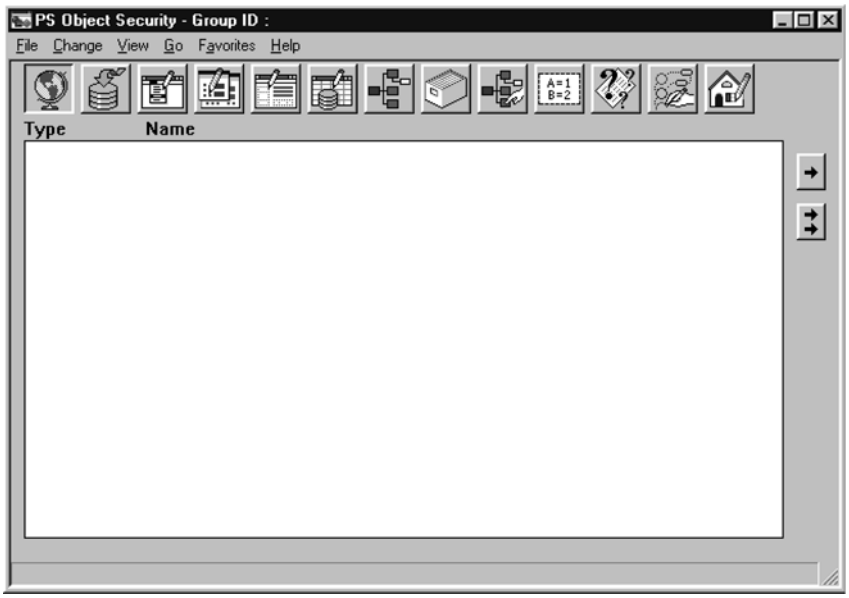


Figure 3.40 Object Security panel

Let's assume that we are working with a developer to enhance a few of our custom applications. These applications include some HRMS objects. The developer is allowed access to these objects only. To define an object group, we must select which objects to add or remove from the group. When defining a new object group, the

objects associated with the object type are selected using the toolbar or menu. To select record objects, select View →Records From the Object Security Menu (figure 3.41).

Navigation: Object Security →View →Records

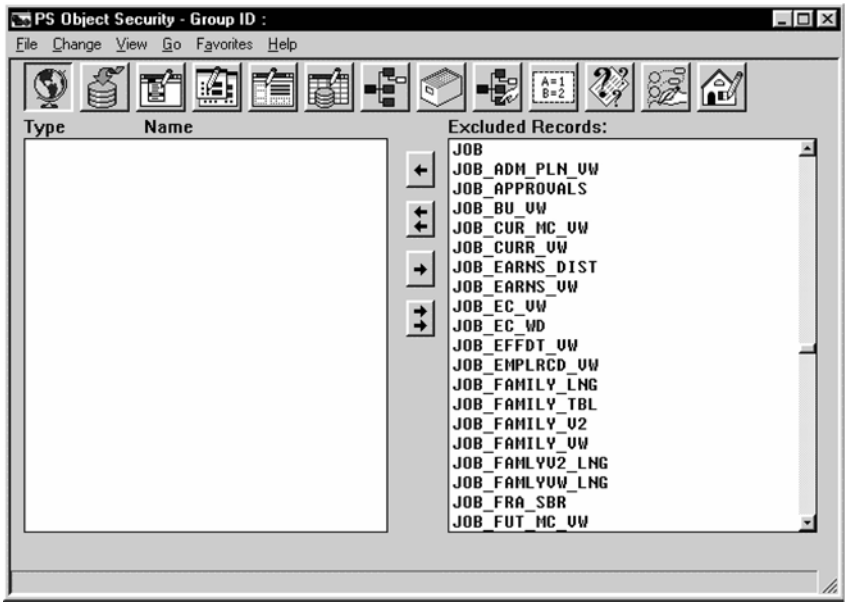


Figure 3.41 View record definition objects

For this example, we can choose the records prefixed with JOB and move them to the left side of the panel. The left side indicates the objects included in the object group and the right side represents the excluded objects. The arrows can be used as follows:

- moves selected object(s) to the left side
- adds all objects in the excluded group to the left side
- moves the select object(s) to the excluded group
- moves all objects to the excluded group

The selected records and panels related to specific HRMS functionality are also added to the object group. The object group build is completed with the addition of panel groups and menus. After all the objects are added, the object group can be saved.

After the object group is saved, the objects can be viewed using the View All menu selection. The selected objects and related types are displayed in the panel (figure 3.42). The Type column contains an identifier for the object type as follows:

- P=Panel
- B=Business Process
- Q=Query
- E=Tree
- R=Record
- G=Panel Group
- S=Tree Structure
- I=Import
- U=Business Process Map
- J=Project
- X=Translate Table
- M=Menu

Navigation: Object Security →File →Save

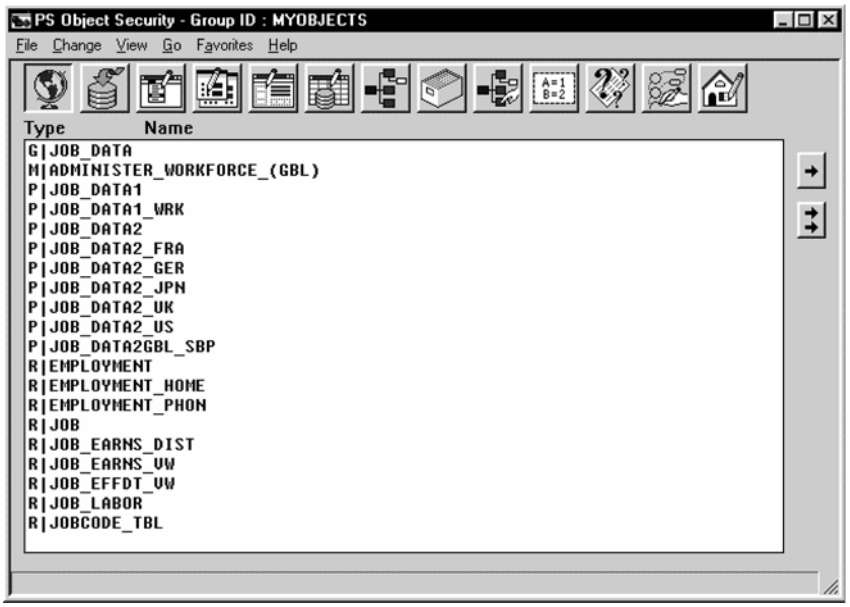


Figure 3.42 Completed object group

3.4.2 Linking object groups to security classes

After the object group is defined, it is then allocated to security class profiles. To assign an object group to a security class, it is necessary to open the class profile (see figure 3.43).

Navigation: Object Security →File →Open →Operator

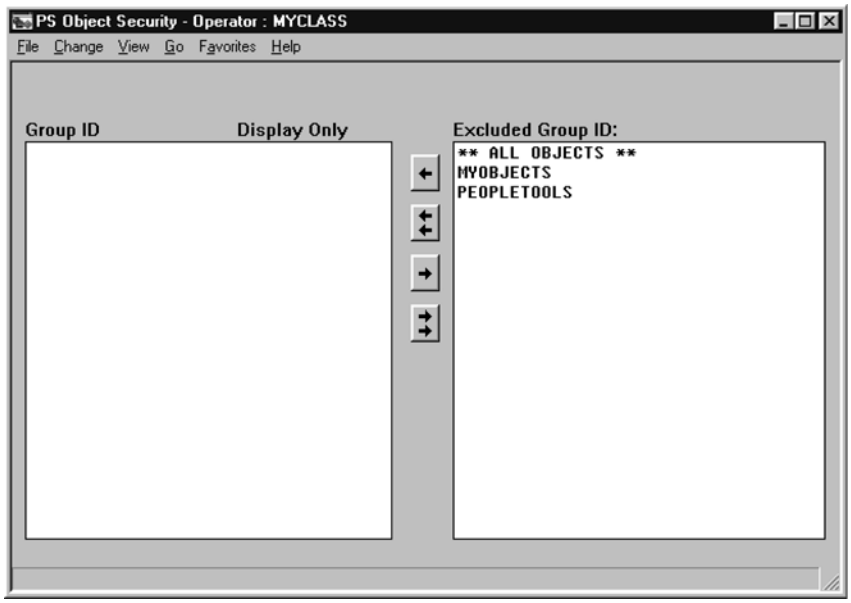


Figure 3.43 Assigning object group to class profile

The arrow buttons can be used to move object groups to the left side. The arrows work in the same manner as when they are used to allocate specific objects, such as records and panels. The left side indicates the group IDs allocated to the class.

To enable view only, select Change, Display Only from the menu. This displays the Object Security list dialog that contains the object groups allocated to the class. An individual object group can be selected or all the groups can be selected by pressing the All button. When objects in a group are specified as display only, it is a cliché for "Look, but don't touch." This implies the objects cannot be modified.

The completed object group and link to class MYCLASS are shown in figure 3.44.

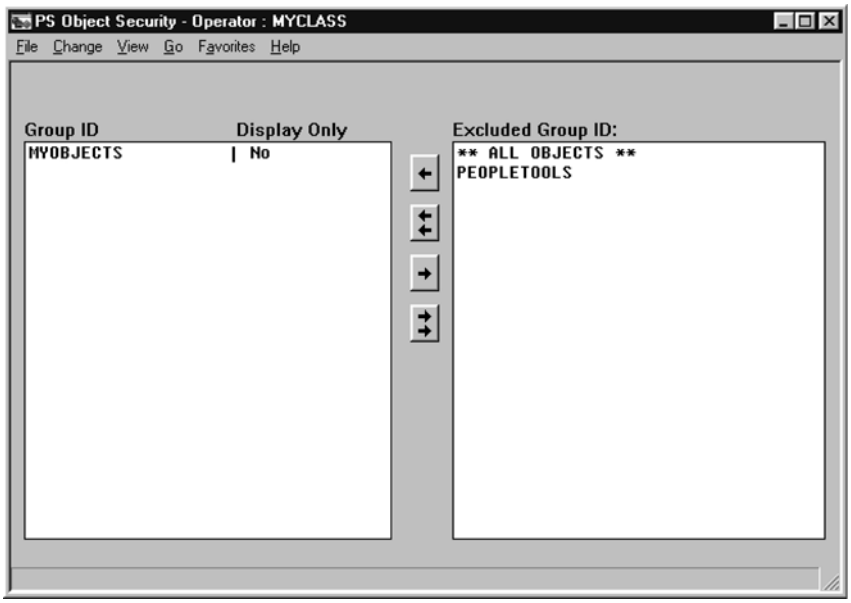


Figure 3.44 Completed object group and class profile

3.5 OPERATOR PREFERENCES

After an operator ID has been established, the Operator Preferences panel can be used to specify defaults such as Business Unit, SetID, Company Code, Country, and Currency Code. Additional settings include standard hours and payroll system. When the operator logs on to the application, default values specified on this panel will be used when necessary. This facilitates global implementations and reduces some level of functionality which is used to provide specific defaults such as country or currency code, based on system identifiers (menu, operator class).

To display the Operator Preferences panel, we can use the menu.

Navigation: Define General Options →Setup →Operator Preferences

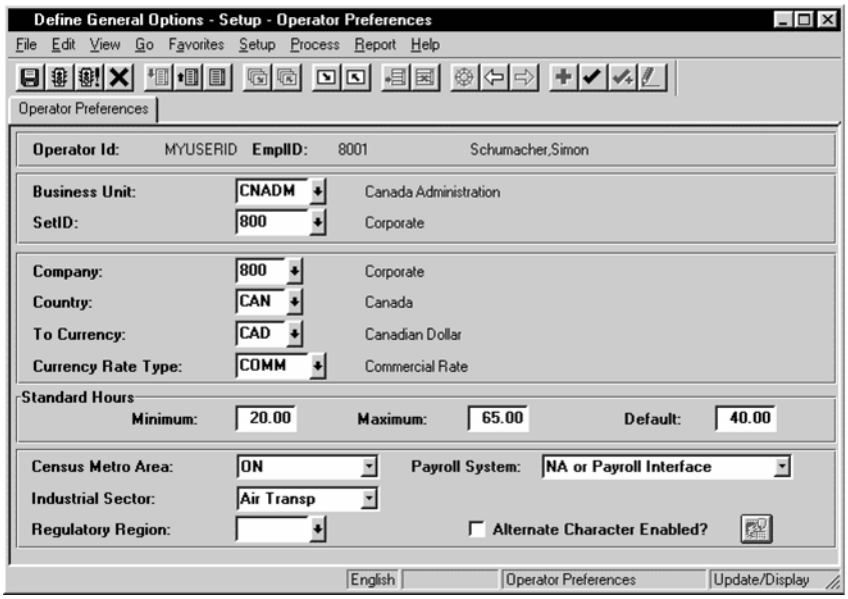


Figure 3.45 Operator preferences

3.6 TREE MANAGER

Tree Manager is a tool that can be used to graphically represent a hierarchy such as the departmental structure within an organization or the relationship between tables in a database. Tree branches or nodes can be traveled up or down, expanded, collapsed, and used to drive processes such as reporting and security. A tree can be used to produce a report identifying the tables an operator class can query.

The illustration in figure 3.46 is an example of a tree and the nodes, branches, and other components that make up a tree.

The following list describes the icons used to represent the structures contained in a tree similar to the one in figure 3.46:

 This icon defines the top of the tree.

 This icon identifies a category that is used to group similar data elements together at a high level. An example of a category in figure 3.46 is TOOLS.

 This icon represents a tree structure. A tree structure represents a hierarchy within a category. ACCESS_GROUP is an example of a tree structure in figure 3.46.

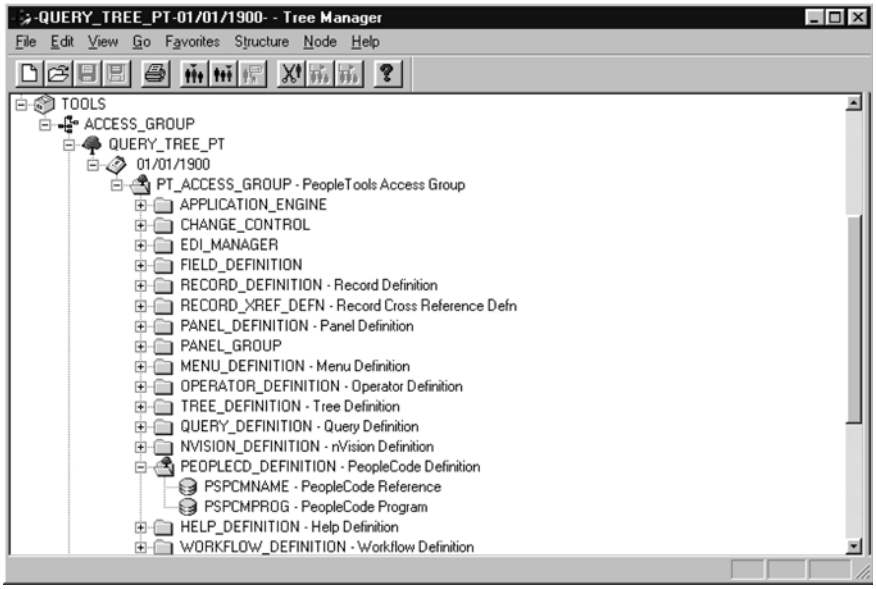


Figure 3.46 A tree and its components

A tree definition consists of a description, higher level category, use of levels section and status. Figure 3.47 illustrates the tree definition for QUERY_TREE_PT. The menu item to access this panel is Edit →Tree Definition.

SetID/Effective Date are the key fields in a tree definition. This information is required when a Tree is opened.

This icon identifies the Tree Branch. When a tree is branched, it is actually divided into two objects. One object is for the new branch and contains the remaining tree components. Within PeopleSoft, not every tree will require or contain branches. When a tree is branched, some improvement in efficiency may exist because less data are available to load. A more important application of tree branches is for security.

This icon represents a branch node.

This icon identifies an expanded node. In figure 3.46, the PT_ACCESS_GRP is an example of an expanded node. In the illustration, many nodes report to it.

This is also an expanded node in which levels are skipped.

This icon represents a collapsed node. When a node is collapsed, the box next to the node contains a "+," indicating nodes are reporting to the collapsed node

but are not displayed by Tree Manager. In figure 3.46 PANEL_GROUP is an example of a collapsed node.

This icon represents a collapsed node. (However, the levels represented by this icon are skipped.)

This icon applies to detail/summary trees.

The icon represents a record definition. In the example (figure 3.46) PSPCMPROG is a record definition. It can also be collapsed or expanded to reveal child nodes.

TIP When changes are made to the departmental security tree, a process is required to update the security profile for operator and operator classes. This process, called `PER505`, confirms that operators will have access to the modified tree structure.

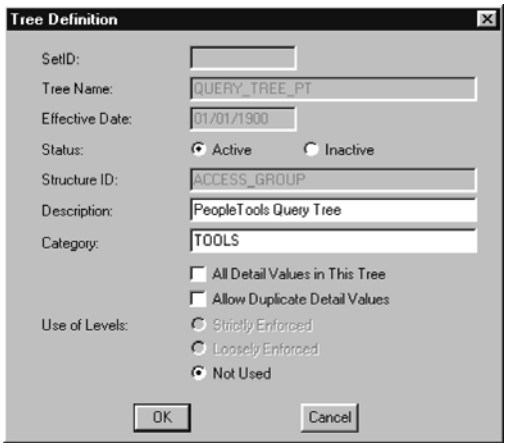


Figure 3.47
Tree definition for QUERY_TREE_PT

Tree can be categorized into four groups: Detail Trees, Summary Trees, Node Oriented Trees, and Query Access Trees.

Detail trees are considered the most basic type of tree. A detail tree rolls up to detail values comprised of tree nodes that combine the detail values and roll up to higher level tree nodes.

Summary trees enable us to re-arrange or re-group nodes from a detail tree. The detail tree itself is not copied in its entirety. Detail values in a summary tree are tree nodes from another detail tree, not values from database fields.

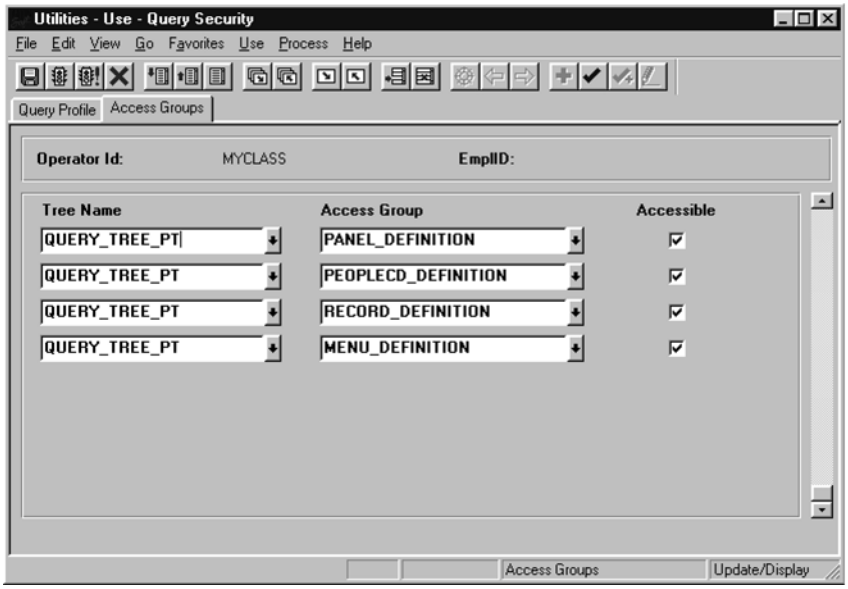


Figure 3.48 Query access trees used with query security

Node oriented trees contain nodes which depict database field values. Node Oriented trees can be used in applications such as HRMS Departmental Security.

Query trees are used in conjunction with PeopleSoft Query. Query trees combine database record definitions into entities identified as access groups. Access groups are used with the Query Security panel where we link an operator ID or class to one or more access groups. Each tree name can contain one or more access groups. A user cannot query records not contained in the access groups linked to their operator class. Figure 3.48 illustrates the use of query access tress with query security.

KEY POINTS

1. Data mover is a PeopleTool that is used to load and unload data from one environment to another. Data Mover can be used to load data across database platforms.
2. Import Manager is another tool that can be used to load data to tables defined by the Application Designer. Import Manager uses the record definition and an ASCII fixed file. The fields on the file are mapped to fields on the record.
3. Security Administrator is used to define Classes of Operators containing access to menus and menu items. An Operator Class is also linked to process groups defined by process scheduler.
4. Operator IDs can be linked to multiple operator classes and one class profile for row level security.
5. Object security is a tool that can be used to limit access to specific objects contained in an object group.
6. Tree Manager is a tool that can be used graphically represent departments within an organization or relationships between database tables.

PART 2

Application development

Application development in PeopleSoft begins with Application Designer. Once the requirements have been specified you can begin building your application in a logical sequence. Here's a simplified example of this sequence. First you define and create any custom field definitions you may need. The field definitions can then be utilized in a record. You can create online applications by placing these records with their associated fields on a panel. Panels can be placed in a panel group and added to a menu. Security access is assigned to operators who will utilize the new panel on the menu. If the new panel is used as a front-end to a batch process, a process definition is created and linked to the panel. These basic tasks can be accomplished very easily using Application Designer, Security Administrator and Process Scheduler. Additional enhancements, simple or complex, may be made as well. We'll demonstrate the use of these and other PeopleTools by building a Problem Tracking application, gradually introducing more advanced features as our development progresses. Some advanced panel design features include working with scroll bars, effective dates, sub and secondary panels and using grid objects. The key to online development in PeopleSoft is the Application Processor which dynamically governs data retrieval and event processing. PeopleCode, PeopleSoft's proprietary language, should always be written to fully exploit the capabilities of the Application Processor. By carefully following along with the Problem Tracking application you'll see all of these exciting features unfold before you. As I mentioned earlier, this is some very fun stuff!

CHAPTER 4

Building your first application

PeopleSoft uses a bottom-to-top methodology to build applications. A bottom-to-top method involves individually collecting all the components used to build higher level components, eventually arriving at a fully developed application. In this chapter, we look at how this methodology is used to design and develop an application.

4.1 IDENTIFYING THE APPLICATION

Our first step in application development is to collect development specifications. The user requesting the application may provide the initial specification. Then, the developer creates technical specifications for user review. To facilitate our discussion on basic functions used in collecting user specifications, we will develop a Problem Tracking application. Our Problem Tracking application will function within PeopleSoft and track both user-reported problems as well as developed solutions in a project life cycle. Hundreds of PeopleSoft implementations require such an application to efficiently track problems and resolutions. We'll build the underlying data model for our application in such a way that the project team can use a variety of search mechanisms to identify resolutions to new incidents. This application will be particularly useful as a production tool to provide customer support in a live environment. Note that, while endless possibilities exist for enhancing this application, we'll limit our discussion here to basic functions, leaving more advanced development concepts for a later review.

4.1.1 Fit/Gap analysis

We begin by identifying our business needs and then evaluating those needs against the PeopleSoft software application package. This process, called Fit Analysis, identifies functions in the PeopleSoft application that fit the business needs. It also helps identify any Gaps that the application package cannot accommodate. We identify Fits and Gaps in a Fit/Gap analysis document, which is then used to build our project plans and delineate development efforts toward successful implementation. Tools provided by PeopleSoft are so easy to use that even new applications can be built with relative ease. In our case, Problem Tracking is a brand new application not available in PeopleSoft. Hence, we need to gather business needs to develop the application from scratch. Let us look at the business functions that should be available in a common Problem Tracking application.

4.1.2 Gathering user requirements

In all the projects in which we have ever been involved, user specifications have proven to be extremely important. User specifications show us how the user sees the end result. Specifications are developed both by the technical and functional users. Several iterations of the specifications are exchanged back and forth between the technical and functional users, depending upon the complexity of the application. Technical users must assist the functional user in identifying the application fields that are utilized in a user panel or report. Specifications also help the developer to stay on track so far as the business needs are concerned and develop for those needs. Even though we won't discuss the stage where specifications are developed in this book, we cannot emphasize enough how important it is to develop with written specifications.

TIP Develop an application with user specifications. This will not only save time but will ensure that you are on the right track.

Let's suppose that our functional and technical users get together, discuss the business functions that should be available in the application, and arrive at the following requirements for the application:

- All problems reported should be categorized into a set number of applications such as Human Resources, Payroll, Benefits, Accounts Payable, General Ledger, and so on.
- All problems reported should identify the end user that reported them.
- All problems reported should have a status for tracking purposes.
- The problems and resolutions should be stored in a public domain so that all users have access to the resolutions.
- The application should be able to identify and prioritize commonly reported problems.
- The basic design of this application should facilitate problem tracking in new and future projects.

Figure 4.1 illustrates some inputs and outputs to and from our application.

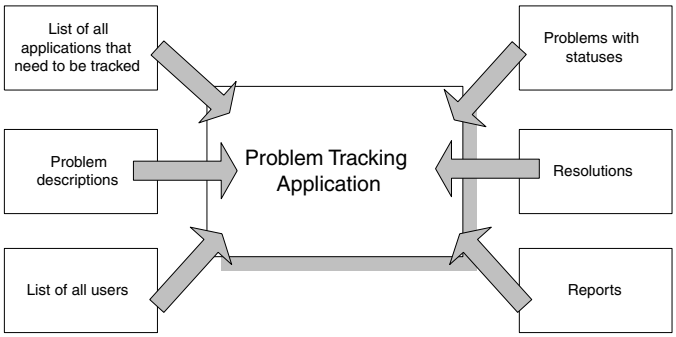


Figure 4.1 Problem Tracking application—Input/Output diagram

4.1.3 Identifying the objects to be developed

Now that we have compiled all the user requirements that we need to build our application, we can create a list of objects that need to be developed in PeopleSoft. How do we go about doing this? Data design is a term that comes to mind. Using PeopleSoft's integrated application development tool, Application Designer, we can build all the objects for our application by following these steps:

1. develop all record definitions that will be used in our application,
2. identify database keys for SQL tables and views,

3 develop all the user interface screens (otherwise known as panels) in PeopleSoft,
4 develop all the panel groups that provide a gateway to our application panels,
5 develop menu items that will hold the application panels and present them to the user, and
6 provide user access to menu items.

4.1.4 Prototype

While gathering user requirements, it is always useful to create a prototype of the application. A prototype allows us to walk the users through the application panels to give them an idea relative to data input and output. Prototypes are also used to gather additional user inputs on data elements missed during the Fit/Gap analysis. Prototypes help in creating final user specifications for development. Figure 4.2 shows the relationships between different objects in a PeopleSoft application.

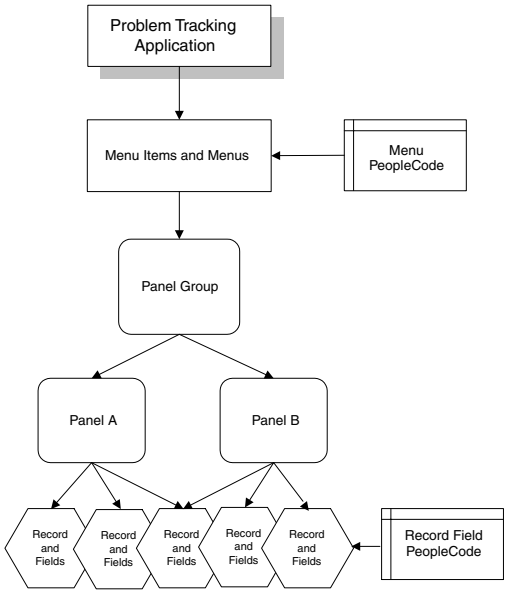


Figure 4.2
PeopleSoft Object Relationships

In this case, the Problem Tracking application is a new application in PeopleSoft. Here we ask the user to walk us through the current system. If this process can be performed during the Fit Analysis stage, then the prototype stage is skipped, and we proceed to the design, development, and unit testing stages.

As you can see in figure 4.2, design is usually performed starting from the top (represented by our application). On the contrary, development is performed from bottom up with objects assembled individually to achieve the end result. The bottom then refers to all the individual objects, and the top refers to the application. First, we

start from the top, designing our application's presentation to the user including all user interface objects such as panels and menus. Then we design and develop the Record definitions, views, PeopleCode, and panel groups, which are building blocks for the user interface. The panels and the panel groups, together with application menus, will provide the user interface to our Problem Tracking application.

We can now start building development specifications. These specifications are incorporated into the project time line in a project plan document. During stages in projects, new problems arise and these problems can be tracked using our application. As mentioned before, we will use the Application Designer tool to develop our application.

4.2 USING THE APPLICATION DESIGNER

Application Designer, is a comprehensive design tool used to build applications in PeopleSoft. Ideally, we want to build all the objects related to this application into a PeopleSoft Project. A Project is a collection of PeopleSoft objects developed to serve a common function. In our case, the purpose of developing objects is to build a Problem Tracking application. Therefore, we collect all objects we develop and include them in one project. Let us take a look at the Application Designer screen before we can start our development (figure 4.3).

Navigation: Go →PeopleTools →Application Designer

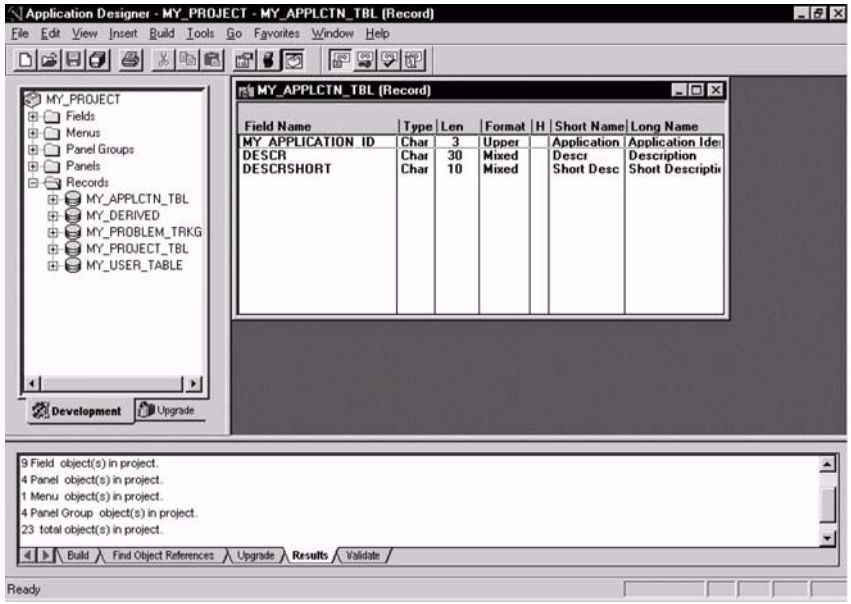


Figure 4.3 Application Designer screen

The left side of the Application Designer screen is called the Project Workspace. When we open projects (*Navigation:* File →Open →Project), all objects in the project are displayed in the Project Workspace. The right side, the Object Workspace, is the actual development area. The bottom tab portion of the screen serves as an output window for showing development progress, object reference searches, and so forth. For example, if we search for object references in the whole database (*Navigation:* Edit →Find Object References), the output appears in the bottom window. In figure 4.3 we can see that a project is already open. The objects in MY_PROJECT are all displayed in the Project Workspace. We can also see the number of objects in our project in the Output Window. The right side is used for development of these objects. Let's go through the process of developing these objects, inserting them into MY_PROJECT as we finish.

The icons that we see in the Application Designer menu bar are all short cuts used for various tasks. When we move the mouse pointer over these icons, a tab appears, denoting the tasks these icons perform. Let's look at all the icons available in the Application Designer screen.

4.2.1 General icons

- *New*—creates new objects.
- *Open*—opens existing objects.
- *Save*—saves the current object. In other words, if we have multiple objects open and we are working on one particular object, only the current object that we are working on will be saved when we choose the Save button.
- *Save All*—saves all objects that are open in the Application Designer screen.
- *Print*—prints the current object.
- *Cut*—deletes and stores the selected item in the clipboard.
- *Copy*—copies the selected item into the clipboard.
- *Paste*—pastes the copied item into the selected area.
- *Properties*—brings up the Properties window for the current object.
- *Build Current*—helps build SQL tables and views in the database. (We will learn more about this in chapter 8.)
- *Project Workspace*—is a toggle to hide or show the project workspace in the Application Designer screen.

4.2.2 Record display icons

These icons appear only when a record definition is open and is the current object in focus:

- *Field Display*—displays all record fields, their attributes, short name, and long name.
- *Use Display*—displays all the search keys, list box items, and defaults for the record definition.
- *Edits Display*—displays all the edits for the record definition.
- *PeopleCode Display*—displays all record fields and their associated PeopleCode events.

4.2.3 Panel design icons

These icons are available only when a panel definition is open and is the current object in focus:

- *Select Group*—selects a group of panel fields for cut, paste, and move operations.
- *Order*—displays the Order panel that is used to change the order of panel fields.
- *Test*—is a toggle that switches between test and design mode.
- *Object Inspector*—is a toggle that hides or shows the Object Inspector tool.
- *Panel Size*—controls the panel sizing properties.
- *Grid*—is a toggle that hides or shows a grid in the Object Workspace.
- *Label Position*—moves the field label to its default position.

- *Text*—adds a static text field into the panel.
- *Frame*—adds a frame into the panel.
- *Group Box*—adds a group box into the panel.
- *Static Image*—adds a static image into the panel.
- *Edit Box*—adds an edit box into the panel.
- *Drop-Down List*—adds a drop down list into the panel.
- *Long Edit Box*—adds a long edit box into the panel.
- *Check Box*—adds a check box into the panel.

- *Radio Button*—adds a radio button into the panel.
- *Image*—adds an image into the panel.
- *Scroll Bar*—adds a scroll bar into the panel.
- *SubPanel*—adds a subpanel into the panel.
- *Push Button*—adds a push button into the panel.
- *Secondary Panel*—adds a secondary panel into the panel.
- *Tree Control*—adds a tree control into the panel.
- *Grid Control*—adds a grid control into the panel.

4.2.4 Panel group icons

- *Insert Panel*—inserts a panel into the panel group.
- *Validate Panel Group*—validates the panel group.

If we double-click on any object that appears in the Project Workspace, the object is opened in the Object Workspace. When we click on the object using the right mouse button, a standard pop-up menu appears showing tasks that can be performed on the object. We can customize the Application Designer by accessing Options from the View menu. We can also customize or resize components of the Application Designer by dragging the edges of Project Workspace, Object Workspace, and Output Window using the mouse.

And, as we also noted under the general icons, the Project Workspace icon is a toggle that shows or hides the Project Workspace.

4.3 CREATING FIELD DEFINITIONS

Let's start by looking at the schema—e.g., the structure—for the first record in our application. We will call this record MY_USER_TABLE and start here for simplicity reasons:

```
Record Name     MY_USER_TABLE

Field Names     MY_USER_ID
                NAME
                EMPLID
                PHONE
                MY_USER_TYPE
```

We have five fields in this record. The first and the last fields are new fields we need to create before we can start building our Record definition. The other three fields are already available in a PeopleSoft HRMS application. Fields in PeopleSoft are defined, as well, not within the scope of a record, but globally throughout the entire database.

Navigation: File →New (From Application Designer)

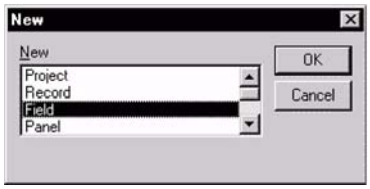


Figure 4.4 Create a new object

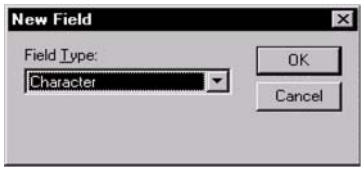


Figure 4.5 Selecting the field attribute

Fields are objects and they can be shared across record definitions. For this purpose, let us suppose that we are working in a PeopleSoft HRMS application. First, we create a new field (figure 4.4).

We choose `Field` as the new object type. Then we choose `Character` as the field type in the New field window as illustrated in figure 4.5.

In the Field Attributes screen (figure 4.6) we specify the field length, long name, and the short name for the field. We also specify the field format as `Uppercase`. We associate the field with a particular field format by defining the family name and the display name for the format (figure 4.5).

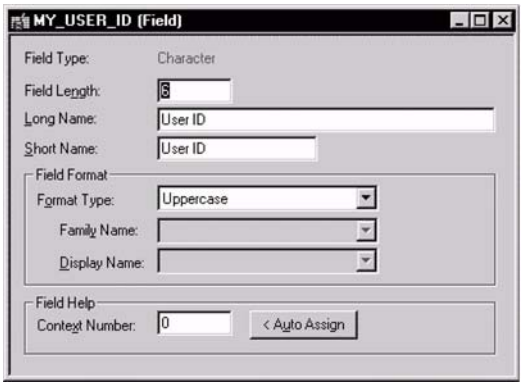


Figure 4.6
Field Attributes Screen

NOTE Fields are individual objects. When field attributes are changed, they are changed throughout the system. The same field can be used across many record definitions.

Field Help Context Number is a number we can use to associate the field with a help text in a Windows-based help file. We click on Auto Assign to automatically assign the next available Context Number from the system. WINHELP can be used to create a Windows-based help file. Since PeopleSoft has reserved context numbers up to 10,000,000, we must use numbers higher than this to associate our fields with Windows-based help.

We can also create the MY_USER_TYPE field by following the same instructions. Let us see how numeric, date, and other types of fields are created in PeopleSoft. In order to do so, let us take a look at other fields available in our application. The navigation is the same except that the correct field type has to be chosen as illustrated in figure 4.5.

PRIORITY is a number field that we will use in MY_PROBLEM_TRKG table. Since this field exists in a PeopleSoft HRMS system, we are going to make use of this field in our record definition. Take a look at the field attribute for this field (figure 4.7).

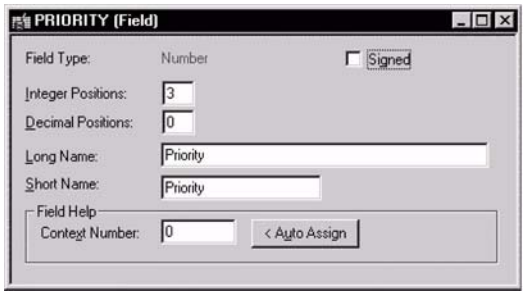


Figure 4.7
Number and Signed Number—Field Attributes screen

We have to specify the Integer and Decimal positions for number fields. The attributes screen is the same for both Number and Signed Number field types except that the Signed checkbox is turned on for a signed number.

Figure 4.8 illustrates the attributes defined for MY_PROBLEM_RESOLTN field, a long character field. We define a maximum length for the long character field by entering a '0' to use the maximum length that the database can accommodate. This can be anywhere from 2000 to 64000 characters, depending upon the database platform. We also define whether the field is to be stored in Raw Binary or Text format. Raw Binary can be used to store embedded `Nulls` in our field.

Figure 4.9 illustrates a Date field in PeopleSoft. For this purpose, we use the INCIDENT_DT field in our application.

In order to resolve Y2K issues, PeopleSoft has presented all dates with inclusive century dates. If the user enters a date without the century, the Default Century

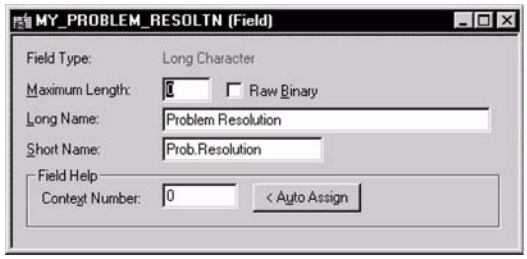


Figure 4.8
Long Character Field Attributes

attribute comes into play. We can enter a number here that determines the century for the date. If a century is not entered, and the two-character year is greater than the number specified in this box, the Application Processor uses the current century for the date; otherwise, it defaults to the next century.

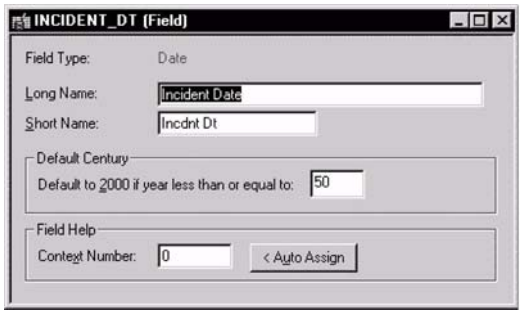


Figure 4.9
Date Field Attributes screen

Figure 4.10 illustrates attributes defined for a Date/Time field in PeopleSoft. The MY_PROBLEM_DTTM field used in our application is a Date/Time field.

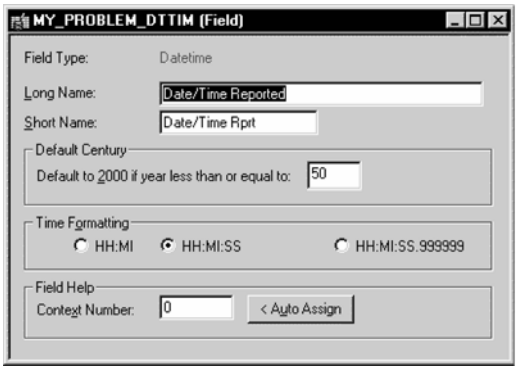


Figure 4.10
Date/Time Field Attributes screen

The Date/Time field attributes are the same as the Date field attributes. The Date/Time field has an additional attribute for the time part. "HH" refers to the hour portion of time, "MI" refers to the minute portion, "SS" refers to the second portion; and "999999" refers to the subsecond portion. We choose "HH:MI:SS" for MY_PROBLEM_DTTM field.

Let us take a look at a Time field in PeopleSoft. Notice in figure 4.11 that the attribute screen only has the time attributes.

By the end of this section, we should have created all the custom fields needed for our application. We can accomplish this task by building a schema for all records

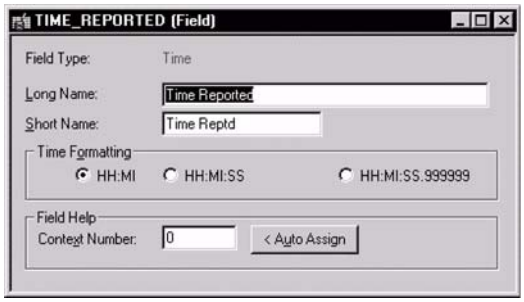


Figure 4.11
Time Field Attributes screen

that will be used in our application. Since fields are building blocks used to build Record definitions, it is convenient to have all the fields in the system ready for inclusion when records are built. Another way of accomplishing this task is to add the new fields when building the Record definition. By creating a schema for all records in our application, we can easily identify the custom fields as opposed to the fields delivered in a PeopleSoft system.

Now that we have started creating custom objects in the system, it is only logical to create a Project that will hold all the objects used in creating our application.

4.4 WORKING WITH PROJECTS

A project is a collection of objects used to develop an application or a subsystem. It's a good habit to save into a project all objects that belong to an application or subsystem. This ensures that the project is complete for application upgrade to other databases. In our case, we are developing a whole new application, and all the objects which we create for our application will be collected and stored as a project. We just created all our custom fields, so now it's time to save them to a project.

By default the Application Designer screen always displays an untitled project upon startup. We had previously saved a number of field definitions. These fields can be inserted into the untitled project either by pressing F7 or choosing "Insert/Current Object into Project" from the Application Designer screen. (We use this last option to insert objects currently open in the Application Designer screen.)

Alternatively, we can bring up a list of objects and then insert them into the project. We can accomplish this by pressing CTRL-F7 or by choosing "Insert/Objects into Project" from the Application Designer screen. We can also insert a whole project into our current project by choosing "insert/Projects into Project" from the Application Designer screen.

Finally, we save this Project by choosing the Save icon from the Application Designer screen. We are then prompted to name the project as illustrated in figure 4.12.

We use MY_PROJECT to insert the other objects we create for our application. Let's take a look at how we define project properties using the Application Designer.

Navigation: File →Save Project from Application Designer

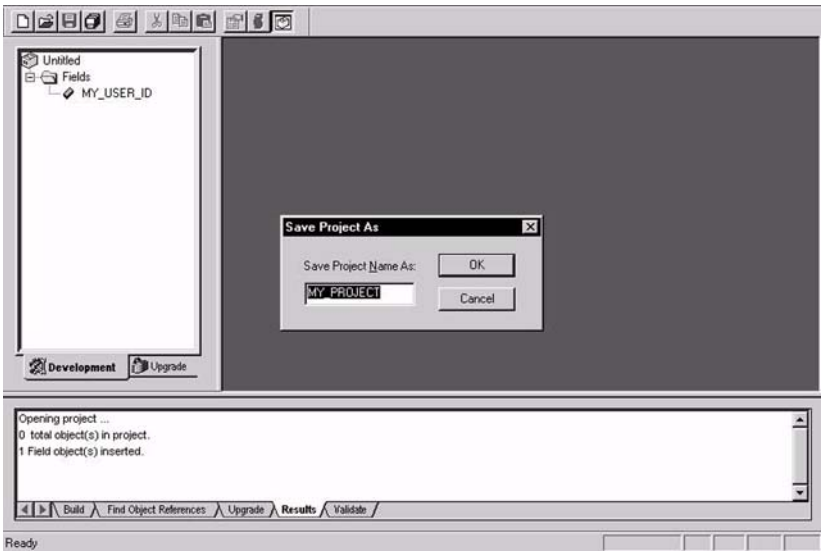


Figure 4.12 Saving a project

The Project Properties window has three different tabs: the General, the Report filter, and the Copy Options (figure 4.13).

Navigation: File →Project Properties (MY_PROJECT is open)

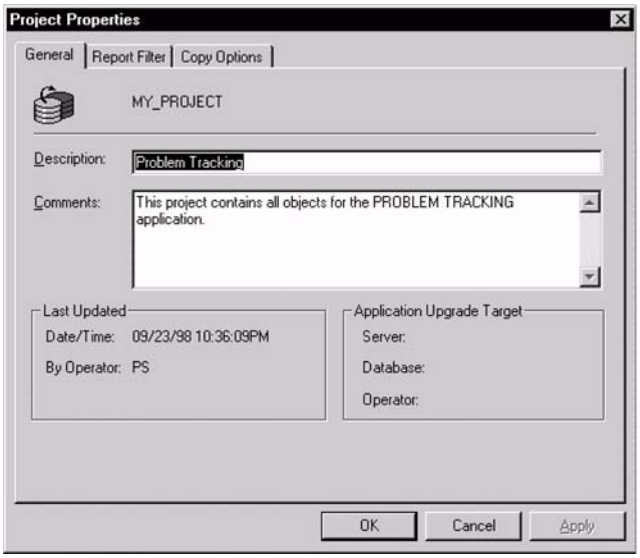


Figure 4.13 Project Properties screen, General tab

4.4.1 General

Under the General tab, we specify a description for the project. The Comments section is used to maintain a log of changes to the project. The LAST UPDATED section displays the date and time when the project was last updated as well as the ID of the operator who last updated the project.

4.4.2 Report Filter

During a major application or tools upgrade, all objects identified as custom objects during the `Upgrade Comparison` process are tagged as `CUSTOM`. We can use the Application Upgrader tool to migrate objects from one database to another. The `Upgrade Comparison` process is a delivered PeopleSoft process used to compare objects between two databases. For example, development and production databases can be compared to identify newly developed objects not in the production database. The `Upgrade Comparison` process uses a database link between two databases. (We will describe more about the `Upgrade Compare` process in chapter 20.)

4.4.3 Copy Options

The Copy Options tab defines parameters for the `Upgrade Copy` process in PeopleSoft. After database objects are compared, the target database can be upgraded to match the source database by running the `Upgrade Copy` process (see chapter 20).

After defining Project Properties, click on the Apply push button to save the properties. MY_PROJECT is complete so far as the properties are concerned. We can add more objects to this project, and the Project Properties will still apply to all the objects in the project.

After saving the project, all objects in the project appear on the left side of the Application Designer screen.

TIP All objects used in developing an application can be inserted into a project. The project as a whole can be migrated to other databases.

NOTE All objects in PeopleSoft contain a Date/Time stamp for their last update as well as the ID of the operator who updated them.

4.5 CREATING A PEOPLESOFT RECORD DEFINITION

A record definition is a collection of fields that defines data storage in the database and data presentation online. Record definitions can be categorized as tables, SQL views, Derived/Work records, Subrecords, Dynamic views, and Query views. We will discuss the different types of record definitions more later in this section. For our application we will create one Derived/Work record. Derived/Work records are only relevant to the online application. They exist only in PeopleTools and not in the database. Derived records, used as temporary holding spaces during panel execution,

can also be used to hold fields that are counters, calculation fields, or command fields. In our case, we will use the Derived record field as a command field to open a document. In chapter 6 we will add more fields to this Derived record to show how the derived fields are used as counters and calculated fields. First, let's consider in detail the steps necessary to create a PeopleSoft record definition:

- create a schema
- identify and create custom fields
- create a record definition
- define Record Definition properties
- define Record Field properties
- perform Data Administration, if necessary

4.5.1 Create a schema

Remember, a schema acts as a structure and is developed before SQL tables and views are built. Similarly, we can create a schema for each record definition used in our application. Our application is built using individual objects. In this case, these individual objects are fields, which are assembled into record definitions using the schema. Listed immediately following are the fields that we will use to build the record definitions for our application:

```
Record Name    MY_USER_TABLE
Field Names    MY_USER_ID
               NAME
               EMPLID
               PHONE
               MY_USER_TYPE

Record Name    MY_PROJECT_TBL
Field Names    MY_PROJECT_ID
               DESCR
               MY_APPLICATION_ID
               START_DATE
               END_DATE
               CONTACT_NAME
               CONTACT_PHONE

Record Name    MY_APPLCTN_TBL
Field Names    MY_APPLICATION_ID
               DESCR
               DESCRSHORT

Record Name    MY_PROBLEM_TRKG
Field Names    MY_PROBLEM_ID
               INCIDENT_DT
               MY_PROJECT_ID
               MY_PROBLEM_STATUS
               PRIORITY
```

```
                    MY_USER_ID
                    MY_PROBLEM_TRACKER
                    CLOSE_DT
                    MY_DOCUMENT_ATTACH
                    DESCRLONG
                    MY_PROBLEM_RESOLTN
                    MY_PROBLEM_DTTIM

Record Name         MY_DERIVED
Field Names         MY_DOCUMENT
                    MY_USER_ID
```

4.5.2 Identify and create custom fields

We have two custom fields in the first record definition. We identify these custom fields by the prefix "MY," which we use to identify the custom objects in our application. In order to build the first record, we first create the two new fields by accessing the Application Designer "File" menu and choosing "New" option. Then we choose `Field` as the object type. Once we have created these fields in the database, we can start building our record definition. We create field definitions by using the techniques described in section 4.3.

4.5.3 Creating a record definition

The PeopleSoft catalog table which stores record definitions is called PSRECDEFN. SQL tables and SQL views, which are record definitions, are database objects as well. SQL tables store permanent data in the database, and SQL views helps us in presenting this data in different ways. For this simple reason, SQL tables and SQL views are the only built-in types of record definitions within the database.

To create a record definition, we start by clicking on the New icon from the Application Designer screen and choosing `Record` as the object type. A new screen appears where we can insert the necessary fields.

Figure 4.14 illustrates a blank record definition. We insert fields into this record definition by first selecting the key fields, which are always the first fields in the field order.

Navigation: File →New →Record

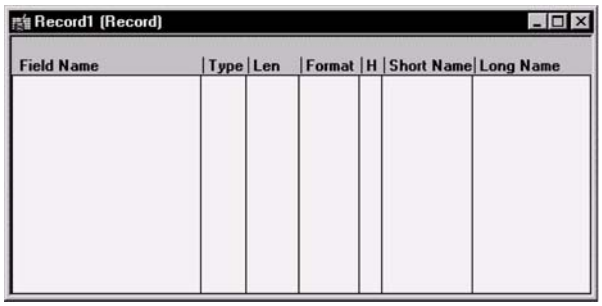


Figure 4.14
New Record screen in the Application Designer

Figure 4.15 illustrates the screen used to insert fields into record definitions. We can either type in the field name or search for the field name by entering the first few characters of the field name and clicking on the Insert push button. When we specify the full field name, the field is inserted into the blank Record screen (figure 4.14). When we search for a field by providing the first few characters of the field name, all field objects matching the selection criteria appear in the bottom. We then highlight the correct field and click on Insert to include the field to the record definition.

Navigation: Insert →Field (Blank Record Definition is open)

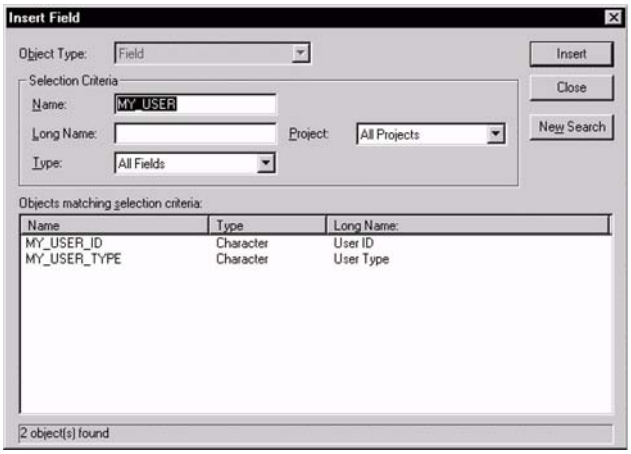


Figure 4.15
Inserting a field into a record definition

TIP Fields are placed next to each other when we start from a blank record screen. When inserting a field into an already existing record definition, determine the field that will be the previous field after the new field is inserted. Highlight that field, then proceed with the insertion. The new field will be placed right after the highlighted field.

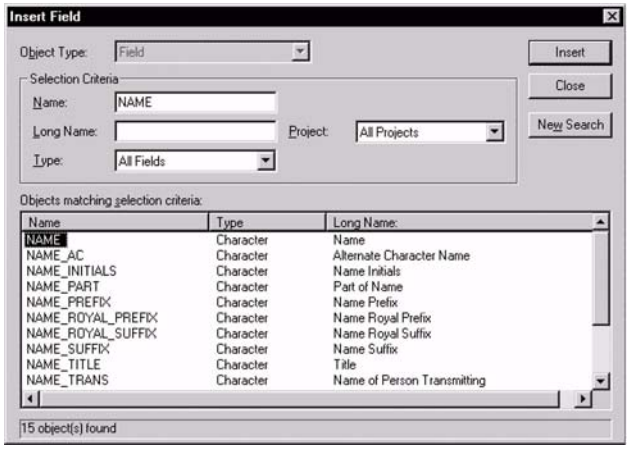


Figure 4.16
Inserting a field into a record definition

After all fields are inserted (figure 4.16), choose Close to finish the selection, then File/Save from the Application Designer screen to save, the record definition. During save time we will be prompted for a record name so name this record MY_USER_TABLE. Notice that we have used a prefix of MY_ to identify that the record definition is a custom one. The record definition for MY_USER_TABLE appears as illustrated in figure 4.17.

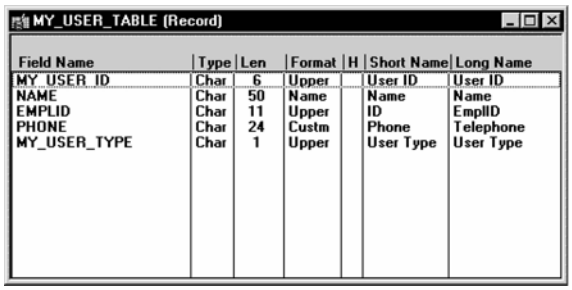


Figure 4.17
A saved record definition screen

Now we need to insert this record definition into our project. We accomplish this by choosing "Insert/Current Object into Project" from the Application Designer menu. Since we have added all the fields that comprise this record, we now define the record attributes required to complete the record definition.

TIP Under the Tools/Options menu, we can specify default options for a project, including a setting for inserting objects into the project. We can choose the option that results in the automatic insertion of an object after the object is modified and saved. As an alternative, the manual insertion option gives us better control regarding objects inserted into the project.

4.5.4 Defining record definition properties

We can also bring up the Record Properties screen illustrated in figure 4.18 by pressing ALT-ENTER from the keyboard. The record definition itself should be open at this time. We complete defining Record Definition properties by entering parameters in the three different tabs—the General tab, the Use tab, and the Type tab—available in the Record Properties screen.

Navigation: File →Object Properties (MY_USER_TABLE is open)

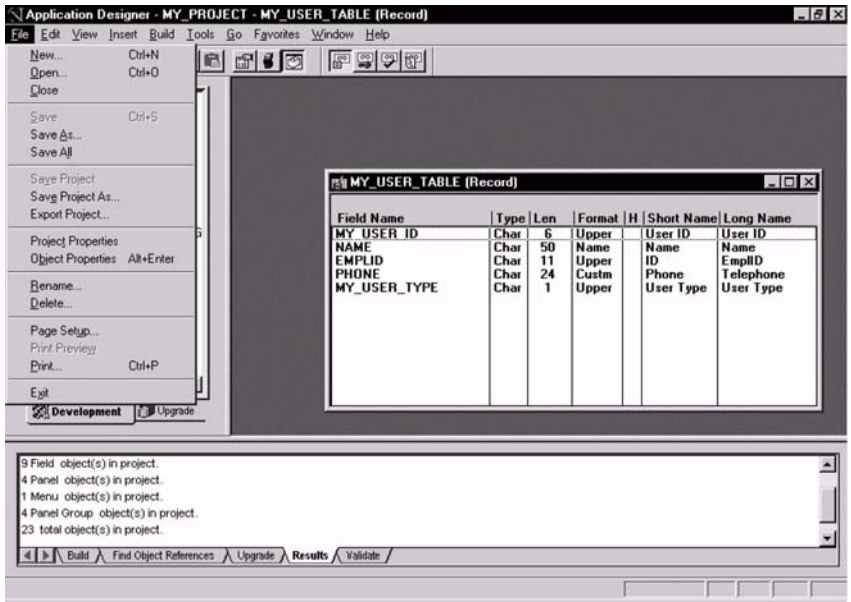


Figure 4.18 Record properties screen in Application Designer

General tab

In figure 4.19 we can see the General tab where we describe the record definition. We can also maintain a log of changes made to the record definition here in the Comments section. Notice that the Last Updated section helps us identify the date and

time the record definition was last updated as well as the ID of the operator who last updated it.

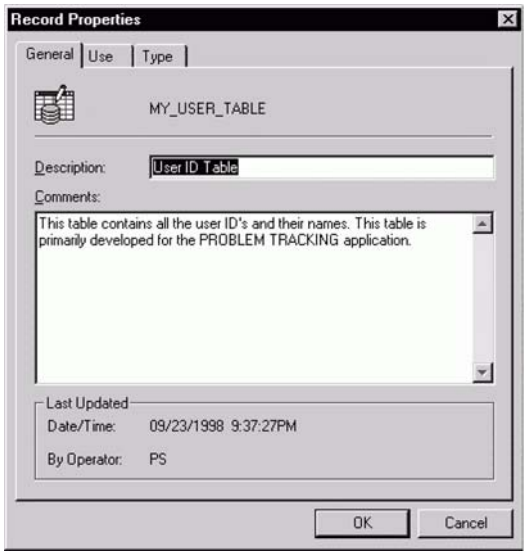


Figure 4.19
Record definition properties—General tab

Figure 4.20 illustrates the Use tab under the Record Properties screen.

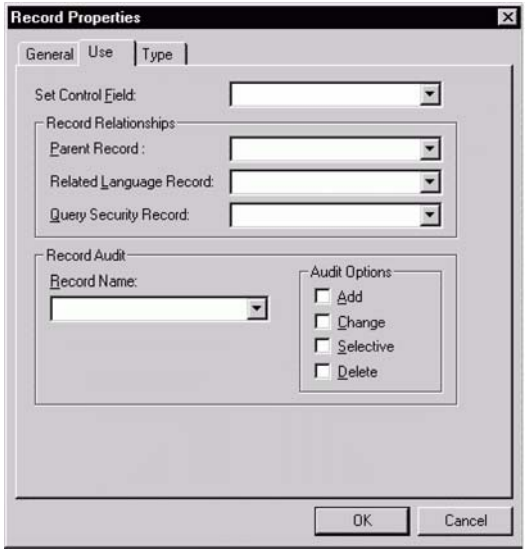


Figure 4.20
Record definition properties—Use tab

Use tab

Let's go through the properties defined in the Use tab. Although we won't be using these features in MY_USER_TABLE, let's quickly walk through these features and briefly explain their usage.

Set Control field The Set Control field is used in a multi-company environment. You can use Set Control field to share this particular table across companies. For example, MY_USER_TABLE can be shared between companies by simply adding SETID to the MY_USER_TABLE definition. The SETID field is a delivered field in PeopleSoft. This field has to be both the first field in the record definition and a search key. We can specify another field from the system as the Set Control field. The value of the Set Control field will be compared to the value of SETID field from MY_USER_TABLE. This comparison controls the selection of data from MY_USER_TABLE. Organizations with multiple companies can use the same record definition we create here to access user definitions for other applications in the PeopleSoft database. So COMPANY field can be the Set Control field that will control the display and access of user definitions from MY_USER_TABLE.

Parent record In the Parent record, you define a record definition that can serve as a Parent record to MY_USER_TABLE. Parent/Child relationships help create automatic hierarchical joins using the PS/Query tool delivered in PeopleSoft. For example, in a PeopleSoft HRMS application, the PERSONAL_DATA record is the Parent record for EMPLOYMENT. Common search keys between these two record definitions define the Parent/Child relationship between them.

Related Language record The Related Language record is used in a multilingual PeopleSoft application. Let's use MY_PROJECT_TBL as an example to better explain this process. Suppose that, MY_PROJECT_TBL is accessed from offices all over the world, and access to this table must be provided in multiple languages. We can create a Related Language record that stores project information in different languages. The Related Language record contains all search keys from the primary record definition as well as LANGUAGE_CD field as an additional key. All other fields entered and stored in multiple languages are also included in the record definition. As a user logs into the application, the user options define the Base Language Code for the user. This allows the user to enter language-specific information stored in the Related Language record. This function is most commonly used in Human Resources applications where users from different countries access employee information in different languages.

Query Security record The Query Security record provides row level security for record definitions in PS/Query. When a Query is created using a record definition, the record definition is automatically joined with the Query Security record to secure the data that the user can view. For example, access to personnel information in a

PeopleSoft Human Resources database can be controlled using PERS_SRCH_GBL as a Query Security record.

Record audit The Record Audit facilitates auditing online updates to data stored in a record definition. By turning on the audit flags in a record definition, we can record all inserts, updates, and deletes. You can either specify a record name defined as a database table to hold the audit information, or if you do not specify a record name and audit flags are turned on, PeopleSoft stores the audit information in a generic audit table called PSAUDIT. Four audit flags may be activated. Each initiates that audit based on user action:

- *Add* triggers the system to populate the audit table when new rows are added to the audited table.
- *Change* triggers the system to populate the audit table when any row in the audited table is changed.
- *Selective* triggers the system to populate the audit table when any one of the fields in the audited table is changed.
- *Delete* triggers the system to populate the audit table when any row in the audited table is deleted.

When we create our own audit table, we have to add three relevant fields to the audit table. They are AUDIT_OPRID, AUDIT_STAMP, and AUDIT_ACTN fields. In addition to these fields, any other field in the audited record definition can be added to the audit table, but the default PSAUDIT table captures enough information required for an audit.

MY_USER_TABLE does not contain any properties defined under the Use tab.

Type tab

Record Type The Record Type tab (figure 4.21) is where we define whether the record definition is an SQL table, SQL view, Dynamic view, Derived/Work, Sub-record, or a Query view. MY_USER_TABLE will be defined as an SQL table.

SQL View Select statement On the right hand side of the properties screen, we define the SQL `Select` statement for an SQL view definition. (We will learn more about SQL views in chapter 7 when we define an SQL view for use in inquiry panels.)

Non-standard SQL Table Name Through the non-standard SQL Table Name option, we give the record definition a non-standard name, one that differs from the normal PeopleSoft standard. PeopleSoft prefixes all tables with a "PS_" prefix. We can override this standard by entering a non-standard SQL table name. All PeopleTools tables follow a non-standard naming convention. The PeopleTools tables and views with non-standard names are referred without the "PS_" prefix. The name still starts with "PS" to identify it as a PeopleSoft catalog table. For example, PSRECDEFN is a PeopleSoft catalog table where record definitions are stored. We will set the record type as an SQL table for our MY_USER_TABLE definition.

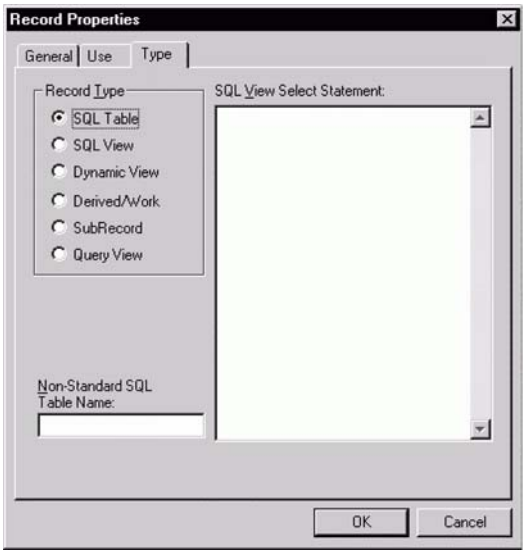


Figure 4.21
Record definition properties—Type tab

4.5.5 Define record field properties

The next step in creating a record definition is to define record field properties. Record field properties contain search key, list box, audit, and default information for a particular record field. We click on Edit/Record Field Properties to define properties for all the fields in our record. Record Field properties apply to fields only within the purview of this record definition. When used in other record definitions, these fields do not share the properties defined here. The record field must be highlighted before we can edit record field properties. The record field properties screen contains two tabs: the Use and Edits tabs.

NOTE Properties defined for a record field apply to the field only within the purview of that record definition. The same field used in another record definition will not necessarily share the same properties.

Use tab

Keys Figure 4.22 illustrates the Use tab under the Record Field Properties screen. Using the following options, a list of keys can be defined for the record field:

- *Key* defines a record field as a database key only.
- *Duplicate Order Key* defines a record field as a duplicate order key in the database.
- *Alternate Search Key* defines a record field as an alternate search key in PeopleSoft. Alternate Search Keys are automatically selected as list box items. A non-unique index is created in the database for all alternate search keys.

- *Descending Key* makes a database key a descending key. Usually the EFFDT and EFFSEQ fields are defined as descending keys to view the latest effective-dated rows in the top of the scroll bar.
- *Search Key* defines the record field as a search key. All fields defined as search keys are also defined as list box items. When the record definition is used as a search record, all Search and Alternate search keys appear on the input dialog box.
- *List Box Item* defines the record field as a list box item. When search key values yield multiple rows, the values of all list box item fields appear in the list box.
- *From Search Field* defines the record field as a From Search field. The search process yields all entries that have values greater than or equal to the value supplied in this field.
- *Through Search Field* defines the record field as a Through Search field. The search process yields all entries that have values less than or equal to the value supplied in this field.

Navigation: Edit →Record Field Properties from the Application Designer

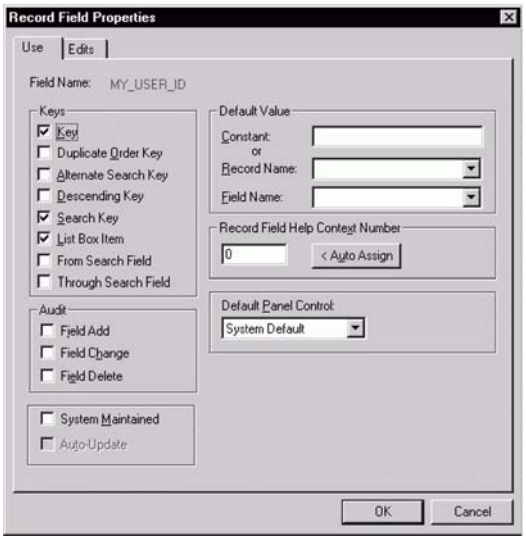


Figure 4.22
Record Field Properties screen—Use tab

We define MY_USER_ID field as a key, a search key and a list box Item. We define NAME and EMPLID fields, from MY_USER_TABLE as alternate search keys. This automatically defines them as list box items as well.

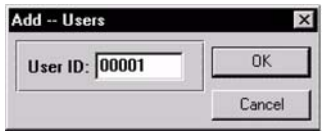


Figure 4.23 Input Dialog box during Add

Let's look at the Input dialog box for our application. Figure 4.23 illustrates the input or search dialog box that appears when users access MY_USER_TABLE using our application. Notice that we are adding users here, hence we have to specify a value for the MY_USER_ID field defined as a search key.

In figure 4.24, you can see how alternate search key fields and list box items appear when we try to access the application panel online using Update/Display action. The list box provides users with values of key fields before they choose the item that they want to view and edit.

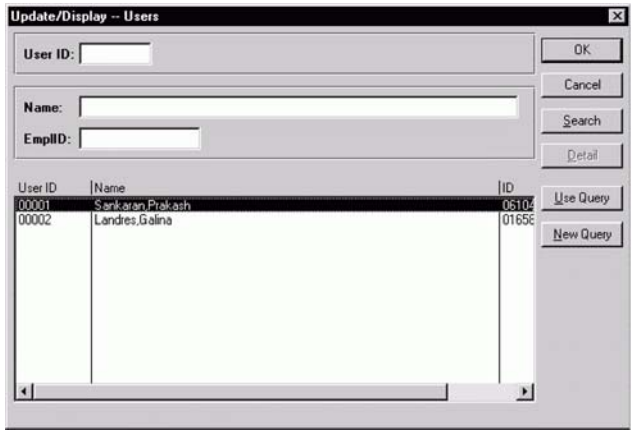


Figure 4.24 Input dialog box and list box during Update/Display

We'll discuss From and Through search keys in greater detail in chapter 6.

Default Value We can specify defaults in the Record Field Properties screen (figure 4.22) for a particular record field. We can either use a constant as the default value or assign a value from another record field. Default values are populated into a field only when the field is blank or zero. We can also use the `FieldDefault` PeopleCode event to provide default values.

NOTE Default values specified in record fields are processed first before `FieldDefault` PeopleCode is executed.

Record Field Help Context Number The Record Field Help Context Number is similar to the Field Help Context Number. The help text can be summoned when the record field is accessed online. (The Help text explains the usage of a field within a record definition.) The Help Context Number associates this record field

to a Windows-based Help text. As we mentioned previously, PeopleSoft reserves Help Context Numbers up to 10,000,000, so we must use a number higher than 10,000,000. We can also assign this number automatically by using the Auto Assign button.

Default Panel Control The Default Panel Control controls the appearance of the record field on a panel. The options available here are Edit Box, Drop-Down List, and System Default. If we choose System Default, the system selects the default panel field type for this record field.

Audit Changes to the record field can be audited online using the Audit option. When the Field Add option is chosen, any new values entered in record fields are audited. When the Field Change option is chosen, any changes to record fields are audited. When the Field Delete option is chosen, all deletes are audited. All audit information is populated into a delivered audit table called PSAUDIT. The audit record name can be overridden by specifying an audit record attached to record definition properties (figure 4.20).

System Maintained The System Maintained field is used for reporting purposes only. We turn on this checkbox if we want the record field to be updated by the system.

Auto Update The Auto Update option is used for Date/Time fields. The system updates the record field to the current date and time. Any user-changes to a field defined for Auto Update will be overwritten with the current date and time. We can use this option for the MY_PROBLEM_DTTIM field in the MY_PROBLEM_TRKG record definition to automatically store the system Date/Time into that field.

Edits tab

After the options under the Use tab are completed, we can proceed to the Edits tab under the Record Field Properties Screen (figure 4.25).

Navigation: Edit →Record Field Properties from the Application Designer →Edits

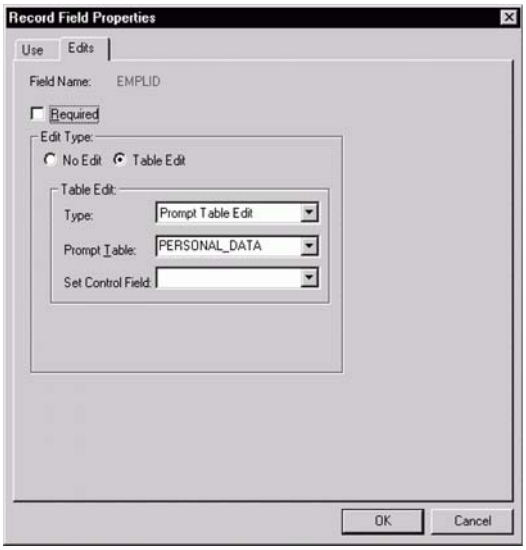


Figure 4.25
Record Field Properties screen, Edits tab

Required This option makes the field a "Required" field, meaning that the user has to enter a value into this field on an application panel that contains this record field. We will make MY_USER_ID a required field in MY_USER_TABLE.

Edit Type The Edit Type option edits the value entered in this field. If we specify No Edit, then the field is not edited. If we choose the option Table Edit, then we have to specify the edit type. Four different types of edits are associated with a record field.

- *Prompt Table Edit* The record field is attached to a prompt record that has the list of valid values. The user is not allowed to choose any value outside the list of valid values. EMPLID field in MY_USER_TABLE uses PERSONAL_DATA as the prompt record. (To learn more about Prompt Processing, refer to chapter 6.)
- *Prompt Table with No Edit* This is the same as Prompt Table Edit except that this option allows the user to enter values not in the list of valid values.
- *Translate Table Edit* This option edits the value entered into this field against the XLATTABLE. Translate values are attached to field definitions. These translate values are activated for the particular record by using this option. If this option is not chosen, translate values attached to a particular field definition is not used in editing. MY_USER_TYPE field has translate values that are activated for MY_USER_TABLE record definition. (To learn more about translate values, refer to chapter 6.)
- *Yes/No Table Edit* This is used for fields that contain a `Yes` or `No` value. Usually record fields defined as checkboxes in panels use this option.

In figure 4.25, you can see the prompt record attached to the EMPLID field in MY_USER_TABLE. Figure 4.26 illustrates the translate values attached to MY_USER_TYPE field in MY_USER_TABLE.

Navigation: View →Translate from the Application Designer Menu

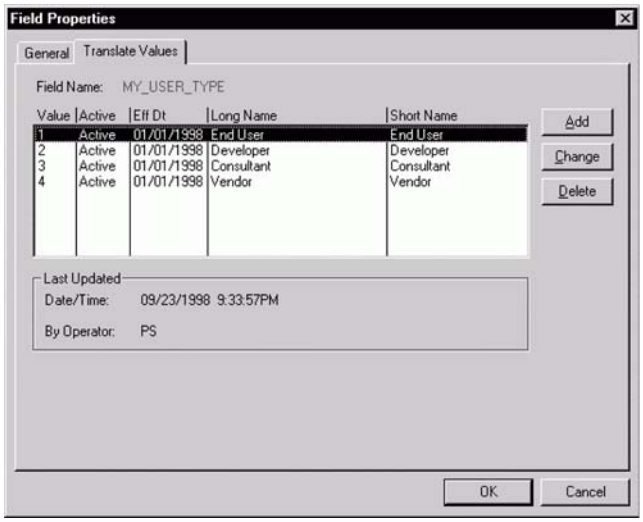


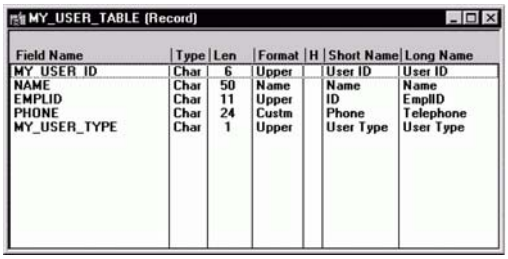
Figure 4.26
Translate values

Set Control Fields Set control fields can be used only with prompt table edits. The set control field specified here will override the set control field defined for the record definition containing this field. The value of the set control field will be used in prompt processing. The prompt record contains the SETID field and the value in the set control field on the panel will be compared to the value of the SETID field from the prompt record.

In PeopleSoft, Record Field Properties are shown in four display formats: Field Display, Use Display, Edit Display, and PeopleCode Display. Figures 4.27 through 4.30 show all four display views as well as a brief description of their content.

Field Display shows the field type, field length, field format, short name, and long name for the record fields.

Navigation: View →Field Display (MY_USER_TABLE is open)

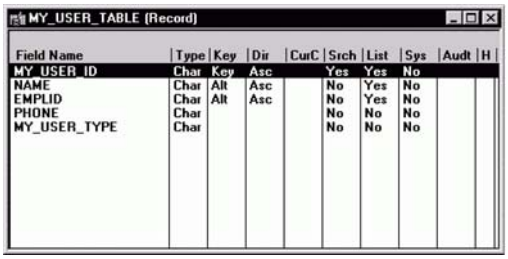

MY_USER_TABLE (Record)

Field Name	Type	Len	Format	H	Short Name	Long Name
MY_USER_ID	Char	6	Upper		User ID	User ID
NAME	Char	50	Name		Name	Name
EMPLID	Char	11	Upper		ID	EmplID
PHONE	Char	24	Custm		Phone	Telephone
MY_USER_TYPE	Char	1	Upper		User Type	User Type

Figure 4.27
Record definition, Field Display

Use Display shows all the properties we entered under the Use tab using the Record Field Properties screen.

Navigation: View →Use Display (MY_USER_TABLE is open)

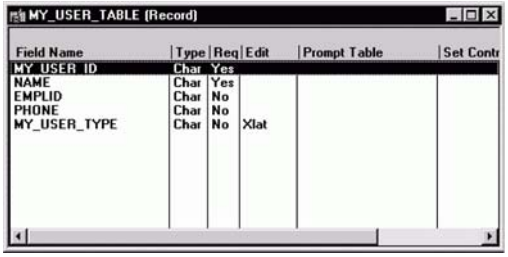

MY_USER_TABLE (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H
MY_USER_ID	Char	Key	Asc		Yes	Yes	No		
NAME	Char	Alt	Asc		No	Yes	No		
EMPLID	Char	Alt	Asc		No	Yes	No		
PHONE	Char				No	No	No		
MY_USER_TYPE	Char				No	No	No		

Figure 4.28
Record definition, Use Display

Edits Display shows all properties we entered under the Edits tab using the Record Field Properties screen.

Navigation: View →Edit Display (MY_USER_TABLE is open)


MY_USER_TABLE (Record)

Field Name	Type	Req	Edit	Prompt Table	Set Contr
MY_USER_ID	Char	Yes			
NAME	Char	Yes			
EMPLID	Char	No			
PHONE	Char	No			
MY_USER_TYPE	Char	No	Xlat		

Figure 4.29
Record definition, Edits Display

PeopleCode Display shows all Record Field PeopleCode in the record definition. (We will discuss adding PeopleCode to record fields in part 3.) A record field which contains a PeopleCode program attached to one of its PeopleCode events will show a `Yes` against the PeopleCode event. In figure 4.30, you can see a PeopleCode program attached to the `FieldChange` PeopleCode event of the EMPLID field.

Navigation: View →PeopleCode Display (MY_USER_TABLE is open)

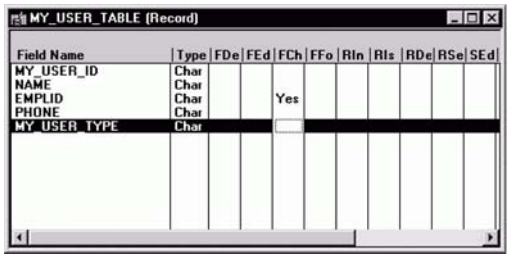


Figure 4.30
Record Definition, PeopleCode Display

4.5.6 Perform Data Administration

Data Administration, a precursor to building an object in the database, is performed in the database. SQL tables and SQL views are database objects and must be built in the database. We use record definitions specified as SQL tables and SQL views to do so. Database objects are built using scripts that use the Data Definition Language (DDL) Model defaults. DDL is used to create data dictionaries in relational databases. It is good practice to generate scripts to build objects and save the scripts in your dictionary of DDL scripts. Not only will this practice prove useful if the record definitions have to be rebuilt in the database, it also provides a record of all DDL statements executed in the database.

Build objects

Let's look at how we build SQL tables and SQL views using record definitions. Figure 4.31 illustrates the screen in the Application Designer tool used to build database objects.

When the Build →Project option is chosen, all records in the project are included in the build scope list. In our example, we open MY_USER_TABLE through the Application Designer, then choose Build →Current Object from the menu.

In the Build screen, we choose the `Create Tables` action. MY_USER_TABLE is defined as an SQL table and included in build process. We choose the Execute SQL Now option for MY_USER_TABLE. MY_USER_TABLE is a new object, therefore, it does not yet exist in the database as a table. When we choose the `Create Tables`

Navigation: Build →Project or Current Object

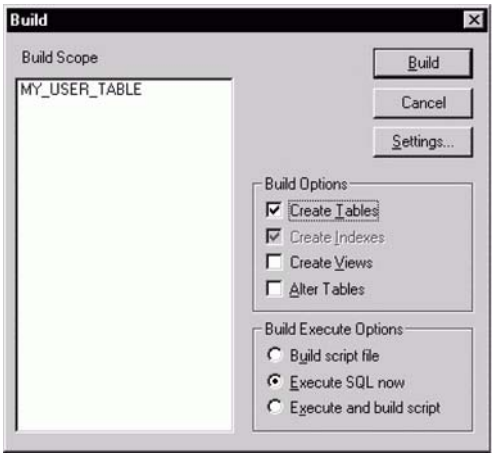


Figure 4.31
Building an object in the database

action, the Create Indexes option is also turned on automatically. This creates all indexes for PS_MY_USER_TABLE in the database.

NOTE SQL tables and views are prefixed with the letters PS_. The MY_USER_TABLE record is called PS_MY_USER_TABLE in the database for example. This unique prefix helps the database administrator identify PeopleSoft tables and views easily.

We will learn more about building SQL tables, views, and indexes in chapter 8.

4.6 CREATING A PEOPLESOFT PANEL DEFINITION

Panel design is a crucial step in building an application. Panels have to be designed to facilitate easier and faster input online. While record design is important for storing data and for data integrity, panel presentation is the primary evaluation factor for the online application. To build an application panel, we need to perform the following steps:

- assemble record fields in the panel.
- define panel field attributes.
- check the panel layout.
- define panel properties.
- save the panel

Let's start by building a simple presentation panel to enter users through the Problem Tracking application.

4.6.1 Assembling record fields in the panel

We open our project and choose File →New from the Application Designer menu. We can then choose `Panel` as the object type.

Figure 4.32 shows the blank panel that we use to assemble record fields. We begin by placing fields in the order of input, starting with the highest level key from the record definition. Since we are building a panel to enter users for our application, we start by choosing fields from MY_USER_TABLE.

Navigation: File →New →Panel

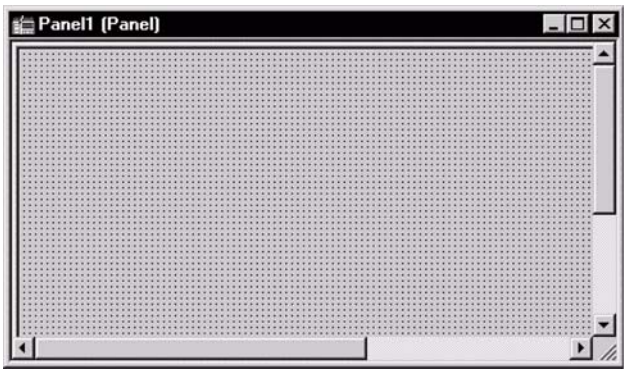


Figure 4.32
Panel designer screen in Application Designer

We insert fields into our blank panel by choosing Insert from the Application Designer menu and choosing the correct panel field type. The first field in MY_USER_TABLE is the MY_USER_ID field that is an Edit Box field.

We choose the panel field type based on how the field will be used functionally. In this example, we either add or update user IDs through our panel. The EMPLID field will be a drop-down list panel field type simply because we attached a prompt record to this field. Because MY_USER_TYPE has translate value attached to it, it can be defined as a drop-down list panel field type. NAME and PHONE are Edit Box panel field types. Let us take a look at how the panel appears with all the five fields from MY_USER_TABLE placed in input order.

In figure 4.33, we have defined MY_USER_ID field as a display-only field. This is because MY_USER_ID is used as the search field to access the panel. EMPLID and MY_USER_TYPE fields have a drop-down arrow indicating that they are drop-down list fields. This means that, by simply clicking on the drop-down arrow, a list of valid values will be presented to the user.

In addition to the five different fields, a Static Text field exists in the panel. This is the example telephone number format shown next to the PHONE field in figure 4.33.

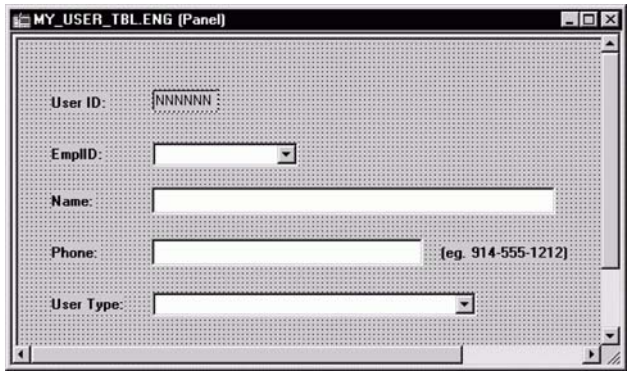


Figure 4.33
Assembled record fields on a panel

4.6.2 Define panel field properties

After placing the fields in a blank panel, we then define panel field properties. Default panel field properties can be changed using three different tabs: the Record, the Label, and the Use tabs. (See figure 4.34.)

To change the Panel Field properties for each panel field, highlight the panel field and bring up the Panel Field Properties screen either by choosing Edit →Panel Field Properties from the Application Designer menu or pressing CTRL-F from the keyboard.

Record tab

Navigation: Right Click on the panel field →Panel Field Properties

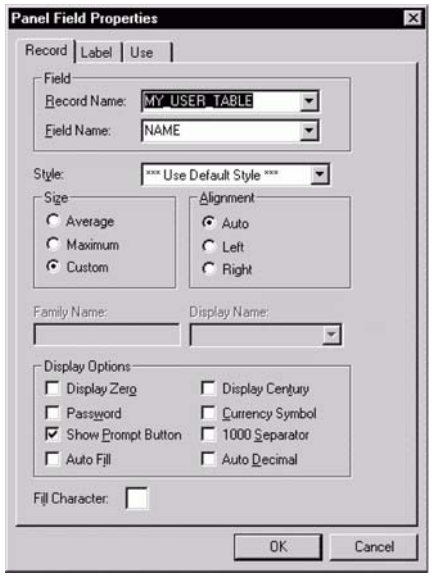


Figure 4.34 Panel Field Properties screen—Record tab

The Record tab (figure 4.35) defines the source, size, and display options for our panel field. The record name is MY_USER_TABLE, and the field name is NAME (figure 4.34). We can also control the size of panel fields under this tab. The NAME field is defined as `Custom`. This enables us to adjust the size of the field on the panel. Panel field size can be adjusted by using the right corners and dragging them to the left to reduce the size or to the right to increase the size.

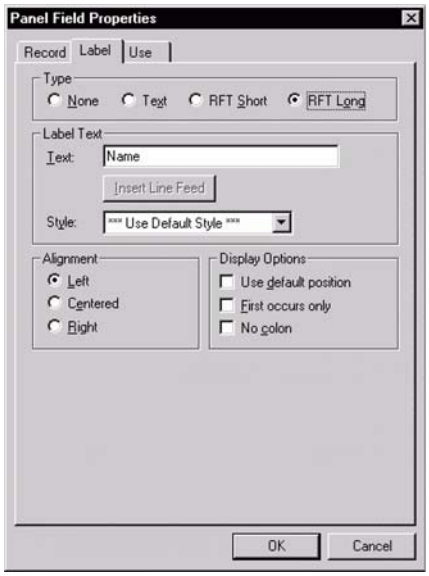


Figure 4.35 Panel Field Properties Screen—Label tab

Label tab

The Label tab (figure 4.35) defines the properties for panel field labels. There are two components to a panel field: one is the panel field itself, and the other is the label for the panel field. Let's take a look at the Label tab for the panel field EMPLID from MY_USER_TBL panel.

In figure 4.35, we have defined the Label type to use the `Long` name from EMPLID field definition. We can also choose to use `Short` name, or `Text` as the panel field label or define the panel field not to use any label. Panel field labels can be aligned to the panel field itself by using the Alignment option.

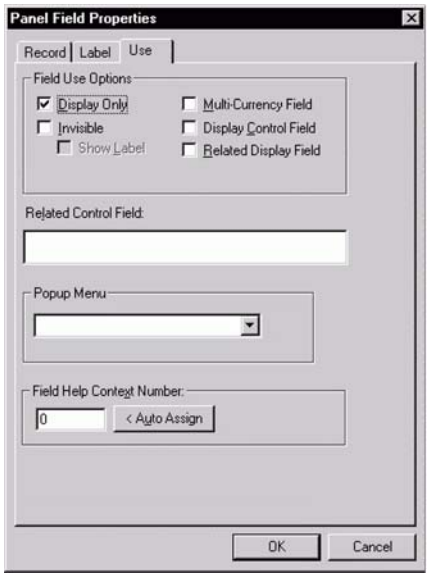


Figure 4.36 Panel Field Properties screen—Use tab

Use tab

The Use tab defines the panel field usage properties. Figure 4.36 illustrates the Panel Field properties for the EMPLID field from MY_USER_TBL panel.

Look back to figure 4.33. Notice that MY_USER_ID field is a display-only field in the panel. We were able to define that under the Use tab for that panel field.

For more about defining Panel Field Properties, see chapter 7.

4.6.3 Checking the panel layout

Once all the fields are laid out on a panel, and the panel field properties have been defined, it is time to check the layout of the fields in the panel. Checking the layout of fields helps:

- check the input order of fields
- check if all key fields are assembled first before other non-key fields
- check if all Related Display fields are placed after their respective Display Control fields (For example, if EMPLID is the Display Control field and NAME is the Related Display field, we have to make certain that EMPLID is before the NAME field in the panel field layout.)
- (if there is a scroll bar in the panel) be sure all fields inside the scroll bar are from one record definition except for Related Display and Derived fields
- make certain that key fields that facilitate prompts on prompted fields in the panel are before the prompted fields themselves. (For example, in a PeopleSoft HRMS application, there is a table called PAY_CALENDAR that contains payroll calendars to process payrolls. This table has three key fields: COMPANY, PAYGROUP, and PAY_END_DT fields. In order to prompt on the PAY_END_DT field, values for COMPANY and PAYGROUP must be available to facilitate the prompt.)

When we save the panel, these checks are enforced, and a message appears if the panel is invalid. We can always change the layout of fields if the panel is invalid at save time.

4.6.4 Define panel properties

Navigation: File →Object Properties

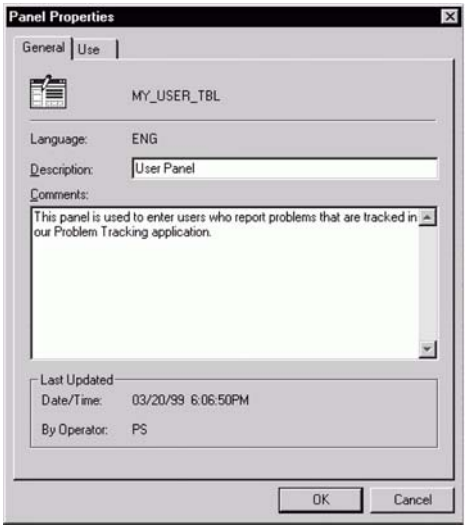


Figure 4.37 Panel Properties screen—General tab

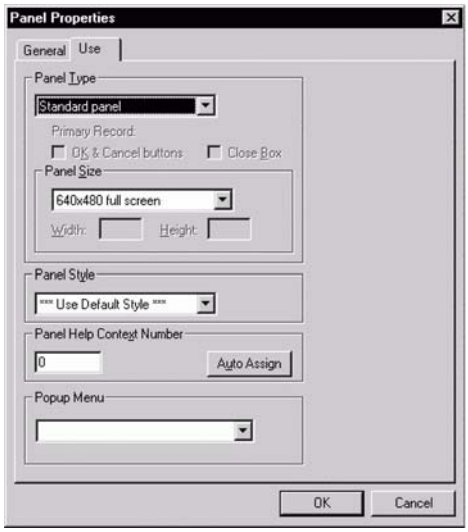
The Panel Properties screen can be brought up either by choosing File/Object Properties from the Application Designer menu or right-clicking on any one of the panel fields and choosing Panel Properties from the pop-up menu. Two different tabs exist in the Panel Properties screen: the General tab and the Use tab. Figures 4.37 and 4.38 illustrate the two tabs in the Panel Properties screen.

General tab

The General tab displays the language, a brief description, comment, and last updated date/time/operator ID for the panel. PeopleSoft allows language versions of panels. Panels can be stored in languages other than English. The

Comments section can be used to maintain a log of changes to the panel. All PeopleSoft objects maintain the last updated date/time and the ID for the operator that last updated the object.

Use tab



The Use tab contains the panel type, panel size, panel style and pop-up menu attributes. The panel can be defined as a standard panel, SubPanel or a secondary panel. Panel size attributes control the panel resolution. We can design the panel to suit VGA or SVGA resolutions. We can define a style for the panel. The whole panel will inherit the style defined here. Panel Help Context Number is used to link the panel with online help. A pop-up menu can be attached to the panel. When the user right clicks anywhere in the panel, the pop-up menu is activated.

Figure 4.38 Panel Properties screen—Use tab

4.6.5 Saving the panel

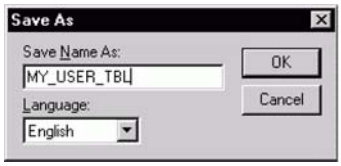


Figure 4.39 Save a panel

Our final step is to save the panel definition. We can save the panel by choosing File/Save from the Application Designer menu or by clicking on the Save tool bar icon. Figure 4.39 illustrates the screen brought up to save the panel definition.

We enter a name for the panel and click on OK to save the panel. The Language drop-down list box allows us to save panels in other languages for multilingual access. The base language appears on this drop-down list. Panel definitions can be saved in languages other than English by logging into the system as a user whose base language is the language in which you want to save the panel.

KEY POINTS

1. PeopleSoft development is performed from bottom-to-top (i.e., PeopleTools objects are developed to reach the top—the fully developed application of sub-systems).
2. User specifications and technical specifications are extremely useful for the development process.
3. The Application Designer tool is an integrated tool delivered with PeopleSoft applications to help in application development.
4. Projects are used to include all objects used to develop an application or sub-system.
5. Fields are individual PeopleSoft objects.
6. Record can be specified as SQL tables, SQL views, Derived/Work records, Query Views, Dynamic Views and Subrecords. Only SQL tables and views are built in the database as objects.
7. Field attributes are the same across all record definitions. Record field attributes apply only to a particular record definition.
8. A panel can be a standard panel, a subpanel, or a secondary panel.
9. Secondary panels are used to organize fields by their function. Subpanels are used to separate repetitive fields, like address fields into a subpanel. Both subpanels and secondary panels can be included in a standard panel.

CHAPTER 5

Providing user access to the application

We have created all objects required to build the user interface and provide access to our application. (Note, that panel groups and application menus can be termed as user interface in PeopleSoft.)

A panel group can have one or more panels in it. Menu items provide user access to functions available under an application menu. Panel groups are attached to menu items. While providing access to a particular Menu Item, any one or all of the panels in the panel group can be chosen. It is important to also note that other aspects of security exist apart from the menu items that this chapter will discuss.

5.1 CREATING PANEL GROUPS IN PEOPLESOFT

PeopleSoft panels are attached to application menus using panel groups. Prior to PeopleSoft version 7, panel groups were part of the menu definition, but starting from PeopleSoft version 7, panel groups are separate objects which nonetheless serve the same purpose of linking panels to application menus.

Panel groups contain one or more panels. They can be created to make panels look more organized. Instead of crowding a panel with fields, multiple panels can be created and attached to a panel group. Panel groups can also be created to organize fields by function. JOB DATA in the PeopleSoft HRMS application is a classic example of one such panel group. Panel groups can be shared across menu items as long as the menu items share the same panel. To create a panel group, we move through the following steps:

1. create a new panel group
2. insert panels into the panel group
3. define panel group properties
4. save the panel group definition

5.1.1 Create a new panel group

We create a new panel group by choosing File/New from the Application Designer screen and choosing panel group as the object. Figure 5.1 shows the blank Panel Group screen that results.

Navigation: File →New →Panel Group

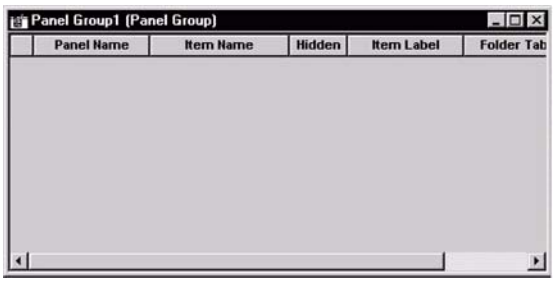


Figure 5.1
New Panel Group screen

5.1.2 Insert panels into the panel group

We start by adding an already-built panel into our new panel group. We insert a panel into the panel group by choosing Insert/Panel from the Application Designer screen. Figure 5.2 illustrates this process.

All panels that match the selection criteria are displayed in the screen. Once we finish inserting all the required panels into the panel group, we close the Insert Panel screen

Navigation: Insert →Panel into Group

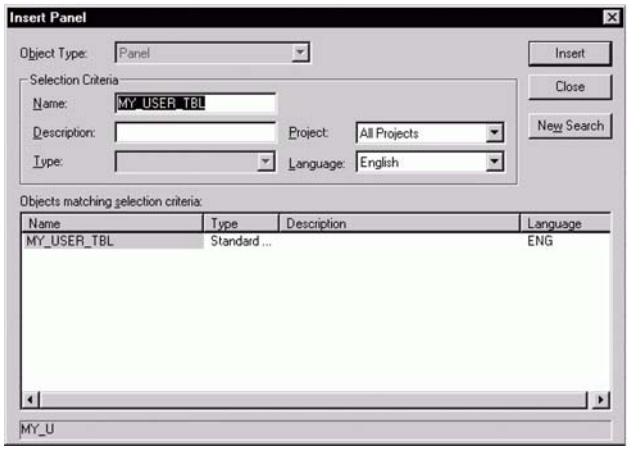


Figure 5.2
Insert panel into panel group

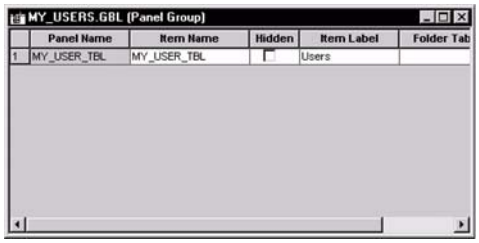


Figure 5.3 Panel inserted in a panel group

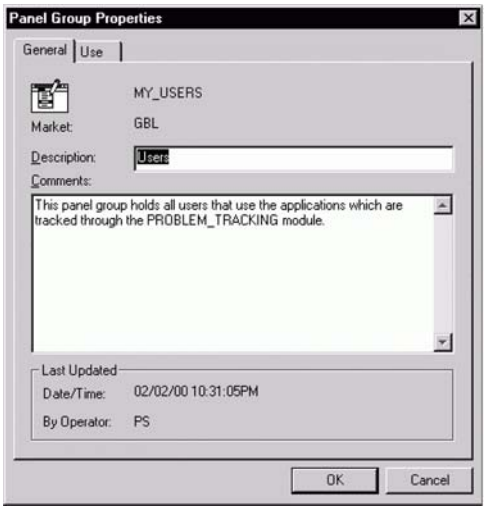
by clicking the Close button. Figure 5.3 illustrates the screen as it appears after the panel is inserted into the panel group.

Item Name and Item Label can be changed to fit the functional description of the panel group. The Hidden flag is used to hide panels in a panel group, a useful option when more than one panel exists in the panel group and one of those panels must be hidden from user access. The Folder tab is the label for the folder after the panel group has been brought up.

5.1.3 Define Panel Group properties

Before we save the panel group definition, we must define the properties for the panel group. The Panel Group Properties window is brought up by choosing Edit/Object Properties from the Application Designer menu. In figures 5.4 and 5.5, you can see the two tabs under the Panel Group Properties window.

General tab



Under the General tab, we enter a brief description and comments for the panel group. The Comments section is used as a modification log for the panel group. The last updated Date/Time as well as the ID of the operator who last updated the panel group is also displayed in the General tab.

Figure 5.4 Panel Group Properties—General tab

Use tab

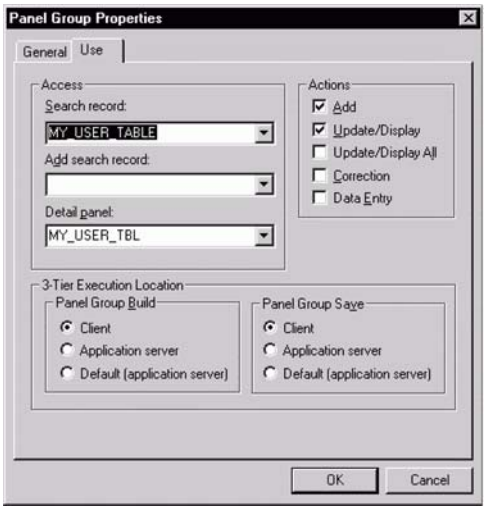


Figure 5.5 Panel Group Properties—Use tab

The Use tab holds the `Search Record`, `Add Search Record`, `Detail Panel Name`, `Actions`, `Build`, and `Save` locations for the panel group. In our example, we define MY_USER_TABLE as the search record. Because we won't specify an Add Search Record, MY_USER_TABLE will be used for `Add` action as well. Authorized actions are `Add` and `Update/Display` for our panel group. `Update/Display All` and `Correction` actions are used for panel groups which access effective-dated record definitions. When using the Tuxedo Application Server in a 3-tier environment, the `Panel Group Build` and the `Panel Group Save` locations come into play. These two parameters determine where the panel group is built and saved during online access.

5.1.4 Save the panel group definition

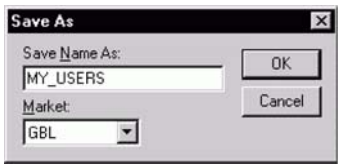


Figure 5.6 Panel Group save

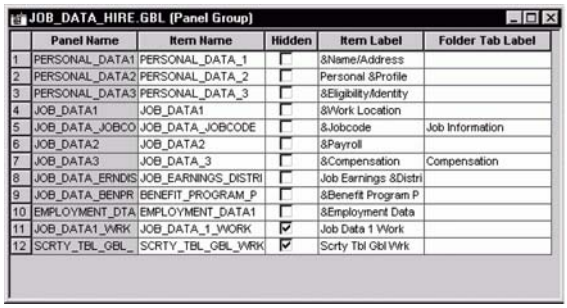
We can now save the panel group definition by choosing File/Save from the Application Designer menu. We will be prompted to enter a name for the panel group (figure 5.6).

The Market field is used to provide custom functionality to a panel group. The same panel group can be saved for two different markets that serve two different functions. For example, in a PeopleSoft HRMS application, the New Hire process can provide the same functionality but with subtle differences in different countries. We can create one panel group using different Markets to suit those subtle differences. Essentially, the two panel groups are two different objects that serve a common function but in different ways.

Let's look at a panel group which contains more than one panel. For this purpose, we use the JOB_DATA_HIRE panel group from the PeopleSoft HRMS system. This panel group has several panels which use the same search record, but display portions of data from tables separated by functional areas. This allows the user to access fields separated by function. Multiple records are also updated through this panel group. The SQL tables updated using this panel group are PERSONAL_DATA, JOB, JOB_EARNS_DIST, BEN_PROG_PARTIC, and EMPLOYMENT. These individual panels contain fields from these records. The user is able to save all the information for an employee in the database.

In figure 5.7, you can see the number of panels attached to the JOB_DATA_HIRE panel group. Notice that two of the twelve panels are hidden. These are used to hold Work/Derived fields used for calculation and panel processing. The user does not see the hidden panels in the application menu. The Item Label is displayed as a subitem under the Menu Item and as folder tab labels unless folder tab Labels are filled.

Navigation: File →Open →Panel Group →JOB_DATA_HIRE


JOB_DATA_HIRE.GBL (Panel Group)

	Panel Name	Item Name	Hidden	Item Label	Folder Tab Label
1	PERSONAL_DATA1	PERSONAL_DATA_1	☐	&Name/Address	
2	PERSONAL_DATA2	PERSONAL_DATA_2	☐	Personal &Profile	
3	PERSONAL_DATA3	PERSONAL_DATA_3	☐	&Eligibility/Identity	
4	JOB_DATA1	JOB_DATA1	☐	&Work Location	
5	JOB_DATA_JOBCO	JOB_DATA_JOBCODE	☐	&Jobcode	Job Information
6	JOB_DATA2	JOB_DATA2	☐	&Payroll	
7	JOB_DATA3	JOB_DATA_3	☐	&Compensation	Compensation
8	JOB_DATA_ERNDIS	JOB_EARNINGS_DISTRI	☐	Job Earnings &Distri	
9	JOB_DATA_BENPR	BENEFIT_PROGRAM_P	☐	&Benefit Program P	
10	EMPLOYMENT_DTA	EMPLOYMENT_DATA1	☐	&Employment Data	
11	JOB_DATA1_WRK	JOB_DATA_1_WORK	☑	Job Data 1 Work	
12	SCRTY_TBL_GBL_	SCRTY_TBL_GBL_WRK	☑	Scrty Tbl Gbl Wrk	

Figure 5.7
JOB_DATA_HIRE panel group definition

5.1.5 Panel groups and process definitions

Process definitions are used to identify batch processes in PeopleSoft, which are executed using the Process Scheduler. Just as menu item definitions are attached to panel groups, so are process definitions. When the user chooses the Run icon (Traffic Light) from an application menu, the Application Processor attempts to match the panel group from the application menu item with Process definitions which contain the same panel group. All matching process definitions are then presented to the user in a list on a Process Scheduler Request panel. A panel group, therefore, is the common link between a process definition and a menu item definition.

(To learn more on how to attach Process definitions to application menus, refer to chapter 27.)

5.2 CREATING APPLICATION MENUS IN PEOPLESOFT

Application menus serve as a gateway to the online application. Application menus can either be data entry panels, inquiry panels, or process panels. Panels that deliver common functions are usually linked to the same application menu. For example, all panels related to setting up payroll tables are linked to a menu called Define Payroll Process in a PeopleSoft Human Resources application. To create an application menu, we

- create a new menu definition
- create new bar items
- create new Menu Items
- define Menu Item Properties
- define Menu properties
- Save the menu definition

5.2.1 Create a new menu definition

Navigation: File →New →Menu

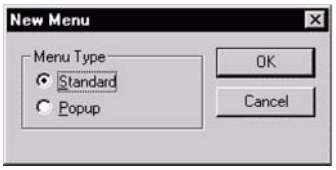


Figure 5.8 New menu

We create a new menu definition as illustrated in figure 5.8.

Standard menus are application menus while pop-up menus are linked to a panel or a panel field. We choose "Standard" for our application menu, bringing up a blank Menu Definition screen (figure 5.9).

Navigation: File →New →Menu →OK

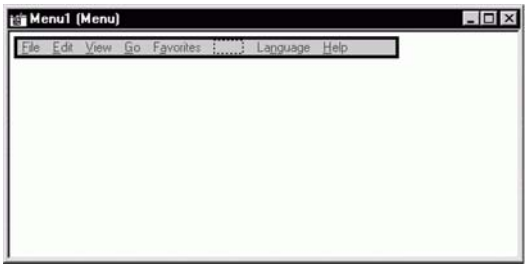


Figure 5.9
New standard menu screen

5.2.2 Create new bar items

The dotted lines, which appear on the blank screen in figure 5.9, represent a bar item. Bar items are groupings of individual menu items. Let's start by defining a bar item and its properties.

By default, a standard menu contains some standard bar items which are components of every application menu. These bar items serve as general purpose items that the end user can use when accessing the application menus. For example, the bar item File is used for saving and canceling panels. The bar item Edit is used for editing functions and other hot key functions used in the application panel. Go serves as a gateway to other application menus in PeopleSoft.

Navigation: Double Click on the dotted lines from a Standard menu screen

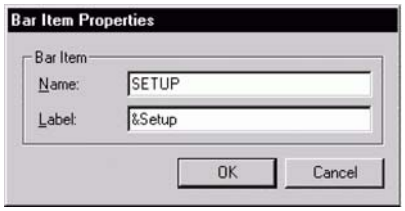


Figure 5.10 Bar Item Properties

Let's consider how we can define a custom bar item attached to an application menu. We define Bar Item properties by double-clicking between the dotted lines.

In figure 5.10 we define a bar item named SETUP with a label of `&Setup`. The character "&" denotes the hot key letter that accesses the menu item using the keyboard without using a mouse. In this example, by pressing ALT-S from the keyboard, the user can access the Setup bar item.

5.2.3 Create new menu items

Once we create a bar item we can add menu items under that bar item by double-clicking on the dotted lines below the Setup bar item. Figure 5.11 illustrates the dotted lines for new Menu Items. The Setup bar item is used to create setup tables for our application. Likewise, we use the Tracking bar item to track all incidents and resolutions in our application. We can create all the menu items that functionally fall under the Setup bar item (figure 5.12).

Navigation: Go →File →Open →Menu →Problem Tracking

Figure 5.11
Menu item properties screen

Navigation: Double-clicking on the dotted lines for Menu Items

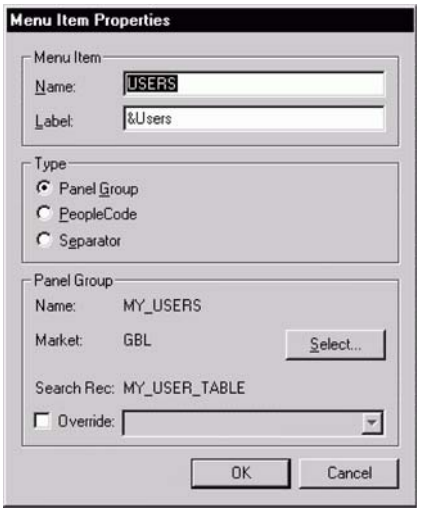


Figure 5.12
Menu item properties screen

5.2.4 Define Menu Item properties

Figure 5.13 illustrates the Menu Item Properties screen. We start by giving the menu item a name and a label. Then we define the menu item as a panel group item.

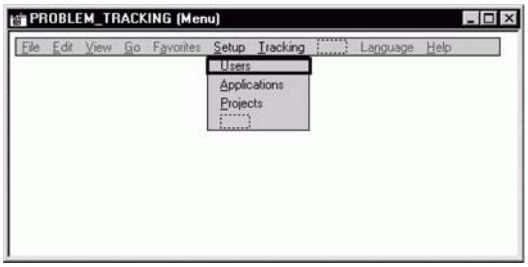


Figure 5.13
Standard menu screen with bar and menu items

Menu items can either be panel groups, Menu PeopleCode, or separator items. When we define the Menu Items as Panel Groups or as PeopleCode items we have to associate the menu items with a panel group. We can choose a panel group associated with the menu item by clicking on the Select button from the Menu Item Properties screen. In our example, we choose MY_USERS as the panel group. We can also override the search record associated with the panel group by clicking the override search record checkbox on and entering an override search record from the Menu Item Properties screen.

TIP We can add more than one menu item under a bar item, and we can add more than one bar item to an application menu.

Before we start adding new menu items to our application menu, we have to add fields, records, panels, and panel groups to build the menu item. Once we design and develop all these objects, we are ready to assemble them into a new menu item for the user.

Menu definitions are stored in tables which are language-related. For each language used in the system, the bar item and menu item descriptions are stored separately. This enables users to view these application menus in different languages.

NOTE Application menus are migrated as a whole from one database to another. The individual bar and menu items cannot be chosen for migration. For this reason, when application menus are migrated across databases, one has to be careful not to overwrite another developer's work in the same menu.

5.2.5 Define menu properties

Figure 5.14 Menu Properties—General tab

Now that we have attached menu items to the application menu, it's time to define properties for the application menu as a whole. Figures 5.14 and 5.15 illustrate the tabs under the Menu Item Properties screen.

The Menu Properties window can be brought up by choosing File/Object Properties from the Application Designer menu. The application menu should be open to perform this operation.

General tab

The General tab contains a brief description and Comments for the

application menu. The Comments section can be used to maintain a modification log for the application menu. The General tab displays the last Date/Time the menu was updated and the ID of the operator who updated it.

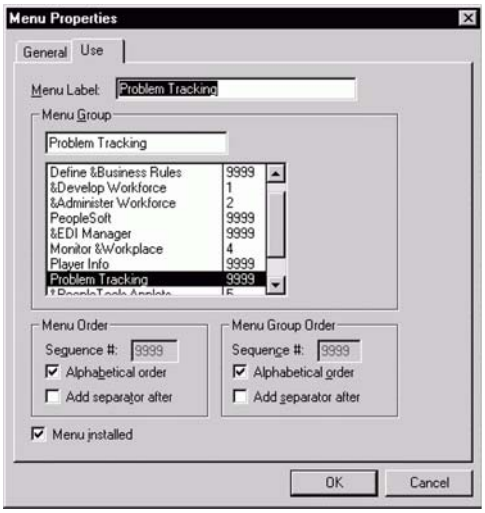


Figure 5.15 Menu Properties—Use tab

Use tab

Under the Use tab, we can provide a label for the menu. This label will be seen under the Go menu online. We can also group one or more application menus together in a menu group. The menu groups appear as the first list when we choose the Go menu. When the names for the menu and the menu group are the same, the menu appears as part of the list under the Go menu. We can also provide a sequence for how the menu groups appear under the Go menu. In addition, the sort order for both the menu and the menu group can be specified under the Use tab.

5.2.6 Save the menu definition

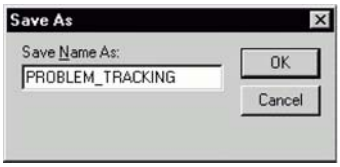


Figure 5.16 Menu definition save

We are now ready to save our menu definition. Figure 5.16 illustrates the Save Panel window for a menu definition.

We can name our application menu PROBLEM_TRACKING and click the OK button to save the menu definition.

5.2.7 Pop-up menus

Pop-up menus are used to access context sensitive information. For example, in fields defined as dates, we can provide a calendar that will pop-up when the user right-clicks on the Date field. Pop-up menus can either be attached to panels or panel fields. The attached pop-up menu is activated when the user right-clicks on the panel field.

Pop-up menus can either be used for panel transfers or for executing PeopleCode. Panel transfers require parameters such as menu name, panel group, panel, and action which are required when transferring to a panel. PeopleCode is attached to the panel field and the PrePopup PeopleCode event is executed before the pop-up menu is shown.

To bring up another panel using PeopleCode, we define the pop-up menu as PeopleCode and attach a PrePopup PeopleCode event to the record field. When you define the pop-up menu as a transfer, only one transfer panel can be defined. This is

a Non-Modal transfer definition. When the pop-up menu is defined as PeopleCode, the DOMODAL PeopleCode function can be used to transfer focus to different panels for different panel field values. This is a Modal transfer definition. (To learn more about PrePopup PeopleCode events, refer to part 3 herein.)

5.3 AUTHORIZING USERS

After we build menu items, we have to authorize user access to these menu items. We can do so by using the Security Administrator tool in PeopleSoft (found in the PeopleTools menu group under the Go menu).

Operator security in PeopleSoft is driven using two fields: the OPRID and the OPRCLASS fields. OPRID is a unique identification given to a PeopleSoft user. Every operator must have a password to log on to the application. An operator may belong to one group or many groupings of operators, otherwise known as operator classes.

Prior to PeopleSoft release 7, an operator could be assigned only to one particular operator class. This made it difficult for system administrators to define unique groupings of operators. System administrators either had to change the business needs or create more classes. Starting with PeopleSoft release 7, an operator can be associated with more than one operator class. All the security attributes of the operator classes translate down to the operator. In other words, if the operator belongs to two operator classes, attributes of both these classes are attached to the operator profile.

Attributes control the creation of operator classes. Every user needs an operator ID to use the PeopleSoft application. It's often a difficult task to find an operator class that fits the user's security profile. Let us look at the attributes attached to an operator class to better understand the previous statement.

The following criteria are used to determine the creation of operator classes in the system:

- *Menu items* Menu items that the operator can access determine the operator class for the user. If a group of operators has the same set of menu items they can access, then they can potentially be under the same operator class, provided all the other criteria are similar.
- *Sign-on times* If the operators can have access to the system at similar time durations, then they can belong to the same operator class. Sign-on times control the time when a user can log on to the PeopleSoft application.
- *Process groups* Process groups are identifiers by which batch processes are differentiated. These identifiers can either be functional identifiers or any other identifier by which the processes are separated. For example, identifiers in a PeopleSoft Human Resources system can be PAYALL, HRALL, BENALL, and so forth, differentiating the processes into functional areas. Hence, operators who run similar processes can belong to the same operator class. When an operator has security to a particular process group, that does not necessarily mean that the operator can run all processes under that process group. The operator has to have access to the menu items that run these processes as well.

- *Functional security* Functional security is row-level security which secures application data in the PeopleSoft system. In a PeopleSoft HRMS application certain fields such as Department, Business Unit and Pay Groups can be used to provide row level security. Likewise in a PeopleSoft General Ledger application, fields like Business Unit, Product, and Location may be used for functional security. Each operator is able to access data based on these functional attributes. For example, a group of operators in the Michigan plant can only access employees who work in the paygroup which has all employees from the Michigan plant. So all operators who have similar access based on functional criteria can belong to the same operator class.

Now that we know the criteria by which operator classes are determined, let's build these classes using the Security Administrator tool.

5.3.1 General attributes

Let's create an operator class called MYADMIN, which will be used to create operators who can access our Problem Tracking application menu. We have defined the Security definition type as `Class of Operators`. We can provide a Business process map for the operator class under the General attributes. Business process maps are graphical representations of application menus which the users can view to access panels. We can control how the user views a Navigator display of menu items using the Configuration Manager.

We can define the Background Disconnect Interval and Online Time-out minutes in the General Attributes screen. Background Disconnect Interval controls the disconnection of icons which stay in the background and use system resources by not getting disconnected. Online Time-out Minutes control the time out of the PeopleSoft session as a whole after a certain amount of idle time. We now save this operator class by choosing File/Save from the menu in the Security Administrator screen. When we are prompted for a name, we name our operator class MYADMIN.

5.3.2 Menu items

By clicking on the Menu Items icon on the left side (figure 5.17), we start authorizing user access to menu items. Select the menu which the user can access by choosing Insert/Menu Name from the Security Administrator screen. All menu items which belong to the user application menu appear on a list box. In our example, we choose PROBLEM_TRACKING as the menu item. Then we can highlight the menu items which the user can access or choose Select All to select all menu items in the application menu. We can also provide Display Only access to a particular menu item by clicking on the Change Display-Only button. Once we have chosen all menu items which the user can access, we click OK to close the Select Menu Items window (figure 5.18).

Navigation: Go →PeopleTools →Security Administrator →File →New

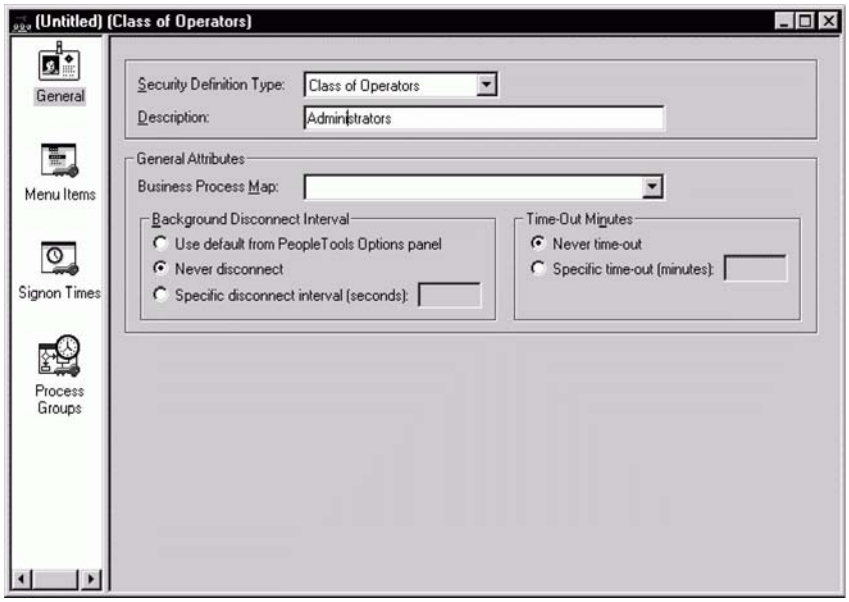


Figure 5.17 Operator Security window—General view

Navigation: Insert/Menu Name under the Menu Items tab

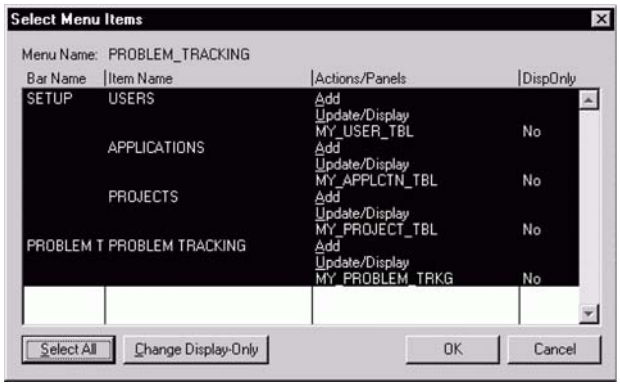


Figure 5.18
Menu items selection screen

Notice all the menu items (figure 5.18) have been highlighted for user access, and the DispOnly column reads "No" on all the selected items. We can add more application menus and menu items for user access by repeating the same steps using other application menus from the Security Administrator screen.

5.3.3 Sign-On Times

By choosing the Signon Times icon from the left side (figure 5.19), we can enter logon times for users. Basically, we enter a sign-on time for each day of the week by using Insert/Signon Times from the Security Administrator screen. Let us take a look at the sign-on times for MYADMIN operator class. We can add more than one interval of time when the user can access the system. For example, we can add 00:00 hours as the starting time and 10:00 as the ending time on Sunday, and also add 13:00 and 15:00 as the starting and ending time on Sunday. This allows the user to access the system only between those times on Sundays.

Navigation: Click on the Signon Times tab

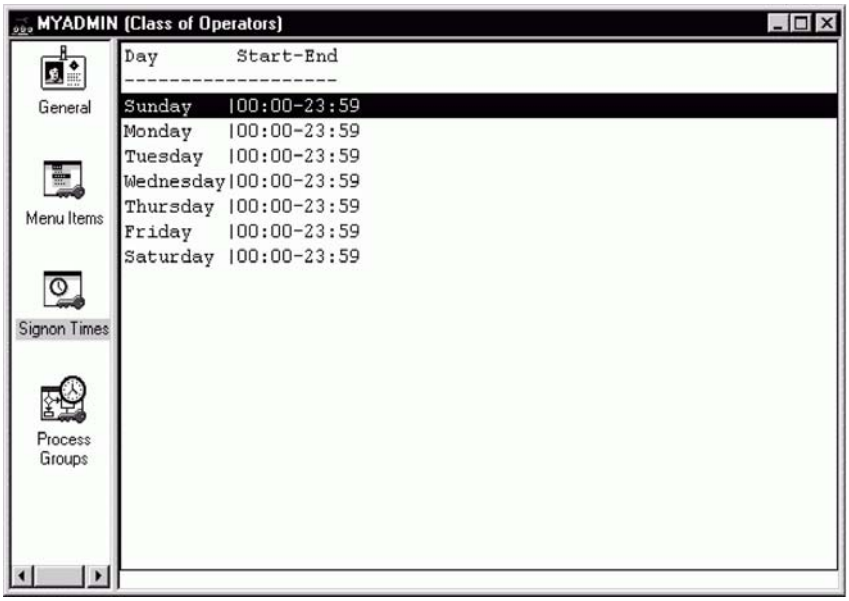


Figure 5.19 Operator Security window—Signon Times view

5.3.4 Process groups

Process groups are groups of processes that identify process definitions in PeopleSoft. Process groups control process security in PeopleSoft. For example, let's say a process named PER005 belongs to the HRALL Process Group. All operator classes, which have access to the HRALL Process Group, will be able to run that process provided they have access to the menu item that runs the process. So we include all the process groups that the operator class can access by using Insert/Process Groups from the Security Administrator screen.

In figure 5.20 we can see the process groups for the MYADMIN operator class.

Navigation: Click on the Process Groups tab

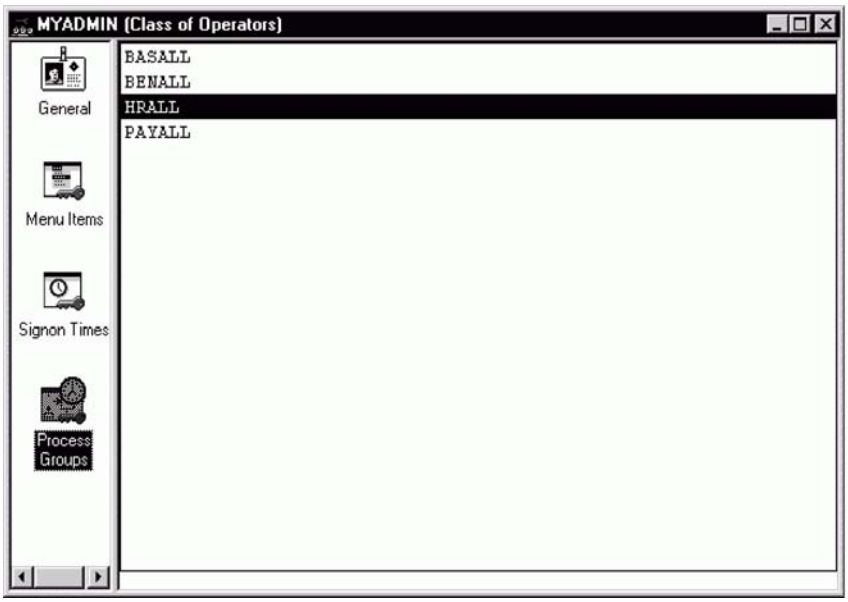


Figure 5.20 Operator Security window—Process Groups view

Figure 5.20 indicates that the MYADMIN operator class can access all processes defined under BASALL, BENALL, HRALL, and PAYALL process groups. The operator class also needs access to the menu items that run these processes. If a hundred processes are defined under the HRALL process group, the user does not always have access to all the menu items that initiate the hundred processes, so menu item access and process group access work hand-in-hand to determine what processes a user can run.

Figure 5.21 illustrates how a process definition is defined and how the process definition attributes are linked to operator security.

The process definition in figure 5.21 belongs to the HRALL and HRCAN process groups. Also the panel groups which are attached to menu items are defined. The combination of both these attributes gives the user access to this process.

Navigation: Go →PeopleTools →Process Scheduler →Use-Process Definitions

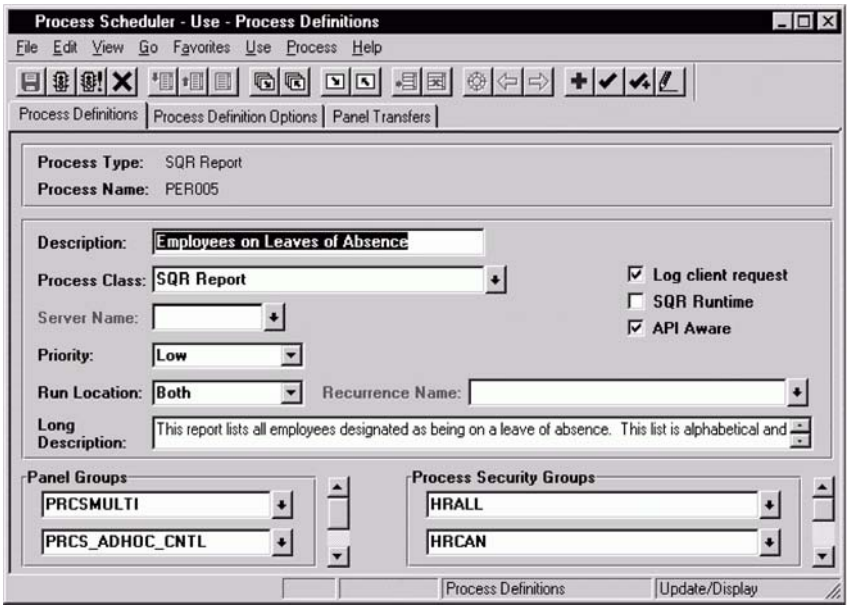


Figure 5.21 Process definition

5.3.5 Process profiles

Every operator class has a process profile which controls certain processing parameters. We can change the process profile for an operator class by choosing Edit/Process Profile from the Security Administrator screen.

Process profiles control printer, output destinations, and process view/update parameters for an operator class. These are default parameters for an operator class, and the operator can override these parameters at the time of running a process. Let us take a look at the process profile screen for MYADMIN operator class (figure 5.22).

In figure 5.22, we can notice that the file and printer destinations are separated by Client and Server destinations. The Server destination is the server where the PeopleSoft Process Scheduler is currently running. Let's review the parameters which control how the operators who belong to this operator class view and update processes and run controls:

- *Allow Process Request—View By*—controls the processes that the operator class can view on the Process Monitor.
- *Allow Process Request—Update By*—controls the processes that the operator class can update on the Process Monitor.
- *Allow Requestor to Override Output Destination*—allows the operator class to override the default output destination in a process request.

- *Allow Requestor to Override Server Parameters*—allows the operator to override the Server parameters in which the process runs.
- *Allow Requestor to View Server Status*—allows the operator class to view the status of the Process Scheduler.
- *Allow Requestor to Update Server Status*—allows the operator class to stop, suspend, or restart the Process Scheduler.
- *Allow Requestor to Update Recurrence Definition*—allows the operator class to define and update Recurrence in a process request.

Navigation: Edit →Process Profile (MYADMIN operator class is open)

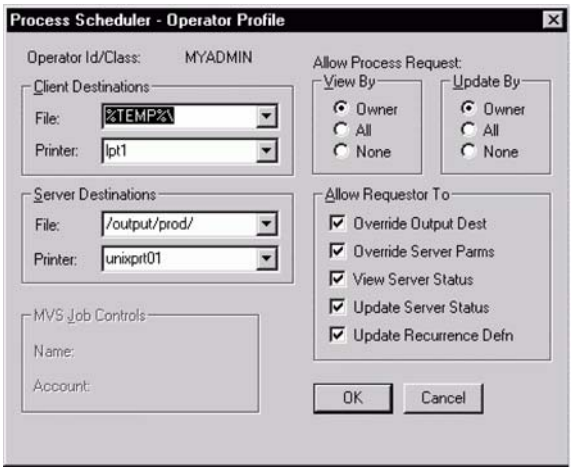


Figure 5.22
Operator Process Scheduler profile screen

Now we need to save the operator class once again. We do so by choosing File/Save from the Security Administrator screen. All attributes defined are now attached to MY_ADMIN operator class.

5.3.6 Creating operators using operator class definitions

From now on, MYADMIN operator class can be used as a template to create actual operators in the system. We can bring up a new screen by choosing File/New from the Security Administrator screen, but first we have to make sure the security definition type is set to `Operator` in order to save an operator definition (figure 5.23).

Only the General tab and the Classes tab are necessary to create an operator definition. The other tabs translate from the operator class attributes for the operator. We now enter the operator attributes to complete the operator definition. Some attributes are required in order to save the operator definition.

Navigation: File →Open →MYOPER

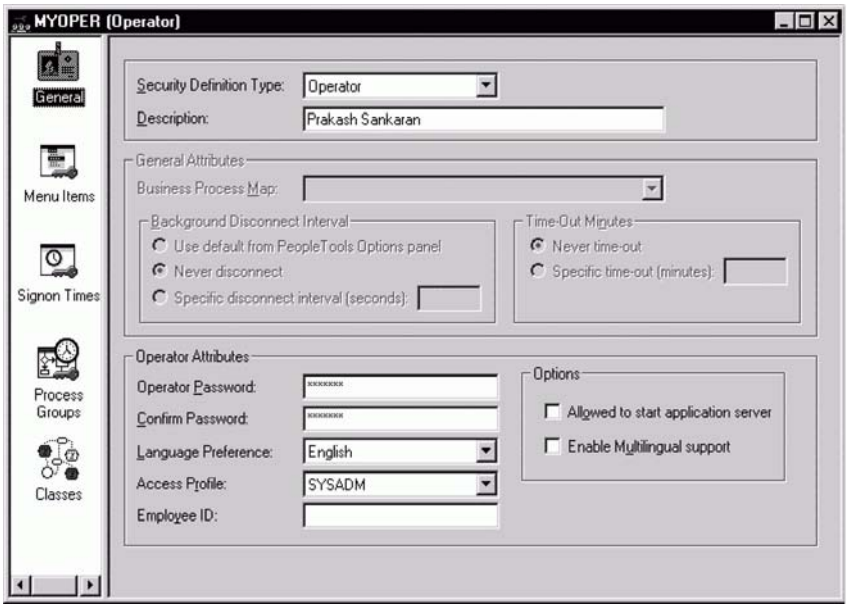


Figure 5.23 Operator definition for MYOPER

General tab

Operator Password The password that the operator uses to sign on to PeopleSoft.

Confirm Password A confirmation to save the password for the operator.

Language Preference The base language with which the operator signs on to PeopleSoft. This plays a significant role, controlling the language used to display descriptions when the operator signs on to PeopleSoft.

Access Profile The access ID the operator needs to login to the database. PeopleSoft uses the access ID to create a session in the database. Access IDs are the only IDs which have access to database tables and views. Using the access ID, operators in the system can access database tables and views. One access ID can be used to provide entry by all users in the system. Therefore, the Database Administrator needs to maintain grants and permissions only for that one access ID in the system. When the operator logs into the system, the operator's access to the system is verified, after which the access ID is used to retrieve data from tables and views.

Employee ID If the operator is also an employee and the Human Resources system is maintained using PeopleSoft, then we can enter the Employee ID for the operator here. In a PeopleSoft Human Resources system, entering the Employee ID prevents operators from changing their own data.

Allowed to Start application server This option allows the operator to start a PeopleSoft application server. For example, a Tuxedo server is an application server running the PeopleSoft application.

Enable Multilingual Support This option lets the operator edit data in multiple languages. By simply accessing the Language menu item from any application menu, the user can switch the language to edit fields in panels that are stored in multiple language or related language tables.

Classes tab

This attribute is used only for operator definitions. As mentioned before, starting from PeopleSoft release 7, an operator can be included in more than one operator class, making it easy for system administrators to create operator classes.

System administrators can create operator classes which define panel access separately. They can also create operator classes which define function security. Then, they can attach these operator classes to the operator. The operator class that defines the panel access provides the operators with the appropriate menu items. The class that defines the functional security secures the data that they will access. For example, in a PeopleSoft HRMS implementation, we can create an operator class which has menu items/panels which a typical HR user can access. This one class provides panel access to all HR users throughout the system. At the same time, we also need to create individual classes that contain the appropriate application security for these users. Let's take a look at the following matrix to better understand this process. We assume that all these users are HR users, and that they belong to different locations processing various paygroups and departments.

In table 5.1, the column on the left contains the actual operator IDs in the system. All three users access the same set of panels and menu items. Hence, they are attached to the HRADMIN class. They will, however, be able to access only the departments and paygroups in their respective locations. So they are also each attached to individual classes (NYCHR, SFOHR, and WDCHR) which contain the respective departments that they are able to access. The same theory can be used across all PeopleSoft applications to arrive at the number of operator classes needed in the system.

Table 5.1 Operator security

Operators	Location	Panel Class	Security Class
SOSGOOD	New York	HRADMIN	NYCHR
CFINNIGA	San Francisco	HRADMIN	SFOHR
GMORDIN	Washington, D.C	HRADMIN	WDCHR

To attach the operator to an operator class we choose Insert/Classes from the Security Administrator screen. We also choose the primary operator class for the operator by clicking on the Primary checkbox (figure 5.24).

Navigation: Click on the Classes tab

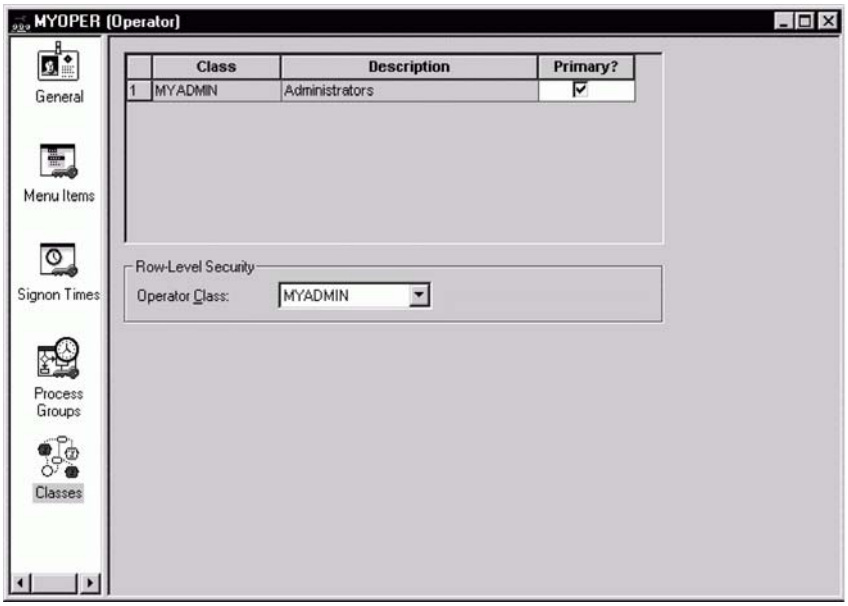


Figure 5.24 Operator Security window—Classes view

The primary operator class controls the application data security for the operator. The primary operator class can also provide panel and menu access for the operator. If we follow this path, we will end up creating more operator classes than necessary, but if we can separate the application data security and the panel security attributes, we can reduce the number of operator classes created in the system.

Row-level security

Navigation: File →Save

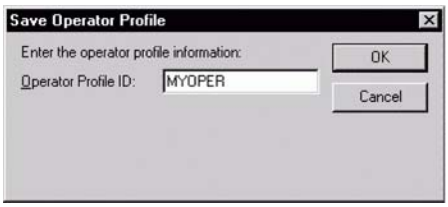


Figure 5.25 Save an operator definition

To secure data, PeopleSoft uses the primary operator class for the user. PeopleSoft release 7 offers a new feature, ROWSECCLASS. This is the row-level security class for the operator. ROWSECCLASS requires a change in the search view definitions that PeopleSoft delivers in its application. Currently, all search views contain either OPRID or OPRCLASS fields. PeopleTools automatically uses the primary operator class at search time. It attaches an extra condition to the `WHERE` clause of the search view to control the selection of data from

the system. If we replace the OPRCLASS or OPRID fields in security views with the field ROWSECCLASS, PeopleTools now uses that class for the operator to control data access.

We now save the operator definition by providing a name. In our example, we provide MYOPER as the name for the operator (figure 5.25).

5.3.7 Understanding functional security (Trees)

We have discussed creating functional security and controlling the data the user can see in the system. Now let's see how we can define the data which the operator class can access. We do this by using the ADMINISTER HR SYSTEM menu in a PeopleSoft HRMS application. In PeopleSoft HRMS, the DEPARTMENT field is used as a key field to control data access.

The system contains a department security tree which contains the organization structure. This security tree contains organization groupings of departments.

A group of departments may report to a location, and a group of locations may report to a divisional office. The divisional office ultimately reports to a corporate office. This organizational hierarchy is built using the tree manager, which, in turn, is used to build the department table in PeopleSoft HRMS. Each node in the tree is an entry in the PS_DEPT_TBL record. All employees in the system are assigned to a department.

Before we see how we can define the departments which an operator class can access, let's look at the department security tree in a PeopleSoft HRMS system (figure 5.26).

Navigation: Go →PeopleTools →Tree Manager

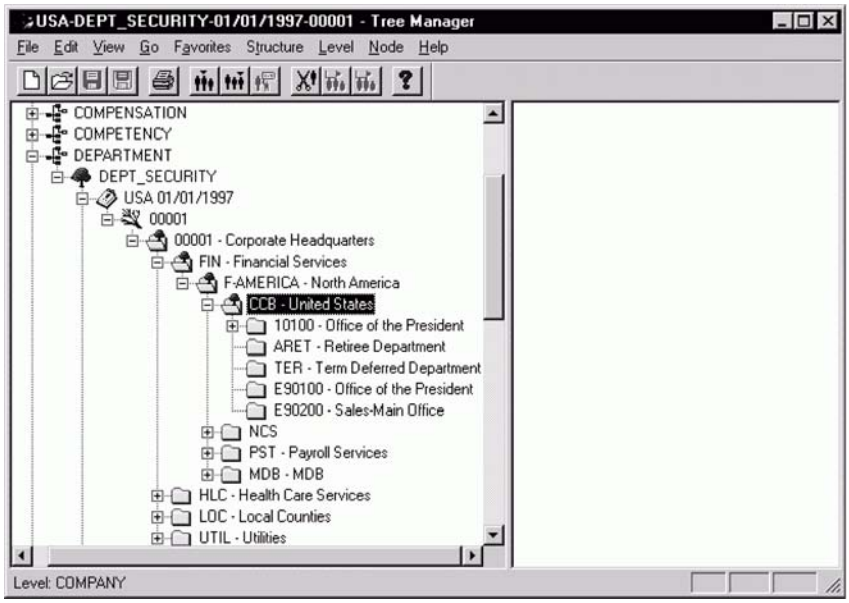


Figure 5.26 Department security tree in PeopleSoft HRMS

If we look at the tree in figure 5.26, we see that it has a structure called DEPARTMENT, which contains the DEPT_SECURITY trees. `00001` denotes corporate headquarters, and FIN, HLC, LOC, and UTIL represent different divisions providing different services within the organization. USA is the set ID for corporate headquarters and all the divisions (services) and departments underneath them.

Now, let's look at the data security screen where we define these values to control data access for an operator class. The Maintain Data Security screen in figure 5.27 has two rows for operator class MYADMIN. The first row provides MYADMIN access to all services, divisions, and departments under `00001` (Corporate). The second row excludes access only to department `10100` (Office of the President). This means MYADMIN can access all employee records for employees who report to `00001` (Corporate), except for those employees who report to `10100` (Office of the President).

These values are saved in PS_SCRTY_TBL_DEPT. This table is used in all the search views in PeopleSoft HRMS which control department security.

Navigation: Go →Administer HR System →Use →Maintain Data Security

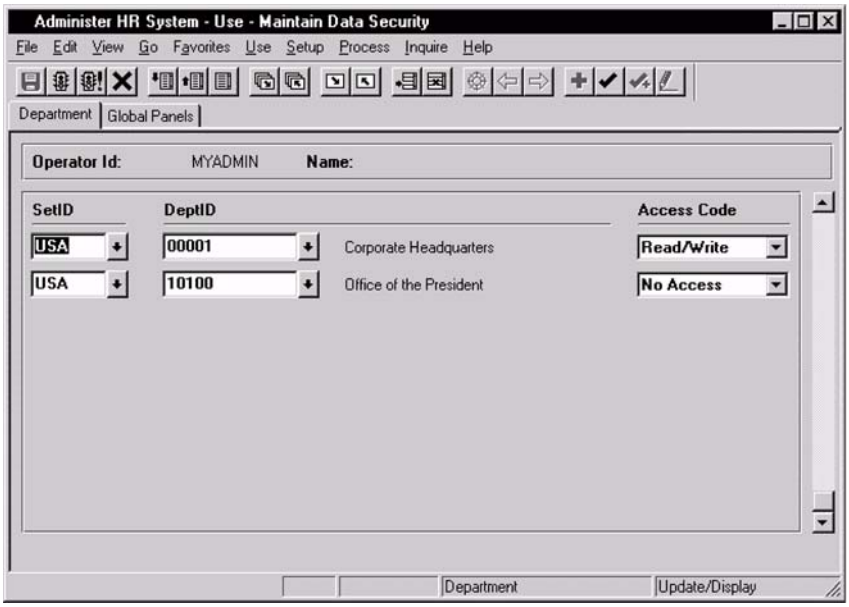


Figure 5.27 Maintain data security in PeopleSoft HRMS

KEY POINTS

1. Panel Groups can contain one or more panels in them. Panel Groups are used to separate panel fields by function. Panel Groups are also used to update multiple record definitions at the same time.
2. Panel Groups attach an application panel to a menu item.
3. Menus can be standard or pop-up menus. Standard menus come delivered with standard bar items. More bar items and menu items can be added to the standard application menu.
4. A menu item is a single unit used to provide access to applications.
5. A panel group containing one or more panels is attached to a menu item. Access can be provided to any one or all of the panels in the panel group.
6. The Security Administrator tool is used to authorize users for access to menu items. OPRID and OPRCLASS fields are the two primary fields used to define Operator Security.

CHAPTER 6

Enhancing your application

6.1 CREATING AND USING PROMPT RECORDS

Prompt records can be either database tables or database views. We utilize a prompt record to create a drop-down list that contains the possible list of values for a field. A user entering values into such a field will be restricted to the values produced by the prompt record. Drop-down lists work only on character fields. (Usually fields that are used as codes have drop down lists behind them.)

Prompt records are attached to record fields, which means only one prompt record can be used for a record field at a given time. This is important because prompt records for a record field can be changed dynamically during panel processing. Before we learn more about dynamic prompt records, let's cover the basics.

Prompt records, which are similar to search records, are primarily put to work using search and database keys. A key difference between prompt records and search records

is the input mechanism. While prompt records are supplied with inputs from fields in the panel, search records are supplied with inputs from input dialog boxes. Nevertheless, the principle behind the workings of prompt and search records is the same.

6.1.1 Principles of prompt records

Prompt records contain fields defined as key fields. The fields prompted are defined as key fields in the prompt records. The prompted field can either be the first key field in the prompt record or any other field in the key field list. When the prompted field is not the first key field on the prompt record, the fields that precede the prompted field must be populated for the prompt to work. Since prompt fields are database keys, they facilitate faster searching during prompt processing. Let's consider a few examples:

6.1.2 Prompt records with a single search key

Prompt records with a single search key are simple enough to understand. The prompted field is the only search key on the prompt record. In this case, no reason exists to supply an input to the prompt record. The prompt list is brought up by clicking on the drop-down arrow or by pressing F4.

In our application, we can create a single key prompt field. In MY_PROJECT_TBL, we can define a prompt record behind MY_APPLICATION_ID field.

Let's walk through this process of defining the prompt record in MY_PROJECT_TBL. First, we have to open the record definition for MY_PROJECT_TBL using the Application Designer (figure 6.1).

Navigation: Edit →Record Field Properties

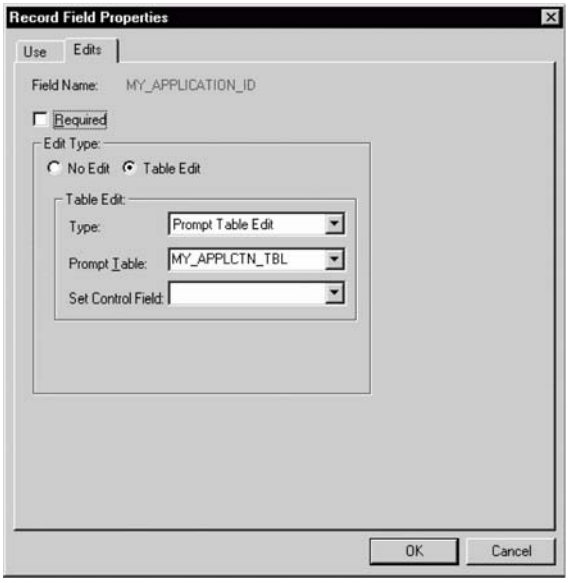


Figure 6.1
Defining prompt records

We used MY_APPLCTN_TBL as the prompt record, so let's take a quick look at the search key definition for this prompt record. In figure 6.2, we can see that the only search key in the prompt record is the MY_APPLICATION_ID field. This is an example of a simple prompt record: the prompt list appears without any input from the panel, and once the prompt record is defined for a record field, the prompt record can be used on any panel that contains that record field.

In figure 6.2, we can see that MY_APPLICATION_ID is marked as a search field on MY_APPLCTN_TBL. All fields that are marked as List Items also appear on the prompt list. Now, let's look at the prompt field and the prompt list as they appear on an online application panel.

Figure 6.3 illustrates the drop-down arrow on MY_APPLICATION_ID field.

Navigation: File →Open →Record →MY_APPLCTN_TBL

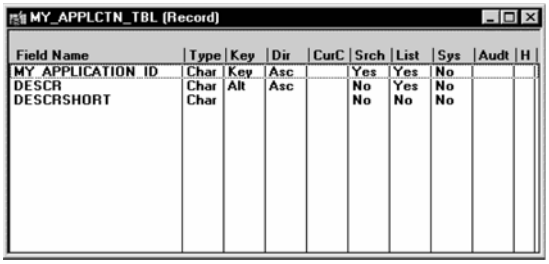


Figure 6.2
Prompt record—Key display

Navigation: Go →Problem Tracking →Setup →Projects (User Application)

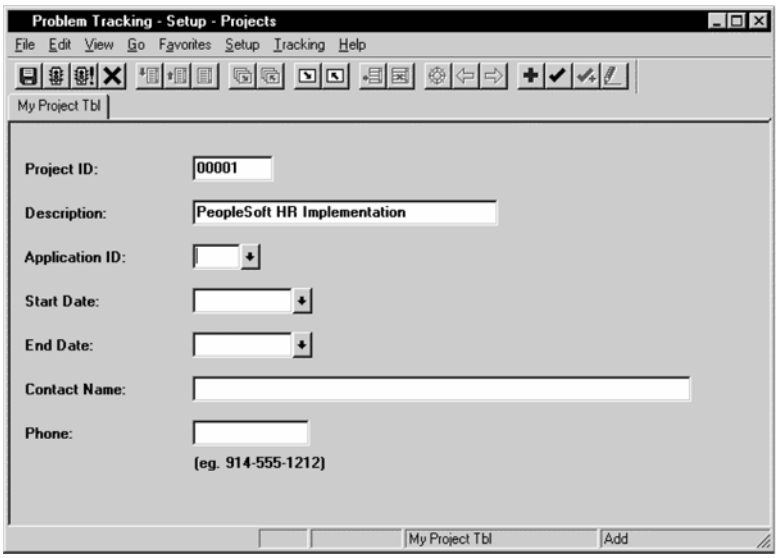


Figure 6.3 Prompt records on an application panel

We press F4 from the field to bring up the prompt list. Now, consider the actual prompt list generated from this field (figure 6.4).

Navigation: F4 from the field (or) Clicking on the drop down arrow

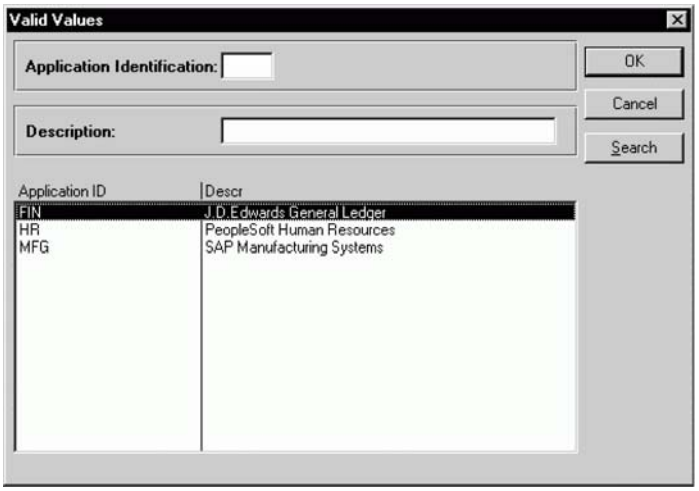


Figure 6.4
Prompt list from the Projects panel

In the prompt list of valid values, both the search key and all the list box items appear. In figure 6.2 we saw the key definition for MY_APPLCTN_TBL. In addition to the MY_APPLICATION_ID field defined as the search key, the DESCR field is defined as a list box item. Since MY_APPLICATION_ID is a code, the DESCR field provides a description to help the user choose the correct application ID from the prompt list. By highlighting any one of the valid values from the prompt list and choosing OK, we are able to populate the panel field with that value.

6.1.3 Prompt records with effective dates

Some prompt records are effective-dated. Every time the characteristics of the stored information changes, a new row is inserted in the prompt table with a new effective date. In PeopleSoft, the Application Processor automatically returns the list of valid values as of the effective date on the panel.

Since we do not have such an example in our application, let's look at an example from the PeopleSoft HRMS application. Let's say that employee job-related information is stored in a record named JOB. This record has two fields in particular that store the COMPANY and the PAYGROUP for the employee. The prompt record for COMPANY is COMPANY_TBL, and this record has an effective date and an effective status. These fields—special fields that the Application Processor treats differently—are named EFFDT and EFF_STATUS in PeopleSoft. The COMPANY field is prompted

from the JOB DATA panel group. Let's take a look at the panel for further explanation on how the EFFDT field affects the prompt.

The panel in figure 6.5 has an effective date which controls the prompt list on the COMPANY field. The effective date on this panel is compared with the effective date on COMPANY_TBL to provide the prompt list. All companies active and effective as of 9/1/1996 appear in the prompt list. EFF_STATUS field on the COMPANY_TBL controls the active and inactive status of the company, and EFFDT field controls the effective date. Now let's look at the key definition for COMPANY_TBL (figure 6.6).

Navigation: Go →Administer Workforce (U.S) →Use →Job Data (PeopleSoft HRMS)

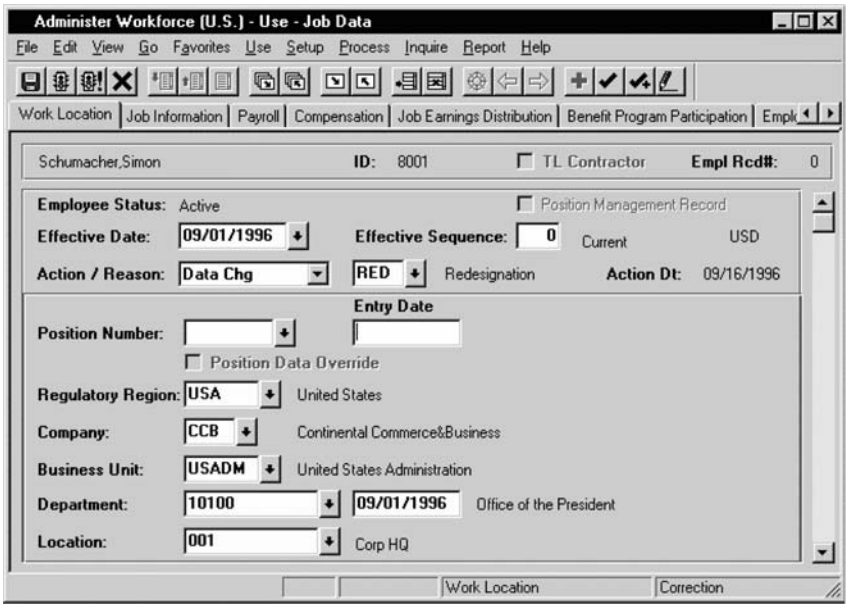


Figure 6.5 PeopleSoft HR panel—Prompt record with EFFDT

Navigation: File →Open →Record →COMPANY_TBL (Application Designer)

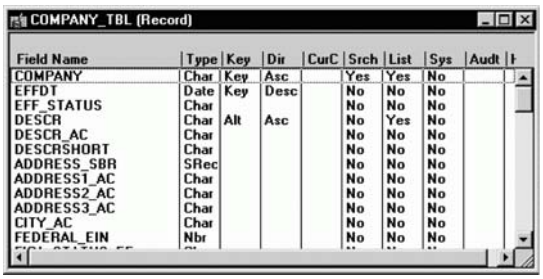


Figure 6.6
Search keys of a prompt record with EFFDT and EFF_STATUS

Notice that COMPANY is the only search key in this prompt record. The effective date on the JOB DATA panel is compared with the effective date on the COMPANY_TBL. Also, the effective status on COMPANY_TBL should be active as of the effective date on JOB DATA. In table 6.1, we can see combinations of EFFDT from JOB and EFFDT, EFF_STATUS from the COMPANY_TBL. The last column on the matrix here denotes whether the company appears on the prompt list.

Table 6.1 Effective date comparison

COMPANY	COMPANY EFFDT	COMPANY EFF. STATUS	JOB EFFDT	Prompt List
CCB	1/1/1996	Active	9/1/1996	Yes
CCB	5/1/1996	Active	9/1/1996	Yes
CCB	11/1/1996	Inactive	9/1/1996	No
CCB	1/1/1997	Active	9/1/1996	No

The first two combinations satisfy the prompt list; the last two entries do not. The COMPANY_TBL entries for the last two entries are either inactive, or, the effective date is in the future. We can deduce that the row in the prompt record should be active on or before the effective date on the panel that contains the prompt field. *If there is no effective date on the panel that contains the prompt field, then the system date is used for comparison.*

6.1.4 Prompt records with multiple search keys

Now, let's consider a situation where a prompt record has more than one search key; using the PAYGROUP field from the JOB DATA panel as an example. This field has the PAYGROUP_TBL as the prompt record. Let us take a look at the search keys on the PAYGROUP_TBL (figure 6.7).

The rules for effective date and effective status still apply to this prompt record. The only difference is the additional search key. PAYGROUP is the field being

Navigation: File →Open →Record →PAYGROUP_TBL (Application Designer)

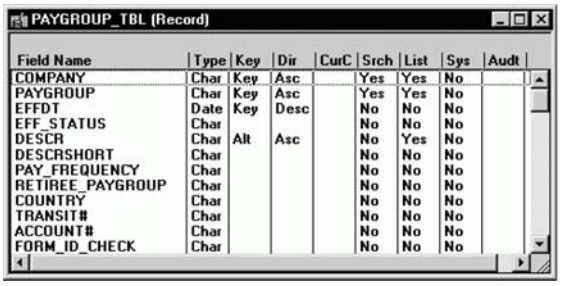


Figure 6.7 Prompt records with multiple search keys

prompted, and it is not the first key on this prompt record. COMPANY is the first search key here.

It is a simple task to make this prompt record work efficiently. The COMPANY field has to be populated with a valid value for the prompt on the PAYGROUP field to work correctly. For this reason, the panel fields have to be laid out in such a way that the search keys appear in the correct order. The COMPANY field can be an input field, a display-only field, or even a hidden field in the panel. Simply by populating the COMPANY field, we are supplying an input to the prompt. The prompt search uses the value entered in COMPANY and produces a list of PAYGROUPS that belong to the COMPANY.

The same rules apply for any number of search keys. All high level keys have to be populated in order for a prompt field to work correctly. In PeopleSoft HRMS, another record, the PAY_CALENDAR, is used as a prompt record to produce a list of payroll end dates. Let's look at the keys on this record to understand how a prompt list is provided for the PAY_END_DT field.

In figure 6.8, we see three search keys: COMPANY, PAYGROUP, and PAY_END_DT fields. In order for the prompt to work correctly on the PAY_END_DT field, the COMPANY and PAYGROUP fields have to be populated with valid values.

Navigation: File →Open →Record →PAY_CALENDAR (Application Designer)

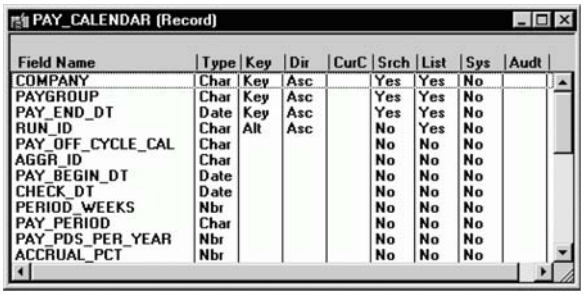


Figure 6.8
Prompt records with multiple search keys (more than two)

TIP Values for all higher level key fields must be supplied for a prompt list to work. If the prompted field is the third key on the search record, the first two key fields must have values for prompt processing. The Application Processor verifies whether any rows in the prompt record satisfy the values in the key fields. If rows are found, a prompt list is provided to the user.

6.1.5 Dynamic prompt records

Sometimes, the prompt record behind a field cannot be determined until runtime. The data contained in the panel dictates the prompt record to be used. The prompt

record has to then be chosen dynamically through a PeopleCode event. Let's walk through an example from the PeopleSoft HRMS application, a variable prompt record defined on the HEALTH_BENEFIT record. This record stores health benefits enrollments. The field that uses the variable prompt is BENEFIT_PLAN.

The first prompt record is an SQL view that contains all benefit plans not defined for COBRA. The second prompt record is also an SQL view that has all benefit plans defined for COBRA. A flag called COBRA_PLAN (in the BEN_DEFN_PLAN record) is set to a value of `Y` for COBRA. Both the SQL views are built using the BEN_DEFN_PLAN record.

We set the variable prompt records using a `RowInit` PeopleCode event in HEALTH_BENEFIT. Any field from the DERIVED record can be placed on the panel, and this field will be populated with the actual prompt record name through a PeopleCode event. The prompt record name on the record field definition is a % sign and the actual field name from the DERIVED record.

For example, if the field name on the DERIVED record is EDITTABLE, then the prompt record name is defined as `%EDITTABLE`. The % sign is recognized as a field from the DERIVED record. As long as this field is populated with the correct record name, the prompt list works correctly. Let's look at how the prompt record is defined on the HEALTH_BENEFIT record (figure 6.9).

Navigation: File →Open →Record →HEALTH_BENEFIT (Application Designer)

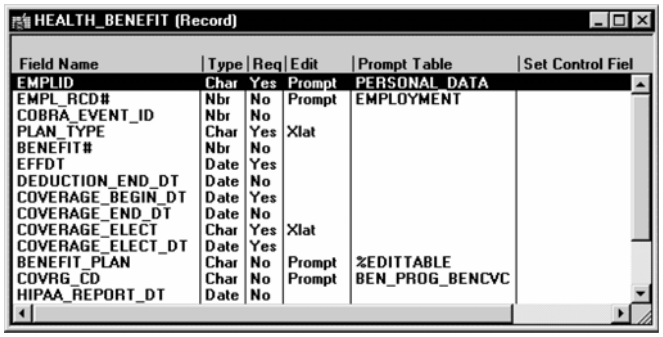


Figure 6.9
Variable prompt record definition

The prompt record is defined as `%EDITTABLE`, which means that the field EDITTABLE from the DERIVED record has to be placed on the panel for the prompt to work correctly. The PeopleCode event that populates this work field appears in figure 6.10.

Figure 6.10 illustrates how the PeopleCode event populates the EDITTABLE field with two different values based on the panel name being used. Also, notice that the `RowInit` event populates this field before the user gets a chance to access the prompt list.

Navigation: File →Open →Record →HEALTH_BENEFIT →View →PeopleCode Display

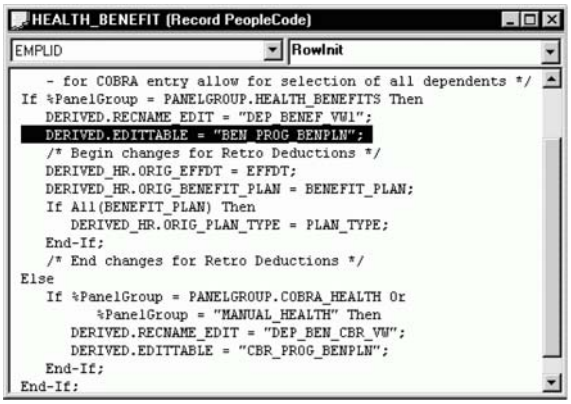
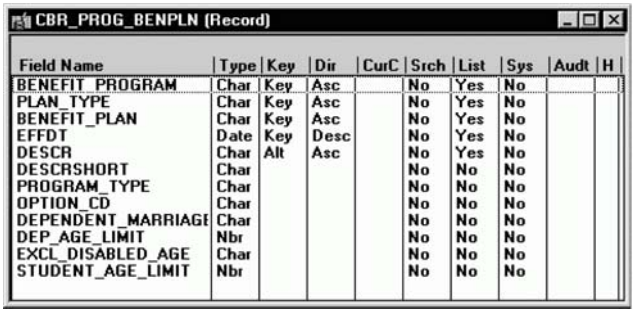


Figure 6.10
Rowlnit PeopleCode event to set variable prompt record

The search keys on the prompt record still work the same way as we have seen before. The search keys have to be the same on all prompt records. The rows returned in the prompt list are different. Let's finish this section by looking at the search key definition for the two prompt records used in the example (figures 6.11 and 6.12).

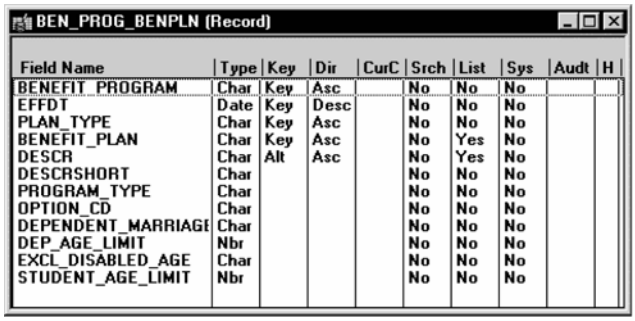
Navigation: File →Open →Record →CBR_PROG_BEN_PLN



Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H
BENEFIT_PROGRAM	Char	Key	Asc		No	Yes	No		
PLAN_TYPE	Char	Key	Asc		No	Yes	No		
BENEFIT_PLAN	Char	Key	Asc		No	Yes	No		
EFFDT	Date	Key	Desc		No	Yes	No		
DESCR	Char	Alt	Asc		No	Yes	No		
DESCRSHORT	Char				No	No	No		
PROGRAM_TYPE	Char				No	No	No		
OPTION_CD	Char				No	No	No		
DEPENDENT_MARRIAGE	Char				No	No	No		
DEP_AGE_LIMIT	Nbr				No	No	No		
EXCL_DISABLED_AGE	Char				No	No	No		
STUDENT_AGE_LIMIT	Nbr				No	No	No		

Figure 6.11
Variable prompt record—Key display

Navigation: File →Open →Record →BEN_PROG_BENPLN



Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H
BENEFIT_PROGRAM	Char	Key	Asc		No	No	No		
EFFDT	Date	Key	Desc		No	No	No		
PLAN_TYPE	Char	Key	Asc		No	No	No		
BENEFIT_PLAN	Char	Key	Asc		No	Yes	No		
DESCR	Char	Alt	Asc		No	Yes	No		
DESCRSHORT	Char				No	No	No		
PROGRAM_TYPE	Char				No	No	No		
OPTION_CD	Char				No	No	No		
DEPENDENT_MARRIAGE	Char				No	No	No		
DEP_AGE_LIMIT	Nbr				No	No	No		
EXCL_DISABLED_AGE	Char				No	No	No		
STUDENT_AGE_LIMIT	Nbr				No	No	No		

Figure 6.12 Variable prompt record—Key display

The key structures are exactly the same on both prompt records. Because the SQL views return different rows based on application context, a variable prompt record is necessary.

TIP Variable prompt records are used with the help of a field from the DERIVED record. For example, if the field from DERIVED record is called RECNAME_EDIT, the prompt record is defined as `%RECNAME_EDIT` in the record that uses the prompt record. A PeopleCode event will populate the variable prompt record name at run time.

6.2 CREATING AND MAINTAINING TRANSLATE VALUES

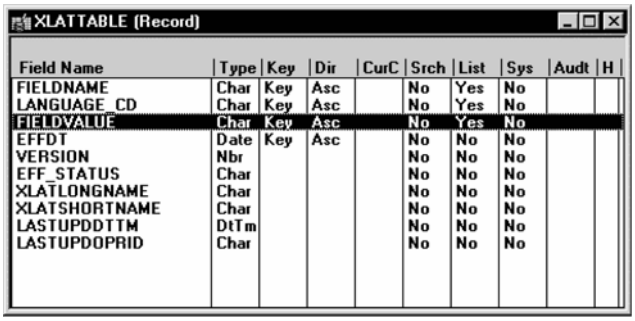
PeopleSoft provides objects that can be used to store a list of valid values for a field. Translate values are different from prompt values in the sense that translate values do not need an individual database table for storage. While each prompt list has its own record, all translate values are stored in one PeopleSoft tool table, called the XLATTABLE. Let's take a look at the fields from XLATTABLE to see how they store translate values (figure 6.13).

XLATTABLE has four database keys to store unique translate values. Let us describe the fields in XLATTABLE.

- FIELDNAME The actual field name in PeopleSoft for which translate values are stored.
- LANGUAGE_CD The language in which translate descriptions are stored,
- FIELDVALUE This is the actual translate value code.
- EFFDT The effective date for the translate value.
- VERSION PeopleSoft maintains version number for caching and upgrading translate values.
- EFF_STATUS Denotes the active or inactive status of the translate value

- XLATLONGNAME The location where a long description can be stored for the translate value.
- XLATSHORTNAME The location where a short description can be stored for the translate value.
- LASTUPDDTTM The last Date/Time the translate value was updated.
- LASTUPDOPRID The ID of the operator that last updated the translate value.

Navigation: Go →Open →Record →XLATTABLE



Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H
FIELDNAME	Char	Key	Asc		No	Yes	No		
LANGUAGE_CD	Char	Key	Asc		No	Yes	No		
FIELDVALUE	Char	Key	Asc		No	Yes	No		
EFFDT	Date	Key	Asc		No	No	No		
VERSION	Nbr				No	No	No		
EFF_STATUS	Char				No	No	No		
XLATLONGNAME	Char				No	No	No		
XLATSHORTNAME	Char				No	No	No		
LASTUPDDTTM	DtTm				No	No	No		
LASTUPDOPRID	Char				No	No	No		

Figure 6.13
Key structure of XLATTABLE

Translate values can be a maximum of four characters. So fields that are one to four characters long and used as codes can be stored in the translate table. Translate values are also effective-dated. The long and short descriptions can be different on different effective dates.

Translate values are associated with a field object. After you have created character fields, you attach translate values to them. Let's look at how we can attach translate values to a field (figure 6.14).

A field must be open in order to add, change, or delete its translate values. The buttons Add, Change, and Delete are used to add, change, and delete translate values respectively.

We also see the last updated Date/Time and the operator ID. Even though translate values are attached to a field object, this does not mean translate values are always used for a field when it is attached to a record definition. The translate values edit can either be turned on or off for a record field. This can be accomplished using the Edits tab in the Record Field Properties screen. Figure 6.15 illustrates the edits defined for fields from MY_USER_TABLE record.

`Xlat` in the Edit column indicates that the translate values edit has been turned on for the record field MY_USER_TYPE in the record definition MY_USER_TABLE. The same field can be used in another record definition without the translate values edit. Figure 6.16 illustrates how translate table edit is turned on for MY_USER_TYPE field in MY_USER_TABLE

Navigation: File →Object Properties →Translate Values tab (MY_USER_TYPE field is open)

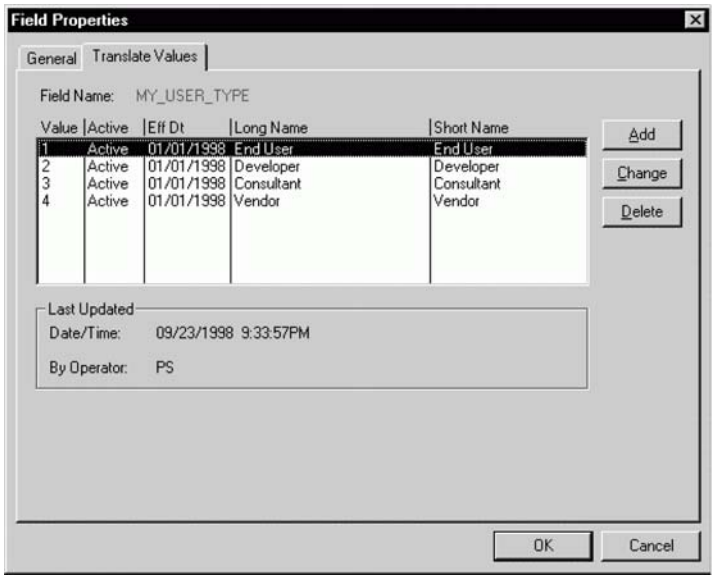


Figure 6.14 Translate Values for a field

Navigation: File →Open →Record →MY_USER_TABLE →View →Edit Display

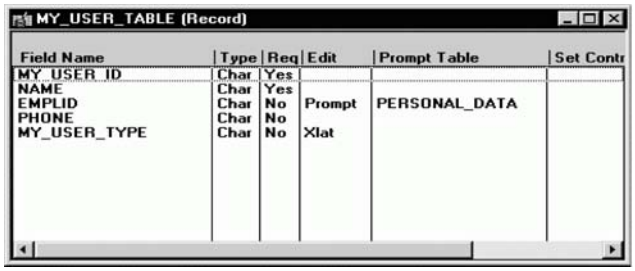


Figure 6.15 Edits display for a record definition

Translate values can appear on a panel either as an edit box or a drop-down list box. Drop-down list boxes can display the Translate long name and short name in the drop-down list. If an edit box is used, then the Translate short or long names can be displayed using a related display field. Figure 6.17 illustrates how translate values appear on a drop-down list within the online application.

Navigation: Highlight Field →Edit →Record Field Properties

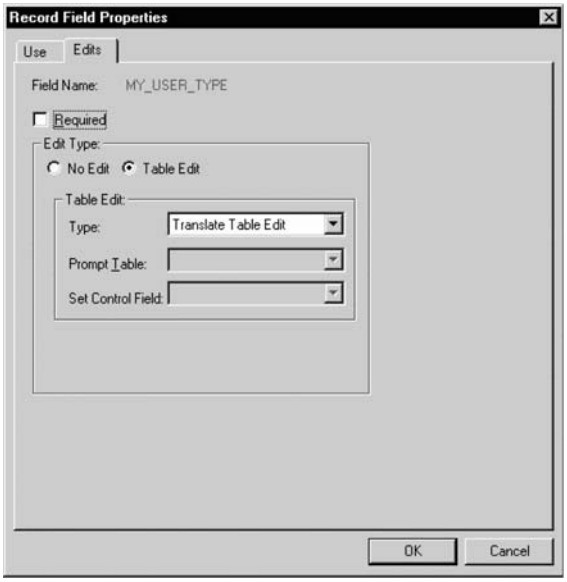


Figure 6.16
Record Field Properties—Edits tab

Navigation: Go →Problem Tracking →Setup →Users

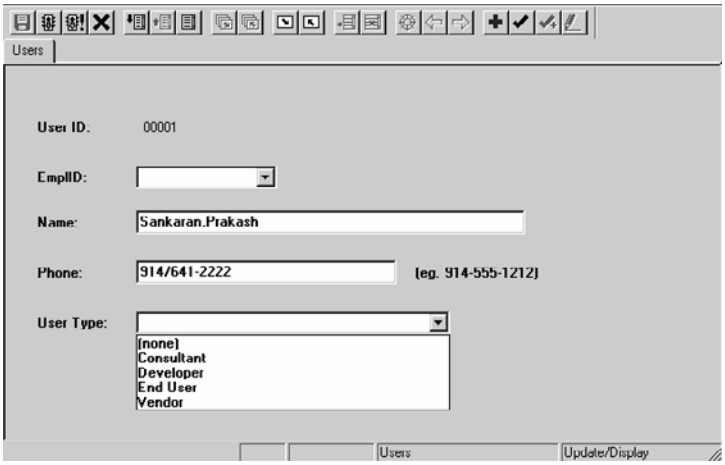


Figure 6.17
User panel with a drop-down list box for translate values

TIP Effective dates on translate tables work the same way as in prompt records. The value of the effective date in the application panel is compared to the effective date in XLATTABLE. All translate values for the record field that are active on or before the effective date in the application panel are included in the translate list.

NOTE Translate values are attached to field definitions. They can be included in records with or without the Translate Table Edit.

6.3 CREATING AND USING SEARCH RECORDS

Search records are used to control and limit the data displayed on a user panel. Search records can be either SQL tables or SQL views. When you create a panel group, you must specify the search record to be used. A search dialog box appears when the user tries to access the panel group, and search values may be entered into fields from search records designated as a search keys or alternate search keys. After the search keys are entered, any matching entries from the search record are displayed in a list box. Besides the search keys themselves, additional fields defined as list box items also appear in the list box.

Search records are specified in panel groups. A panel group, no matter how many panels it contains, needs a search record. The same search record can be specified in more than one panel group. For example, in PeopleSoft HRMS, the same search record is used for employee lookup. These panel groups are designed to look up employees either by using an employee ID or employee's last name. Search records are designed to limit the number of rows the user can access at any given point of time.

As we saw in chapter 4, a field has to be a database key in order to be a search key. Also, not all database keys are defined as search keys in PeopleSoft. This concept leaves us with four different types of search records:

- search records without any keys
- search records with search keys
- search records with search keys and database keys
- search records with `From` and `Through` search keys

6.3.1 Search records without keys

Sometimes, search records do not have any search keys or database keys. We see this only in instances when we do not want users to be prompted for any input. This means data selection can be performed without any input.

INSTALLATION table (from the PeopleSoft HRMS application which contains general information about the application) is an excellent example. It has fields such as the last employee ID assigned, default country, default currency code, commit counts, and so forth. This table has only one row and does not contain any database keys or search keys, making it possible to look up data from the INSTALLATION table

without any input. We may encounter other instances where we may want to bring up panels without providing any input. In such instances, we can use the INSTALLATION table as the search record. One example in the PeopleSoft Benefits Administration application is the BAS ACTIVITY panel. This panel retrieves all rows from the BAS_ACTIVITY table without any input.

6.3.2 Search records with search keys

Search keys are fields that appear on the input dialog box. Fields from the input dialog box determine the rows that appear as list box items. Fields from the Input dialog box may be display-only items on the panel.

If values are entered on all search keys, and an entry is found in the search record matching those values, the Application Processor brings up the panel directly without providing a list box. If values are entered only in certain fields, then a list box appears matching the items entered in the search fields. Fields defined as search keys are always defined as list box items, but list box items are not all search keys. List box items help the user identify data that will appear on the panel.

Let's take a look at how search records are built and how they control the data selection. To do so, we will take a look at our Problem Tracking application and see how users are added and updated. By looking at the underlying table, we can easily determine the keys that can be used in the search.

In figure 6.18, MY_USER_TABLE is used as a search record to add and update users. Here we are using the data record itself as the search record. We see that MY_USER_ID is the only search key field in this search record. When we perform a partial search on the MY_USER_ID field, the Application Processor retrieves all entries that match the value entered in MY_USER_ID field. NAME and EMPLID fields are used as alternate search keys. Alternate search keys work the same way as search keys, except that they are not unique.

In figure 6.19, we input a "0" in the MY_USER_ID field. We have all entries that matched the partial key search in the list box. Let us try the same search with no input.

Navigation: File →Open →Record →MY_USER_TABLE

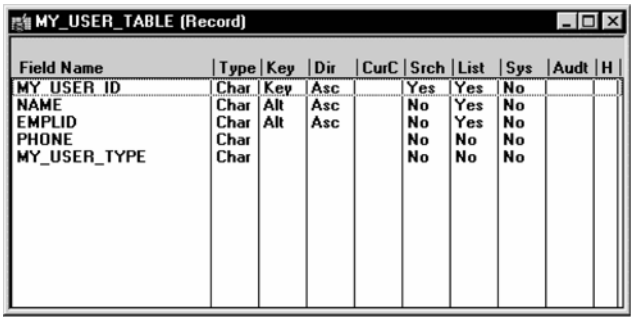


Figure 6.18 Search keys on a search record

Navigation: Go →Problem Tracking →Setup →Users

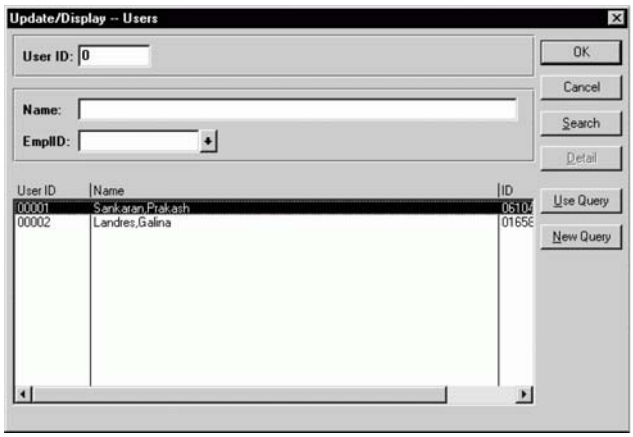


Figure 6.19
Search keys and list box items (partial search)

When no input is supplied, all rows from MY_USER_TABLE are listed in the list box (figure 6.20).

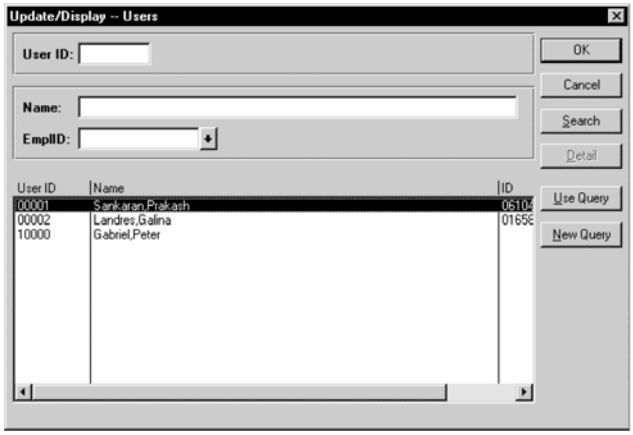


Figure 6.20
Search keys and list box items (no input)

NOTE The number of entries retrieved for the list box is limited to the first 256 rows returned by the Application Processor. Only the first 256 rows are displayed in the list box. When the number of entries exceeds the limit, message is issued to the user specifying that there are more than 256 rows that match the input supplied.

Now let's look at a search record that has more than one search key. In PeopleSoft HRMS the pay calendar table uses more than one search key as input.

Only the COMPANY field was entered in the search dialog box. The Application Processor retrieved all pay calendar entries that belong to company CCB. The message at the bottom in figure 6.21 indicates that more matching entries exist than those displayed in the list box.

Navigation: Go →Define Business Rules →Define Payroll Process →
Setup 2 →Pay Calendar Table

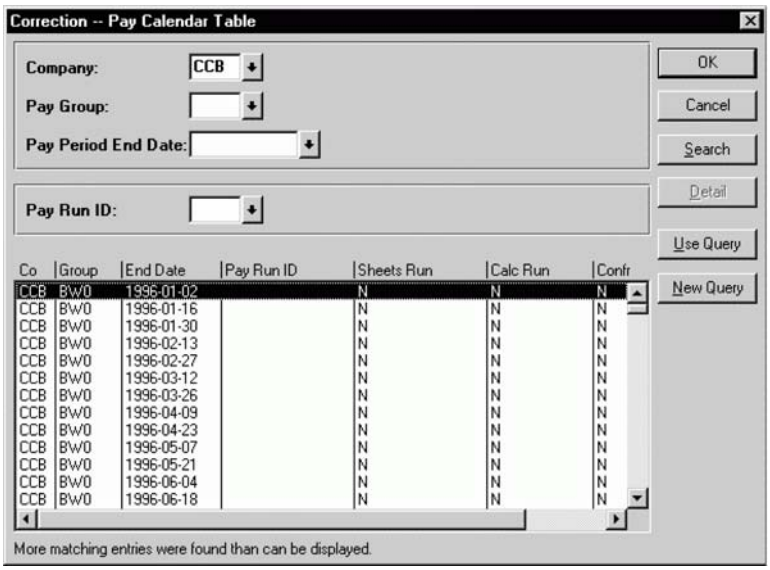


Figure 6.21 Search records with more than one search key

We can narrow the search by inputting a value into the PAYGROUP field. When a value is input into the COMPANY, PAYGROUP, and PAY_END_DT fields, the Application Processor attempts to match the values with entries in the search record. If a match is found, the Application Processor proceeds to display the panel without providing a list box.

RUN_ID field is an alternate search key field. It is a duplicate key that may list more than one entry in the list box. Let us take a look at the key definition for the PAY_CALENDAR record (figure 6.22).

Navigation: File →Open →Record →PAY_CALENDAR

PAY_CALENDAR (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt
COMPANY	Char	Key	Asc		Yes	Yes	No	
PAYGROUP	Char	Key	Asc		Yes	Yes	No	
PAY_END_DT	Date	Key	Asc		Yes	Yes	No	
RUN_ID	Char	Alt	Asc		No	Yes	No	
PAY_OFF_CYCLE_CAL	Char				No	No	No	
AGGR_ID	Char				No	No	No	
PAY_BEGIN_DT	Date				No	No	No	
CHECK_DT	Date				No	No	No	
PERIOD_WEEKS	Nbr				No	No	No	
PAY_PERIOD	Char				No	No	No	
PAY_PDS_PER_YEAR	Nbr				No	No	No	
ACCRUAL_PCT	Nbr				No	No	No	

Figure 6.22
Search record with search keys

6.3.3 Search records with search keys and database keys

Some search records use search keys as well as database keys that are not defined as search keys in the search. In the PeopleSoft HRMS application, the record EMPL_COMP_SRCH is used to look up employee details based on the company for which they work. All payroll balances and tax data panels use this as the search record. This search record, in addition to two search keys, also has a database key that controls the search. This database key is the OPRCLASS field which narrows down the search based on the department security setup for the operator.

Figure 6.23 illustrates the key definition for EMPL_COMP_SRCH record. The Application Processor automatically includes certain fields in the search criteria. Two such fields, the OPRID and the OPRCLASS fields, when included in a search record, are automatically included in the search.

Navigation: File →Open →Record →EMPL_COMP_SRCH

EMPL_COMP_SRCH (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H
EMPLID	Char	Key	Asc		Yes	Yes	No		
COMPANY	Char	Key	Asc		Yes	Yes	No		
OPRCLASS	Char	Key	Asc		No	No	No		
ACCESS_CD	Char				No	No	No		
NAME	Char	Alt	Asc		No	Yes	No		
NAME_AC	Char				No	No	No		
LAST_NAME_SRCH	Char				No	No	No		
NID_COUNTRY	Char				No	No	No		
NATIONAL_ID_TYPE	Char				No	No	No		
NID_DESCRSHORT	Char				No	No	No		
NATIONAL_ID	Char				No	No	No		

Figure 6.23
Search records with search keys and database keys

PeopleSoft uses OPRID and OPRCLASS fields to provide application security. Department security in a PeopleSoft HRMS application is controlled using these two fields. Security definitions are attached to the OPRCLASS field, while search views include the tables that store these security definitions. When the Application Processor

automatically includes OPRCLASS in the search, security definitions for that particular OPRCLASS secure the data which the user can access. Each PeopleSoft user, or OPRID, is attached to an operator class or OPRCLASS.

Since OPRCLASS and OPRID fields are available as system variables during the panel session, it makes sense to use them automatically in the search. In figure 6.24, we can see that OPRCLASS does not appear in the input dialog box.

Navigation: Go →Compensate Employees →Maintain Payroll Data U.S. →Use → Employee Tax Data

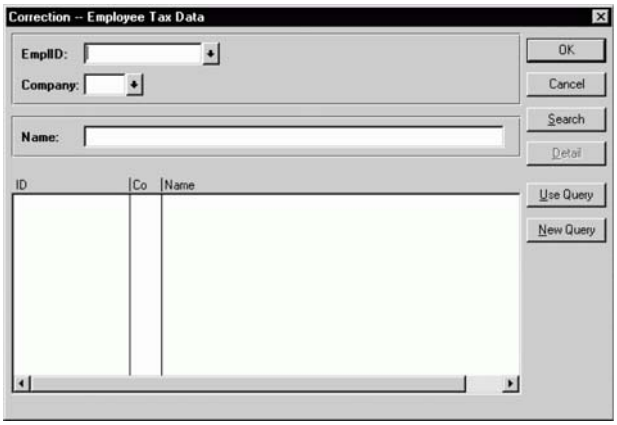


Figure 6.24
Search records that use search keys and database keys

6.3.4 Search records with From and Through search keys

PeopleSoft 7 introduced a new feature that allows `From` and `Through` search keys to be defined in search records. All rows that are greater than or equal to the `From` value and lesser than or equal to the `Through` value are fetched from the search record. For example, we can define the MY_PROBLEM_STATUS field from MY_PROBLEM_TRKG record as a `From` or `Through` search key.

The MY_PROBLEM_STATUS field is defined as an alternate search key. Figure 6.25 illustrates the results returned in the list box when a value of `3` is entered in that field.

The Application Processor found an exact match and displayed the application panel directly without providing a list box. Let's define the MY_PROBLEM_STATUS field as a `Through` search key in MY_PROBLEM_TRKG record (figure 6.26).

Under the Edits tab in Record Field Properties, we can define the MY_PROBLEM_STATUS field as a `Through` search field. Figure 6.27 illustrates the rows retrieved using the same input value of `3`.

The Application Processor retrieves all entries from MY_PROBLEM_TRKG record which have a value less than or equal to `3` in the MY_PROBLEM_STATUS field. Because MY_PROBLEM_STATUS field is defined as a `Through` search key, more than one entry is found to match the entries in the search record. The Application Processor provides a list box with all matching entries.

Navigation: Go →Problem Tracking →Tracking →Track Problems →Update/Display

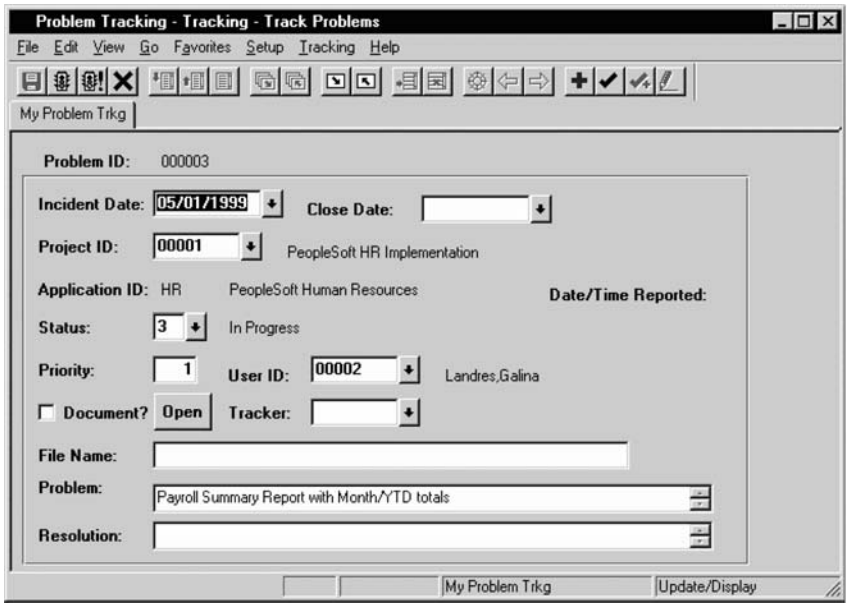


Figure 6.25 Search key—exact match

Navigation: File →Open →Record →MY_PROBLEM_TRKG →
HighLight MY_PROBLEM_STATUS → Edit → Record Field Properties

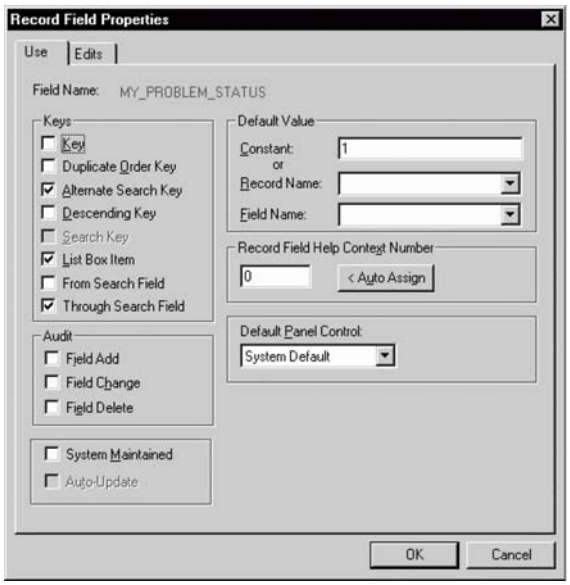


Figure 6.26 Search record with Through search key

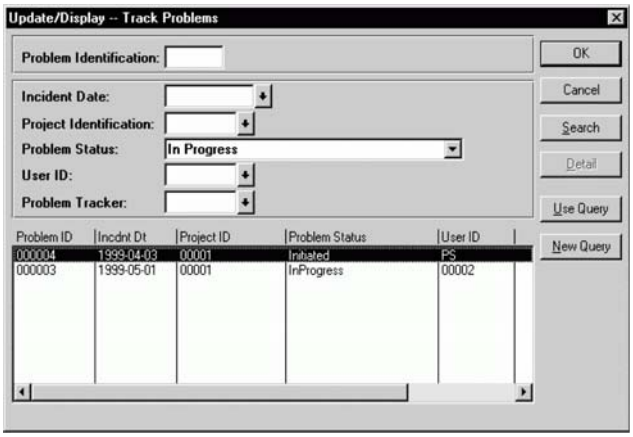


Figure 6.27
Through search key results

6.3.5 Create and define search records

To create and define search records, we must:

- create a record definition for the search record
- define search keys and list box items
- build the search record definition in the database as a table or view
- attach search records to panel groups

Let's create and define a search record, using the Problem Tracking application. We will add and update incidents and problems using the application. The database table that is being updated is the MY_PROBLEM_TRKG table. We already created the MY_PROBLEM_TRKG record prior to this. Let's look at the search keys for this table and determine if we can use it as a search record.

Navigation: File →Open →Record →MY_PROBLEM_TRKG

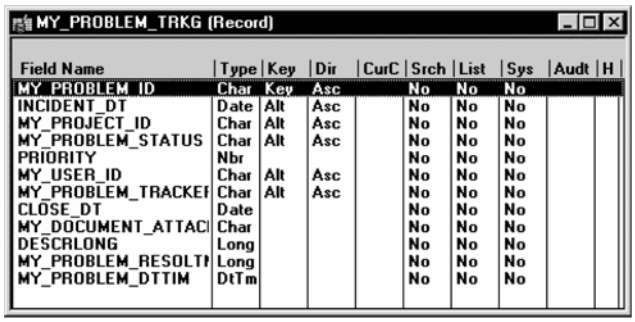


Figure 6.28
Record definition showing database keys

MY_PROBLEM_ID is the only unique key field in MY_PROBLEM_TRKG record. We define this field as a search key. In addition to defining search fields, a number of other fields can be used as alternate search keys in this table. You can see the final search key and list box definition in figure 6.29.

Navigation: File →Open →Record →MY_PROBLEM_TRKG

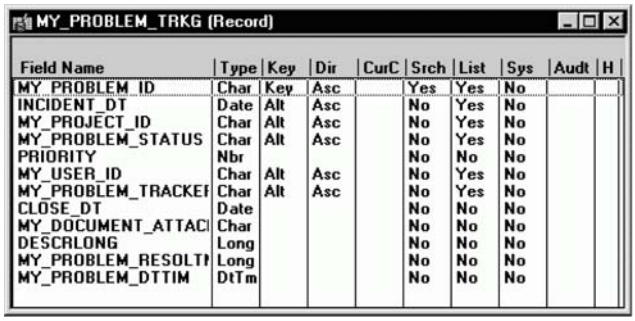


Figure 6.29
Record definition showing search keys and list box items

Let's look at how we defined the search keys and the list box items.

Navigation: Highlight Field →File →Edit →Record Field Properties →Use Tab

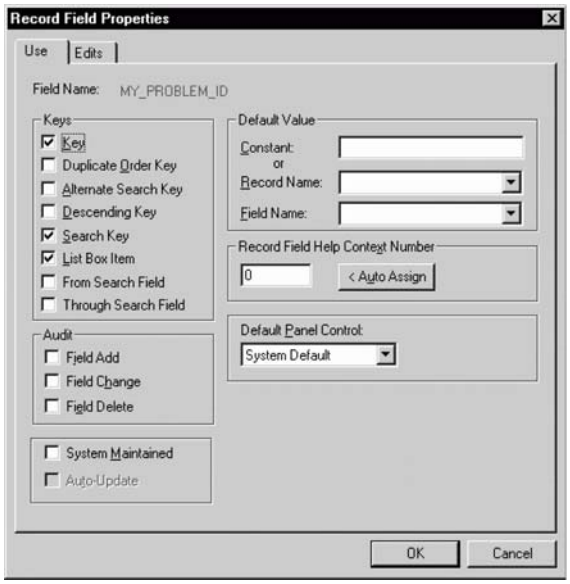


Figure 6.30
Define Record Field properties

All the fields can be highlighted one at a time, and the properties can be defined under the Use tab.

Our next step is to build the search record in the database. If the search record is an SQL table and the table already has data, we do not want to recreate the table. If the search record definition is an SQL view, however, then we can go ahead and recreate the SQL view any time.

In this next step, we attach the search record to the panel group definition using the panel group MY_PROBLEM_TRKG. After opening the panel group, we define the search record for that panel group under the Use tab.

Navigation: File →Open →Panel Group →MY_PROBLEM_TRKG →File → Object Properties

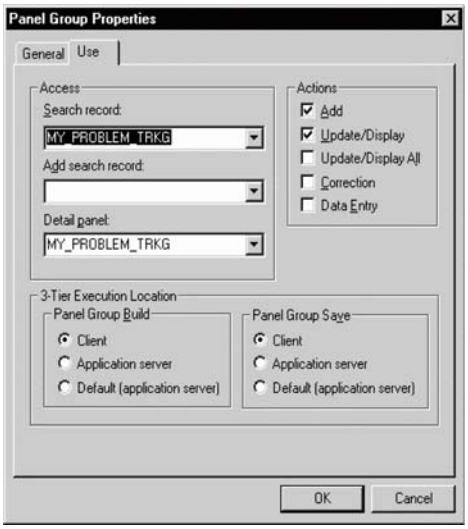


Figure 6.31
Panel Group Properties—Use tab

We attach two search records to the panel group definition. Add Search Record is used for `Add` action and the regular search record is used for `Update/Display`, `Update/Display All`, and `Correction`. If the Add search record is not specified, the regular search record is used for `Add` action as well.

When the panel group is attached to a menu item, the search record can be overridden at that time using the Menu Item Properties screen to accomplish this task. Figure 6.32 illustrates how we can override search records defined in panel groups. By turning on the Override checkbox, we can define the override search record on the drop-down box located to the right of the checkbox.

Navigation: Edit →Menu Item Properties (PROBLEM_TRACKING menu is open)

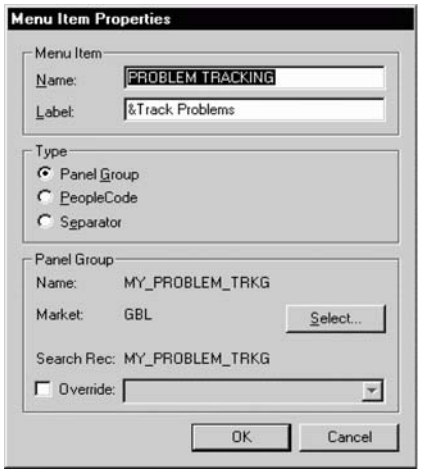


Figure 6.32
Menu Item Properties

NOTE Search keys are database keys as well. Alternate search keys are defined as non-unique database keys.

NOTE Values input in search fields are matched with entries in the search record. When only one entry is found matching the input, the Application Processor displays the application panel without providing a list box. When the number of entries found matching the input is more than one, then a list box is provided with the matching entries.

6.4 WORKING WITH DERIVED/WORK RECORDS

Derived/Work records are used as temporary storage records during Application Processing. Derived records can be used for a number of other purposes such as:

- counters and totals
- push buttons and command fields
- display messages
- temporary holding fields
- to define dynamic prompt records

The most common uses of Derived/Work records, however, are as command fields, counter fields, and total fields. Derived records are also called work records. For the purpose of this section, we will refer to them as derived records.

Derived records are record definitions which are relevant only to the online application. They do not exist in the database as an object. Only records defined as SQL tables and views are stored in the database. Derived records are populated only during

the panel session. Once the panel is cancelled, data stored in the derived record is lost. Derived records may be shared across application panels. Because they are not stored in the database, they can also be used across multiple panel sessions at the same time. Only the field placed on the application panel remains in memory. The other fields from the derived record are not available in the panel buffer. Let's look at a few examples of derived records in use.

6.4.1 Using derived records as counters and totals

Fields in derived records can be used as counters and totals. The fields can either be displayed on the panel or hidden and used only for calculations. PeopleCode events can be attached to these derived fields just like any other record field. We have a panel that shows totals by status in our Problem Tracking application. Figure 6.33 illustrates an application panel from our Problem Tracking application that makes use of derived records.

Navigation: Go →Problem Tracking →Tracking →Problems—Totals by Status

Figure 6.33 Application panel using a derived record

NOTE Fields from derived records can be used across multiple panels. Only the fields that are placed in the application panel are available in the panel buffer. Other fields from the derived record are not available for access. One instance of a derived field in a panel session does not interfere with another instance of the same field in another panel session.

The panel in figure 6.33 shows the total number of problems by status code. We can use a derived field to compute the total number of problems/incidents tracked using our application by following these steps:

- add a derived field that can hold the grand total to MY_DERIVED record
- place the derived field in MY_TRKG_STATUS panel
- create a PeopleCode event to compute the grand total

These three simple steps will give us the total number of problems/incidents tracked in the system. In PeopleSoft, already defined fields can be reused. We can add any field that accommodates totals to our derived record. We do not have to necessarily create a new field in the system. Let's see how we can find such a field in the system. First, we open the MY_DERIVED record definition. We then retrieve a list of fields (defined in the system) that start with the letters TOT. Figure 6.34 illustrates this process.

We can use any field already available in the system. We choose TOTAL_COUNT field for this purpose. In figure 6.35, we can see the TOTAL_COUNT field included in MY_DERIVED record definition.

In our example, we use a record defined as an SQL view to compute the total number of problems by problem status. Let's take a look at the definition for the SQL view used in our example (figure 6.36).

Navigation: Insert/Field (MY_DERIVED Record Definition Open)

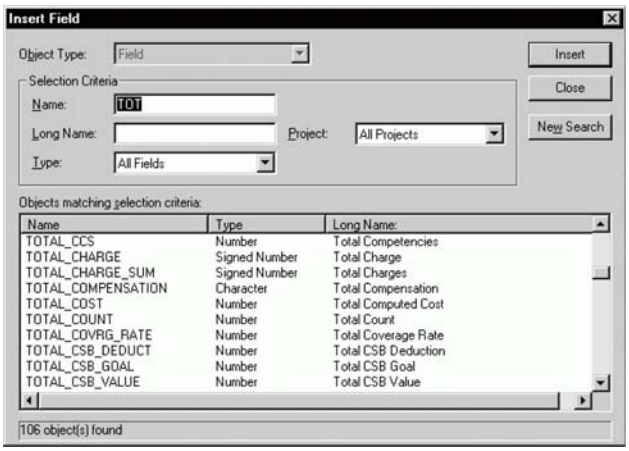


Figure 6.34
Fields defined in the system

Navigation: File →Open →Record →MY_DERIVED

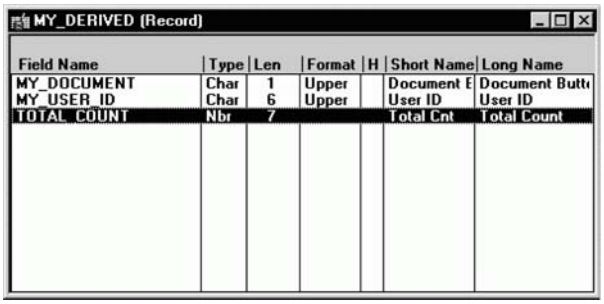


Figure 6.35
Adding a derived field

Navigation: File →Open →MY_TRKG_STATUS

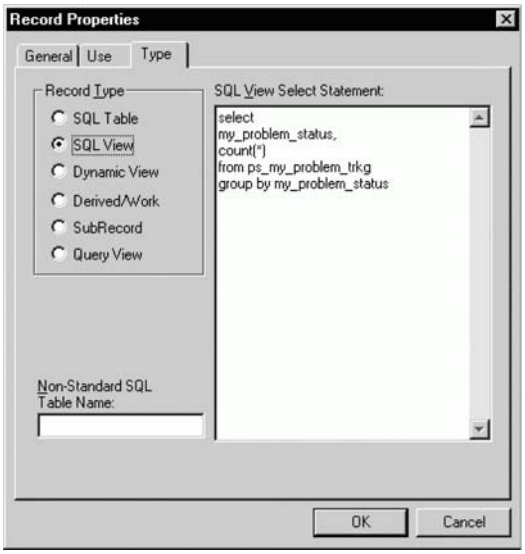


Figure 6.36
SQL view definition

MY_TRKG_STATUS has a TOTAL_COUNT field that contains totals by problem status. Our goal is to produce a grand total of all problems/incidents tracked in the system.

In order to compute the grand total, the individual totals from the scroll bar have to be added together. We can populate the computed grand total into the TOTAL_COUNT field from MY_DERIVED record. First we need to place the field from the derived record on our panel.

TIP The same field definition can be used in multiple record definitions. A field from different record definitions can be used in the same application panel as well.

In figure 6.37, we see an additional field in the bottom of the panel. This is the physical location of the TOTAL_COUNT field from MY_DERIVED record where the field is displayed when the panel is brought up.

Navigation: File →Open →Panel →MY_TRKG_STATUS

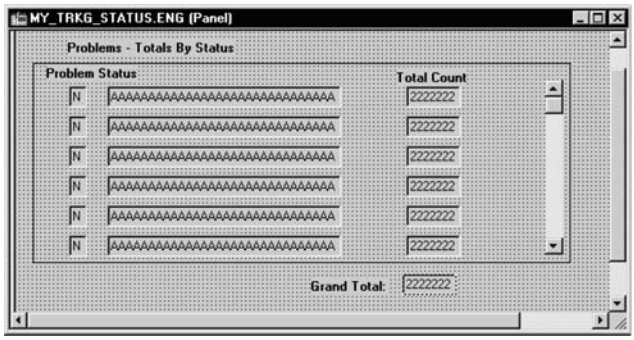


Figure 6.37
Panel definition with derived record field

Because we are using this field as the grand total, we need only one instance of this field. If we place it below the scroll bar in the panel field layout, we will have many instances of this field. (Figure 6.38 illustrates how the TOTAL_COUNT field from MY_DERIVED record is placed above the scroll bar.)

Now let's create a `RowInit` PeopleCode event on MY_TRKG_STATUS record to populate the grand total field. As rows are loaded on the scroll bar, we can add the totals from each row in the scroll bar to the TOTAL_COUNT field in MY_DERIVED record. We access level zero fields from inside the scroll bar by referring to them with

Navigation: Layout →Order (MY_TRKG_STATUS panel is open)

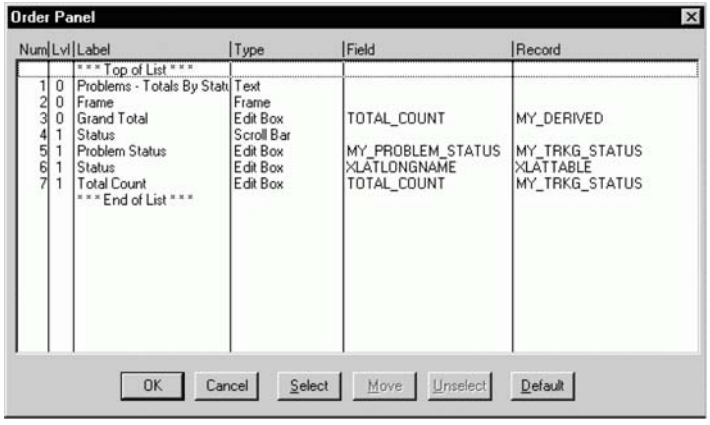


Figure 6.38
Panel field layout for MY_TRKG_STATUS panel

the proper record name prefix. (Figure 6.39 illustrates the PeopleCode program that computes the grand total.)

Navigation: Highlight TOTAL_COUNT →View →View PeopleCode
(MY_TRKG_SATUS record is open)

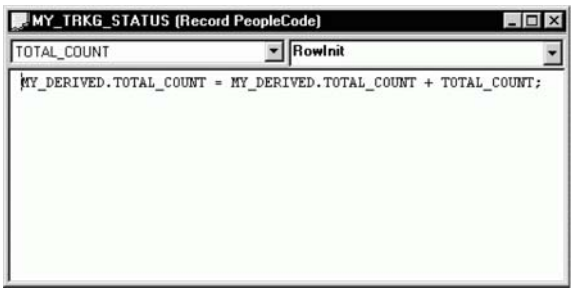


Figure 6.39
PeopleCode which computes grand total

We refer to the field in MY_DERIVED record as MY_DERIVED.TOTAL_COUNT. Notice that the TOTAL_COUNT field from MY_TRKG_STATUS record appears without any record prefix. PeopleCode can refer to all fields from one record without using any record prefix. (Figure 6.40 illustrates the application panel with the grand total field.)

Navigation: Go →Problem Tracking →Tracking →Problems—Totals By Status

Figure 6.40 Application panel using derived record

6.4.2 Using derived records to display messages

In our next example, we see how a derived field is used as a message text field. The JOB_DATA1 panel in PeopleSoft's HRMS application contains the JOB_PANEL_MSG field from the DERIVED_HR record. This field helps the user determine whether the row being displayed is a current, a future, or a historical row. In figure 6.41, this display message appears in the application panel.

Navigation: Go →Administer Workforce U.S. →Use →Job Data →Correction

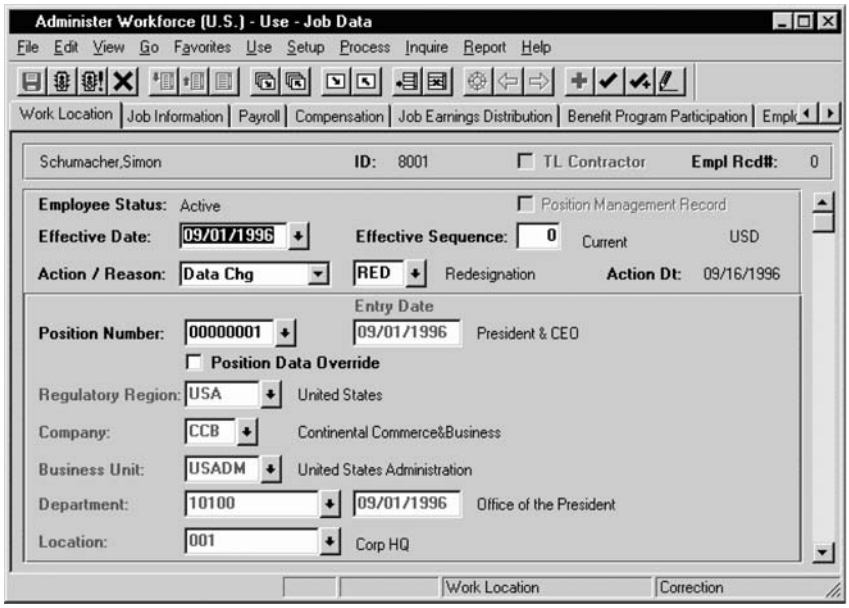


Figure 6.41 Job Data panel using a derived field as a message field

Notice the message "Current" to the right of the effective sequence field. This message is populated into the derived field using a PeopleCode event. The Application Processor executes a `RowInit` PeopleCode event as rows are loaded into the scroll bar. The PeopleCode program performs logic to determine what message should be displayed in the derived field.

In the next section, we will learn how to use push buttons in PeopleSoft. In the process of doing so, we will also learn another application that uses derived records.

6.5 USING PUSH BUTTONS

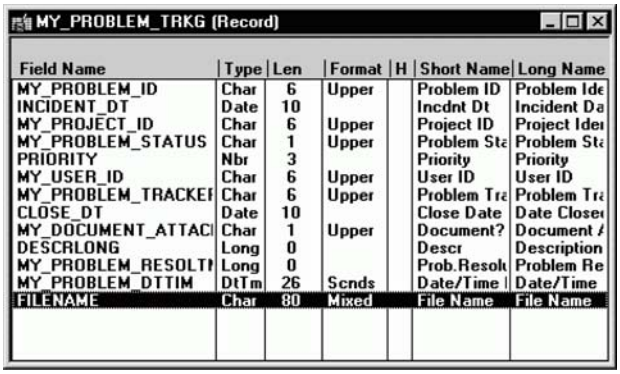
Push buttons are panel fields which can be activated to execute events. Push buttons are also called command buttons. Push buttons can be defined as command buttons, secondary panels or processes. When push buttons are defined as commands, they

execute a `FieldChange` PeopleCode event attached to the panel field. When they are defined as secondary panels, they activate a secondary panel. When they are defined as a process, they execute a batch process.

Let us look at our Problem Tracking application again. In MY_PROBLEM_TRKG panel, we added a panel field called MY_DOCUMENT from MY_DERIVED record. This derived field can be used to display documents associated with the problem being tracked.

In our example, we use Microsoft Word as the document type to document problems. We can add a field to the MY_PROBLEM_TRKG record which holds the full path and filename for the Microsoft Word document. We use the field, FILENAME, which already exists in the system (figure 6.42).

Navigation: Go →File →Open →Record →MY_PROBLEM_TRKG


MY_PROBLEM_TRKG (Record)

Field Name	Type	Len	Format	H	Short Name	Long Name
MY_PROBLEM_ID	Char	6	Upper		Problem ID	Problem Ide
INCIDENT_DT	Date	10			Incdnt Dt	Incident Da
MY_PROJECT_ID	Char	6	Upper		Project ID	Project Ider
MY_PROBLEM_STATUS	Char	1	Upper		Problem Sta	Problem Sta
PRIORITY	Nbr	3			Priority	Priority
MY_USER_ID	Char	6	Upper		User ID	User ID
MY_PROBLEM_TRACKEI	Char	6	Upper		Problem Tra	Problem Tra
CLOSE_DT	Date	10			Close Date	Date Closer
MY_DOCUMENT_ATTACI	Char	1	Upper		Document?	Document A
DESCRLONG	Long	0			Descr	Description
MY_PROBLEM_RESOLTI	Long	0			Prob.Resolt	Problem Re
MY_PROBLEM_DTTIM	DtTm	26	Scnds		Date/Time	Date/Time
FILENAME	Char	80	Mixed		File Name	File Name

Figure 6.42
Adding a field to a record definition

Figure 6.43 illustrates how the push button is added into the MY_PROBLEM_TRKG panel.

First, we define panel field properties for the push button. Two tabs exist where we define the properties for a push button.

Under the Record tab, we define whether the push button is a command, process or, secondary panel. In this example, the push button is a command button. We specify MY_DERIVED as the record and MY_DOCUMENT as the field for the push button.

Under the Label tab, we specify a label for the push button. We also define the font for the label under this tab. Figure 6.44 and 6.45 illustrates the Panel Field Properties screen for the push button.

After saving the panel definition we are ready to take a look at the push button as it appears in the application panel (figure 6.46).

Navigation: Insert →Push Button (MY_PROBLEM_TRKG panel is open)

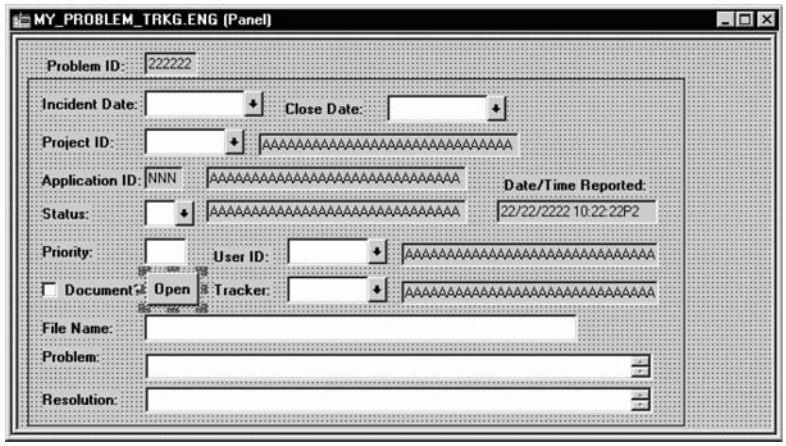


Figure 6.43 Inserting a push button in an application panel

Navigation: Edit →Panel Field Properties (MY_DOCUMENT field is highlighted)

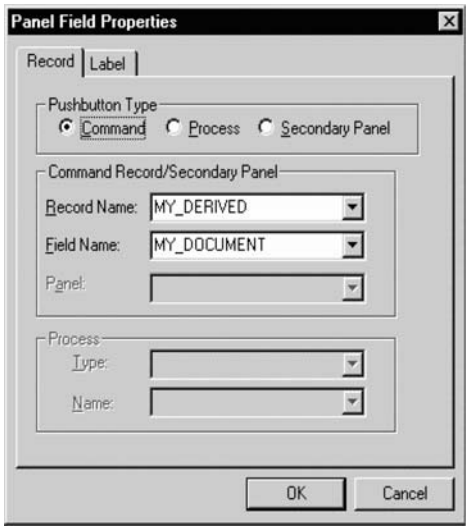


Figure 6.44 Push button properties—Record tab

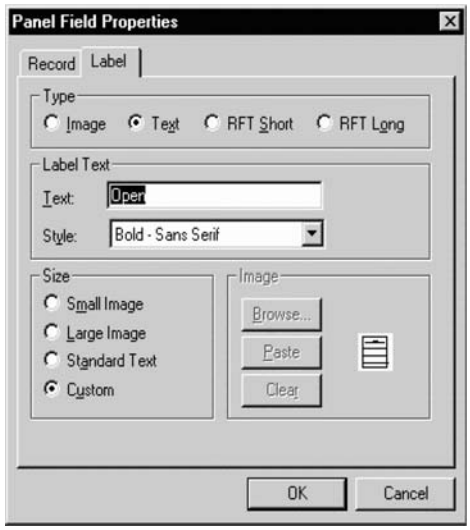


Figure 6.45 Push button properties—Label tab

Navigation: Go →Problem Tracking →Tracking →Track Problems

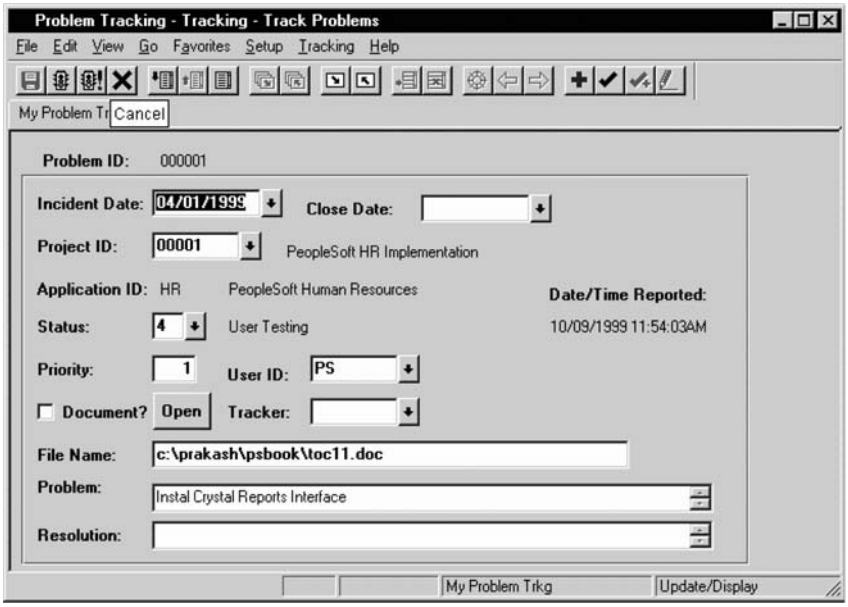


Figure 6.46 Push button field in an application panel

As soon as the user activates the push button, the PeopleCode program attached to MY_DOCUMENT field's `FieldChange` PeopleCode event is executed. This PeopleCode event executes Microsoft Word along with the full path and filename. Figure 6.47 illustrates the PeopleCode event which accomplishes this task.

Navigation: File →Open →MY_DERIVED →View →View PeopleCode Display

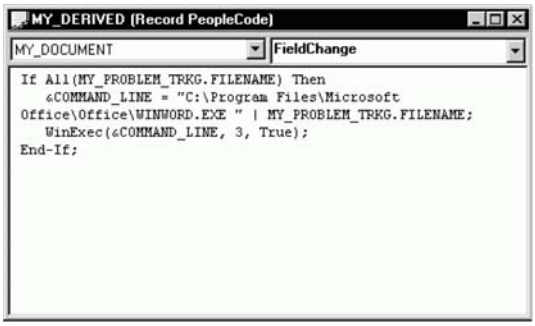


Figure 6.47 PeopleCode event attached to a push button

This PeopleCode event uses the `WinExec` function. This is a synchronous operation. The Application Processor will wait for the user to view and close the document before it continues to process other events.

NOTE When push buttons are used as command buttons, they are always associated with a `FieldChange` PeopleCode event. When push buttons are used as secondary panels, it is important to note that the actual secondary panel has to be placed after the push button field in the panel field layout.

KEY POINTS

1. Prompt records are used to provide a list of valid values for a panel field.
2. Prompt processing is performed with the help of database and search keys on the prompt record.
3. Translate values also provide a list of valid values for character fields one to four characters in length. All translate values in the system are stored in one record called the XLATTABLE.
4. Search records are used to control and limit the data displayed on a user panel.
5. Search fields and alternate search fields on a search record are displayed in a dialog box during search processing.
6. When the Application Processor finds an exact match for values entered in search fields, it displays the panel without providing a list box. When the number of rows that match the values entered in search fields is more than one, the Application Processor provides a list box with matching entries.
7. Derived/Work records are used in applications which require Total fields, Push Button fields, message fields, dynamic prompt records, etc.
8. Push Buttons are used as commands, processes, or secondary panels.

CHAPTER 7

Advanced panel design features

This chapter covers some advanced features used when building a panel in PeopleSoft. Panels serve as user interfaces to the application. A number of added features in PeopleSoft 7 help the developer build powerful application panels that can perform a variety of tasks.

7.1 WORKING WITH SCROLL BARS

It is safe to say that over half the panels in a PeopleSoft application are built with scroll bars. In some cases, scroll bars are used to display multiple rows of data from the same Record definition. This is the simplest way to describe it. Scroll bars are also used to display records that have a Parent/Child relationship and to maintain historical data using effective dates.

Scroll bars have counts that determine the number of rows displayed on the scroll and define the panel buffer information. For example, if one scroll bar is on a panel, then the level above the scroll bar is referred to as level 0 while the scroll itself is referred to as level 1. Several "level 1" scroll bars can exist on the same panel, most probably displaying data from multiple record definitions. Each one of these "level 1" scroll bars can have scroll bars below them.

Scroll bars from the same record definition can span multiple panels (also known as panel groups). Each of these panels contains a group of fields from the same Record definition. Fields are grouped by functions within these panels. JOB DATA is an example of a group of panels that display fields from the same record definition across multiple panels on the same scroll level.

Scroll bars—usually hidden scrolls built from work records—can also be used as work scrolls. Sometimes work scrolls are used to update an SQL table. The scroll bar contains fields from an SQL view and, during save time, those fields update the underlying SQL table. The POSITION_DATA group of panels in PeopleSoft HRMS performs updates on incumbents from changes made to positions.

Scroll bars can be used in a panel to:

- display multiple rows of data from one record definition uniquely identified by one or more key fields
- display effective-dated rows
- display records with parent/child relationships
- act as work scrolls

7.1.1 Multiple rows on scroll bars

By simply building a panel with a single scroll bar and assembling the required fields from the record definition we can use scroll bars to display multiple rows of data from the same record definition. We have an example in our Problem Tracking application built in chapter 6. This panel shows the total number of problems by problem status.

Figure 7.1 shows a panel that displays the problem status and the total number of problems. The panel has a related display field that shows the translate description of the problem status field. The Grand Total field is placed on level zero before the scroll bar in the order of fields because only one occurrence of the grand total field can exist on the panel. Inside the scroll bar, however, we have multiple occurrences of the problem status field showing on the scroll bar.

Navigation: File →Open →Panel →MY_TRKG_STATUS

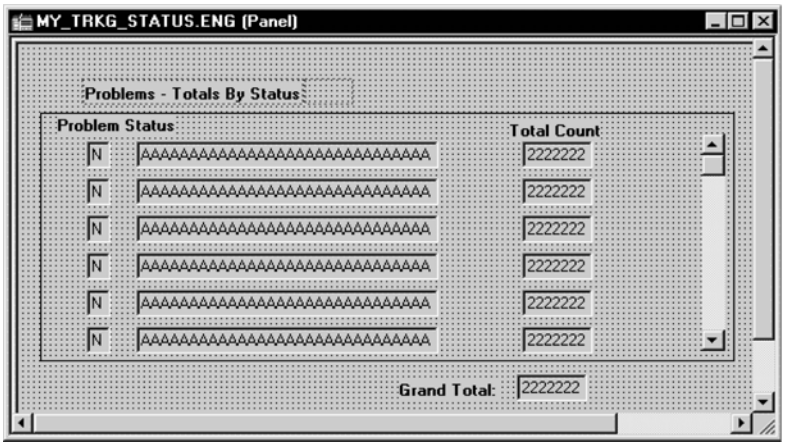


Figure 7.1 Panel displaying multiple rows on the scroll bar

NOTE On a scroll bar, only fields from one record definition can exist unless they are related display or derived fields. This limitation forces us to create multiple scroll bars or build a group of panels.

Given that a scroll bar can contain fields from only one record definition, we assemble record definitions, which are child records, on a lower level scroll. The description field in figure 7.1 is a related display field. All related display fields have an associated control display field. In this example, the problem status field controls the value displayed as the description. Let's look at the panel field layout to better understand what we've just described.

Looking at the panel fields layout (figure 7.2) we notice three things described in the previous section.

- All the fields below the scroll bar are either from the MY_TRKG_STATUS record definition or they are related display fields such as XLATLONGNAME from the XLATTABLE.
- The MY_PROBLEM_STATUS field from the MY_TRKG_STATUS record definition is placed before the XLATLONGNAME field from XLATTABLE. This is because MY_PROBLEM_STATUS field is the Display Control field, and XLATLONGNAME is the Related Display field.
- The TOTAL_COUNT field from the MY_DERIVED record definition is placed before the scroll bar. This will help us show one grand total of all the problems/incidents in our application. Of course, PeopleCode has to be used to show the grand total field in this work field as illustrated in figure 6.39.

Navigation: Layout → Order (MY_TRKG_STATUS panel is open)

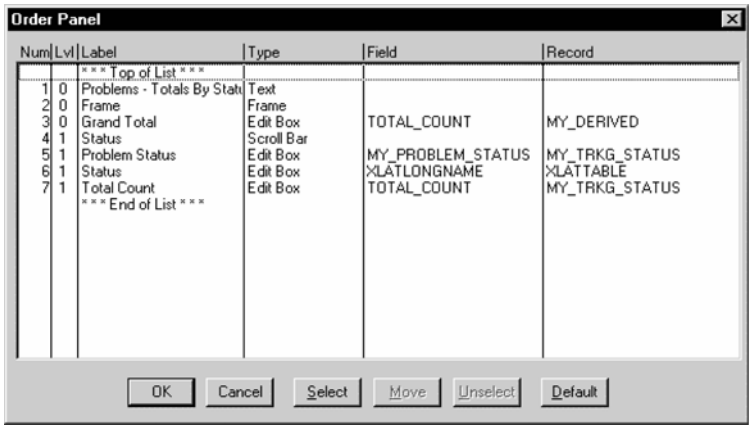


Figure 7.2 Panel fields layout

NOTE A scroll bar has an occurs level and an occurs count. Occurs level is the scroll level number. Occurs count is the number of rows that can be displayed when the panel is brought up. Fields are placed after the scroll bar in the panel field layout so that multiple occurrences of data can be displayed on the scroll bar. Any field that should be displayed only once on the panel must be placed before or above the scroll bar in the panel field layout.

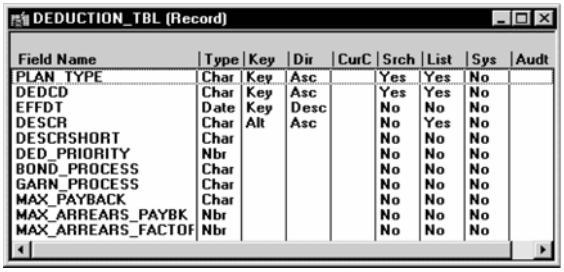
7.1.2 Parent and child records on scrolls

In our Problem Tracking application, we do not have a parent and child record assembled on a panel with a scroll bar. Let's look instead at the PeopleSoft HRMS application where we have examples we can review to learn more about Parent and Child records on a scroll. One good example is the deduction table, which contains three different Child records. The deduction table is populated using a group of panels. Let us look at the key structure of the DEDUCTION_TBL (figure 7.3) and its child record, the DEDUCTION_CLASS table (figure 7.4).

DEDUCTION_TBL has three key fields. If we look at the DEDUCTION_CLASS table, we notice that it has two more keys than the DEDUCTION_TBL.

From figure 7.4 we can deduce that, for a row of data in the DEDUCTION_TBL, we can possibly have multiple rows of data in the DEDUCTION_CLASS table. Of course, we assume that these record definitions are going to be viewed and updated using the same panel group. These tables are actually updated using a group of five panels in the PeopleSoft HRMS system.

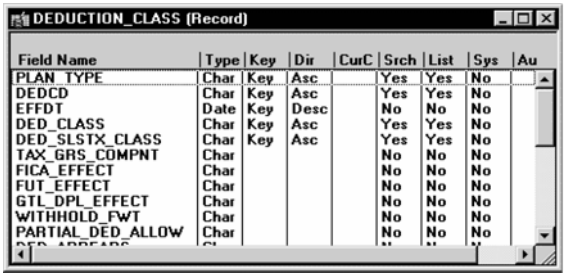
Navigation: File →Open →Record →DEDUCTION_TBL


DEDUCTION_TBL (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt
PLAN_TYPE	Char	Key	Asc		Yes	Yes	No	
DEDCD	Char	Key	Asc		Yes	Yes	No	
EFFDT	Date	Key	Desc		No	No	No	
DESCR	Char	Alt	Asc		No	Yes	No	
DESCRSHORT	Char				No	No	No	
DED_PRIORITY	Nbr				No	No	No	
BOND_PROCESS	Char				No	No	No	
GARN_PROCESS	Char				No	No	No	
MAX_PAYBACK	Char				No	No	No	
MAX_ARREARS_PAYBK	Nbr				No	No	No	
MAX_ARREARS_FACTOF	Nbr				No	No	No	

Figure 7.3
Key structure of a parent record

Navigation: File →Open →Record →DEDUCTION_CLASS


DEDUCTION_CLASS (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Au
PLAN_TYPE	Char	Key	Asc		Yes	Yes	No	
DEDCD	Char	Key	Asc		Yes	Yes	No	
EFFDT	Date	Key	Desc		No	No	No	
DED_CLASS	Char	Key	Asc		Yes	Yes	No	
DED_SLSTX_CLASS	Char	Key	Asc		Yes	Yes	No	
TAX_GRS_COMPNT	Char				No	No	No	
FICA_EFFECT	Char				No	No	No	
FUT_EFFECT	Char				No	No	No	
GTL_DPL_EFFECT	Char				No	No	No	
WITHHOLD_FWT	Char				No	No	No	
PARTIAL_DED_ALLOW	Char				No	No	No	

Figure 7.4
Key sctructure of a child record

Because DEDUCTION_CLASS is a child record to DEDUCTION_TBL, they can be placed on panels which contain scroll bars. There can be multiple rows of data in the DEDUCTION_CLASS table for a unique combination of PLAN_TYPE, DEDCD, and EFFDT fields from the DEDUCTION_TBL. The first three keys in both the tables have the same value.

Let's look at the panels that contain these record definitions (figure 7.5). DEDUCTION_TABLE2 panel contains fields from DEDUCTION_TBL and DEDUCTION_CLASS record definitions. DEDUCTION_TBL is on the level one scroll bar, and DEDUCTION_CLASS is on the level two scroll bar.

For a particular deduction code with an effective date, multiple deduction classes can be entered using the level two scroll bar. If there are multiple records on the level one scroll bar with different effective dates, then multiple rows in DEDUCTION_CLASS table are associated with every effective-dated row in DEDUCTION_TBL.

Navigation: File →Open →Panel →DEDUCTION_TABLE2

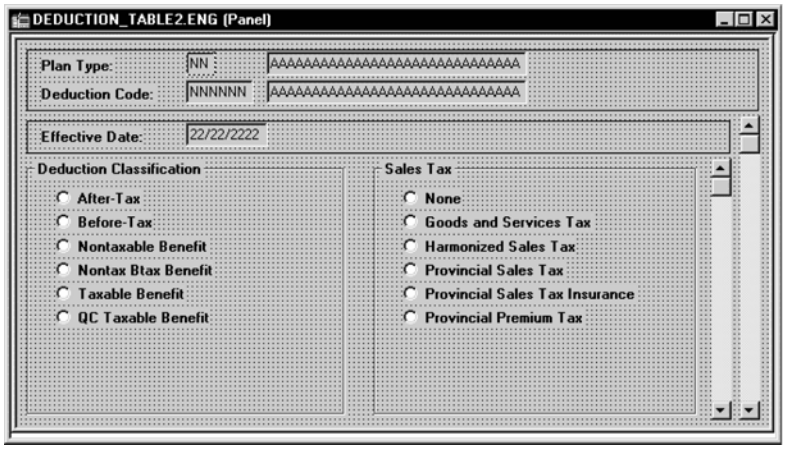


Figure 7.5 Panel with parent and child records on scrolls

Look at the panel layout and check the order in which the fields were laid out when the panel was built (figure 7.6).

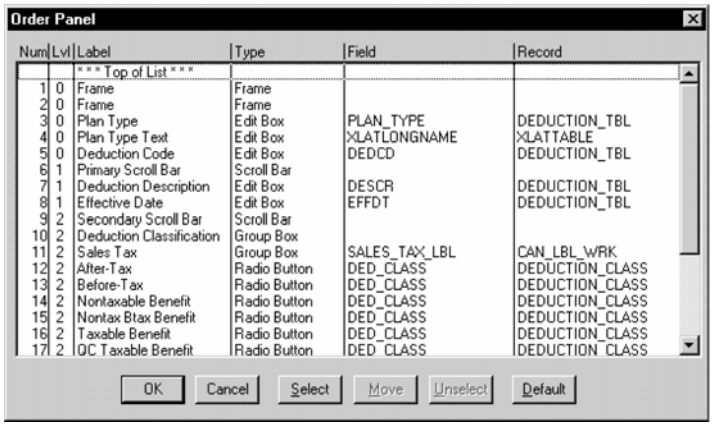


Figure 7.6 Panel layout with parent and child records

Fields from the DEDUCTION_TBL record are placed inside the level one scroll bar, and fields from the DEDUCTION_CLASS record are placed inside the level two scroll bar. The key fields—PLAN_TYPE and DEDCD—are seen only once in this panel and belong to level 0. Level 0 fields are usually populated from the search dialog box. The key fields automatically propagate from level 0 to other scroll levels during save time. Using the level one scroll bar, we can easily enter multiple rows into DEDUCTION_TBL for the same PLAN_TYPE and DEDCD fields, but for different effective dates. Likewise, using the level

two scroll bar, we can enter multiple rows into DEDUCTION_CLASS for a particular PLAN_TYPE, DEDCD, and EFFDT combination.

NOTE The level above the level one scroll bar is called level zero. Any field that has a single occurrence is placed on this level. Usually fields from search/input dialog boxes are placed on level zero. The search key fields are placed once on level zero, and the values in these fields automatically propagate to child records in other scroll levels satisfying the parent/child relationship.

7.1.3 Scroll bars used as work scrolls

Scroll bars can be used as work scrolls in PeopleSoft for two main purposes: to load multiple rows of data from application tables for access and reference; and to update one or more rows of data from the work scroll into database tables and views. Work scrolls are usually hidden scroll bars or placed in a hidden panel within a panel group.

In PeopleSoft HRMS, one example of a work scroll resides in the JOB DATA group of panels. Let's look at the JOB DATA panel group definition to see how the whole panel containing work scrolls is hidden.

The panel group in figure 7.7 in fact has two hidden panels, both containing work scrolls. For our purposes, we will look at the JOB_DATA1_WRK panel which contains multiple work scrolls.

Navigation: File →Open →Panel Group →JOB_DATA

JOB_DATA.GBL (Panel Group)

	Panel Name	Item Name	Hidden	Item Label	Folder Tab Label
1	JOB_DATA1	JOB_DATA1	☐	&Work Location	
2	JOB_DATA_JOBCO	JOB_DATA_JOBCODE	☐	&Job Information	Job Information
3	JOB_DATA2	JOB_DATA2	☐	&Payroll	
4	JOB_DATA3	JOB_DATA_3	☐	&Compensation	Compensation
5	JOB_DATA_ERNDIS	JOB_EARNINGS_DISTRI	☐	Job Earnings &Distri	
6	JOB_DATA_BENPR	BENEFIT_PROGRAM_P	☐	&Benefit Program P	
7	EMPLOYMENT_DTA	EMPLOYMENT_DATA	☐	&Employment Data	
8	JOB_DATA1_WRK	JOB_DATA1_WRK	☑	Job Data1 Wrk	
9	SCRTY_TBL_GBL_	SCRTY_TBL_GBL_WRK	☑	Scrty Tbl Gbl Wrk	

Figure 7.7
Panel groups with hidden panels

TIP You can place work scrolls in hidden panels within a panel group. Alternatively, you can make the work scrolls themselves invisible.

Look at the number of scroll bars this panel contains (figure 7.8). This panel accesses more record definitions than all the other panels in this panel group combined. In PeopleSoft HRMS, when an employee is hired, a number of related tables have to be populated during save time. These work scrolls help perform that task.

Navigation: File →Open →Panel →JOB_DATA1_WRK

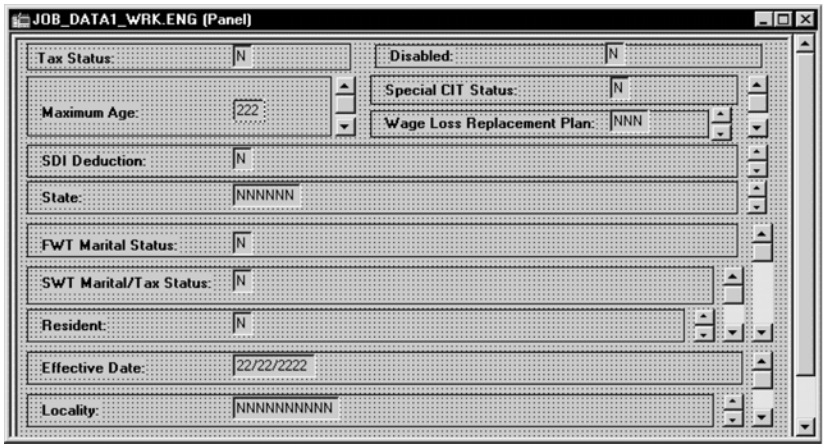


Figure 7.8 Hidden panel with work scrolls

TIP When a field from a record definition is placed on a scroll bar, all the other fields from that record definition are also available in the panel buffer. This is true for record definitions that are SQL tables and view. This means any field from that record definition can be accessed directly using PeopleCode. This does not apply to derived records.

The work scrolls on the JOB_DATA1_WRK panel contain only one field from each record that they update. Because all fields from a record in a work scroll are available in the panel buffer, the Application Processor is able to update all of them. The field labeled "Maximum Age" in figure 7.8 belongs to the record CAN_TAX_TBL, which is placed on scroll level number one. Let's look at the properties for this scroll bar.

Notice in figure 7.9 that this scroll bar is defined as No Auto Select. This means data are populated into the scroll using a PeopleCode event, not the Application Processor. Based on an action, a PeopleCode event populates data into CAN_TAX_TBL scroll bar. When the panel is saved, this data are automatically saved into the database table.

Now let's look at the field from figure 7.8 labeled "State" (in the center of the panel). This field is from the TAX_LOCATION2 record definition and is also in scroll level one. Let us take a look at the properties of the scroll bar which contains this field.

Navigation: Edit →Panel Field Properties (JOB_DATA1_WRK Panel is open)

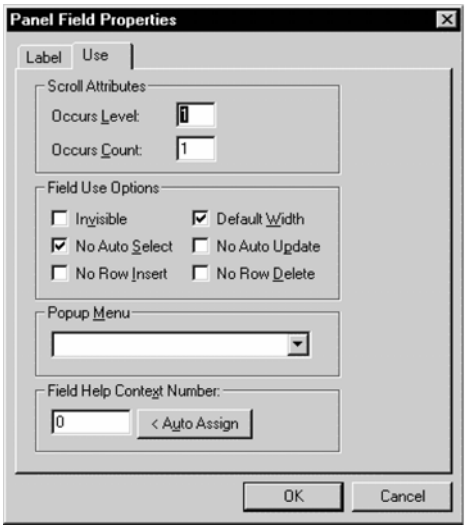


Figure 7.9
Scroll bar properties

TIP Use the No Auto Select feature under Scroll Bar properties to disable the Application Processor from populating data into scrolls automatically. Similarly, use the No Auto Update feature to disable the Application Processor from saving data in scrolls to the database.

In figure 7.10, the scroll bar is set for auto select. As soon as the TAX_LOCATION_CD is filled up with a value on JOB_DATA1 panel, this scroll will be populated in the panel buffer. Also, for every effective-dated row in the JOB record, corresponding values will be built into this scroll bar.

Let us take a look at the definition for the TAX_LOCATION2 record. In figure 7.11 we see that TAX_LOCATION_CD field is a search key on TAX_LOCATION2. As soon as TAX_LOCATION_CD field is available in the panel buffer the Application Processor automatically selects data into the work scroll. Prompt records follow the same concept, using search key values to produce a prompt list.

NOTE When the No Auto Select option is turned off, scroll bars are set for auto select. If a record in such a scroll bar has key fields, the Application Processor populates the fields from the record as soon as values for these key fields are available in the panel buffer. The key fields must be either in the same scroll level or in higher level scrolls.

Navigation: Edit →Panel Field Properties (JOB_DATA1_WRK Panel is open)

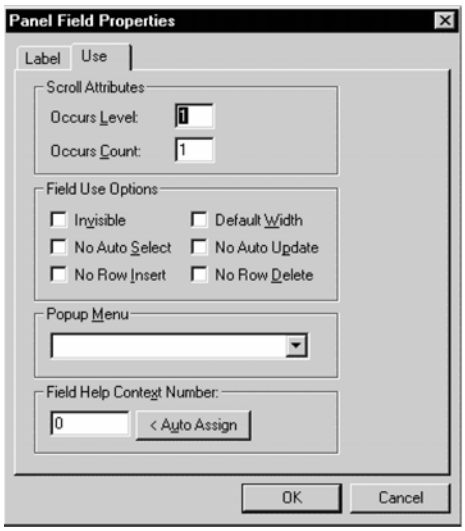
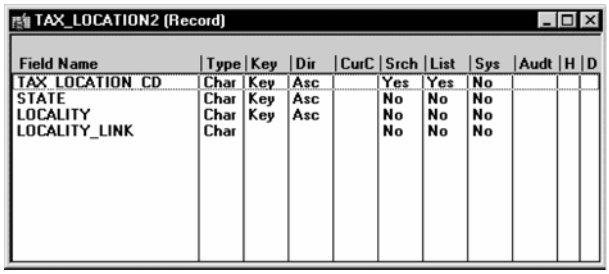


Figure 7.10
Scroll bar properties

Navigation: File →Open →Record →TAX_LOCATION2



Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H	D
TAX_LOCATION_CD	Char	Key	Asc		Yes	Yes	No			
STATE	Char	Key	Asc		No	No	No			
LOCALITY	Char	Key	Asc		No	No	No			
LOCALITY_LINK	Char				No	No	No			

Figure 7.11
Key structure of TAX_LOCATION2

Now let's look at the panel field layout for the JOB_DATA1_WRK panel (figure 7.12).

All the work scrolls in this panel (figure 7.12) are level one scroll bars. Some are set for No Auto Select and are populated by a PeopleCode event. For most work scrolls in the panel, No Auto Update option is turned off. This enables the Application Processor to save data from the scroll buffer to the underlying database table during save time.

PeopleCode functions that update scrolls are `InsertRow`, `UpdateValue`, `ScrollSelect`, `ScrollSelectNew`, and so on. (We discuss these PeopleCode functions in chapter 16.) Let us take a quick look at the PeopleCode program that populates the FED_TAX_DATA scroll bar.

As illustrated in figure 7.12 only the FWT_MAR_STATUS field from FED_TAX_DATA record is placed on the panel. But, take a look at the PeopleCode

Navigation: Layout →Order (JOB_DATA1_WRK panel is open)

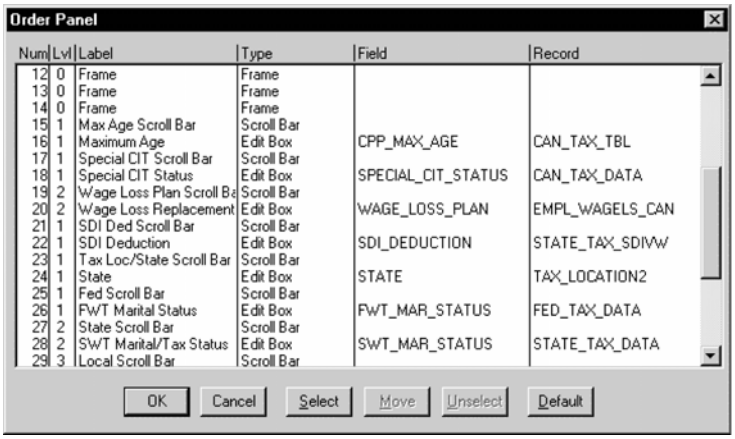


Figure 7.12 Panel field layout of a hidden panel with work scrolls

program shown in figure 7.13. Here you can see other fields from FED_TAX_DATA being referenced and populated. The entire FED_TAX_DATA record definition is available for reference in the panel buffer.

Navigation: File →Open →Record →FUNCLIB_PAY →Highlight Field →View PeopleCode

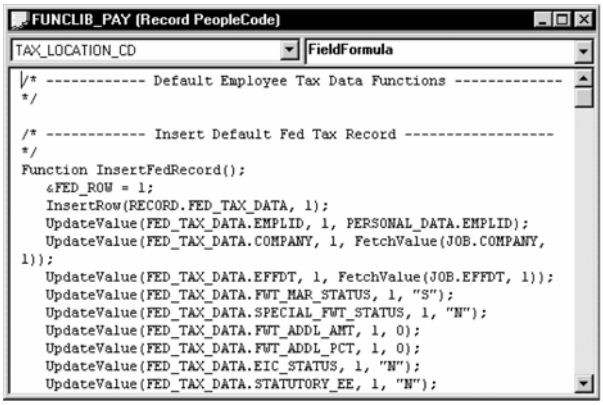


Figure 7.13 PeopleCode event that updates work scrolls

This PeopleCode program is written as a function and included in a function library. This function is also stored in a derived record that contains many payroll functions. The two key PeopleCode functions used to insert and populate rows on the scroll bar are `InsertRow` and `UpdateValue`. Similarly, STATE_TAX_DATA and LOCAL_TAX_DATA scrolls are populated using the same PeopleCode functions.

When the panel is saved, the Application Processor automatically populates the underlying database tables.

7.2 WORKING WITH EFFECTIVE DATES

Effective date and effective sequence are two of the most important fields used in a PeopleSoft application. Effective Date and Effective Sequence fields are used to maintain historical data in PeopleSoft. These two fields create an audit trail of changing application data. They also enable PeopleSoft batch applications to perform retroactive and future-dated processing. Effective-dated data rows are accessed in PeopleSoft using scroll bars. The effective date field is usually the first field on the scroll bar in the panel field layout. Effective dated processing is based on menu item actions. Menu item actions are `Add`, `Data Entry`, `Update/Display`, `Update/Display All`, and `Correction`.

Based on the action chosen by the user, the number of rows fetched from the database varies. Let us see how menu item actions affect the number of rows selected.

Add When an `Add` action is used to access a panel group, the Application Processor checks for the existence of rows in the search record with keys in the input dialog box. If the row already exists in the database table, a message is issued to the user (figure 7.14).

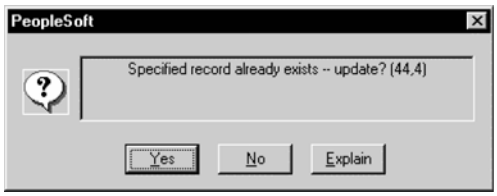


Figure 7.14
Record exists during Add action

In the event that record already exists and if the user chooses "Yes" to the message in figure 7.14, menu item action is automatically changed to `Update/Display`.

Update/Display When the `Update/Display` action is used to access a panel group, only the current and future dated rows are selected from the table. In the `Update/Display` mode, data on the current effective-dated row cannot be changed. With the exception of the effective date field itself, other data on a future dated row may be changed.

Update/Display All When the `Update/Display All` action is used to access a panel group, all the effective-dated rows from the table are selected. As with `Update/Display`, data on the current effective-dated row cannot be changed. With the exception of the effective date field, all other fields may be changed.

Correction When the `Correction` action is used to access a panel group, all effective-dated rows from the table are selected. All the selected rows can be changed including the effective date field. For this reason, `Correction` must be authorized only to users who are administrators and understand the implications of correcting data.

Let us look at a sample set of data to understand historical, current, and future-dated rows in a table. We can use the DEPARTMENT table from the PeopleSoft HRMS application to illustrate the concept (figure 7.15).

Navigation: Go →Define Business Rules →Manage Human Resources U.S. →Use →Department Table →Correction

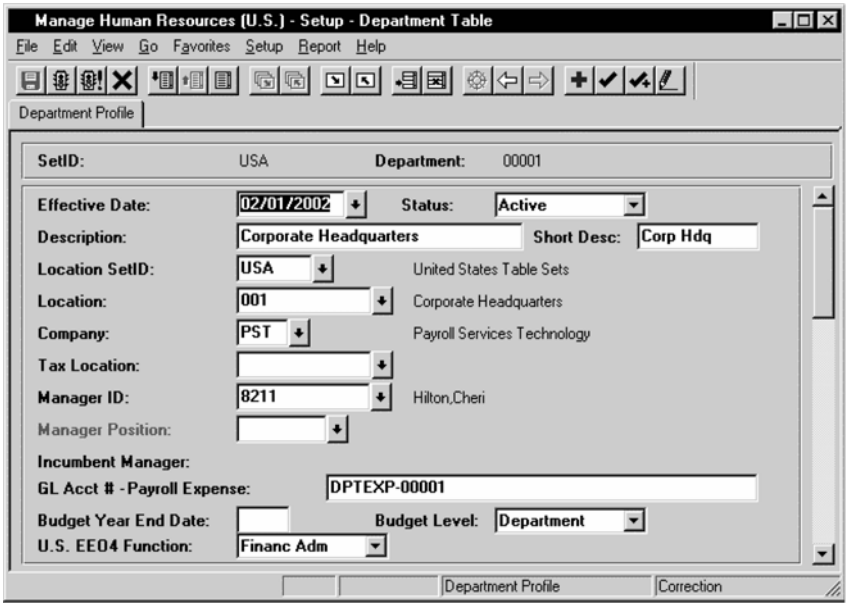


Figure 7.15 Panel with effective dates on scrolls

We brought up the panel in figure 7.15 using the `Correction` action. This loads all the effective-dated rows on the scroll bar. Notice that EFFDT is the first field inside the scroll bar. Let's take a closer look at the effective dates from all rows from the DEPARTMENT table loaded in the scroll bar. The rows are sorted in a descending sequence on the scroll bar (table 7.1).

NOTE Effective dates are used to maintain historical data. Scroll bars are used in panels to view historical effective-dated rows.

Table 7.1 Effective dates from the DEPARTMENT table

Effective Date	Definition	Description
02/01/2002	Future	This date is in the future compared to the current date.
06/15/1996	Current	This date is current and effective as of today.
01/01/1960	History	This date is in history.

Let us see what happens when current and history rows are changed using the `Update/Display` and `Update/Display All` actions.

A message (figure 7.16) is issued.

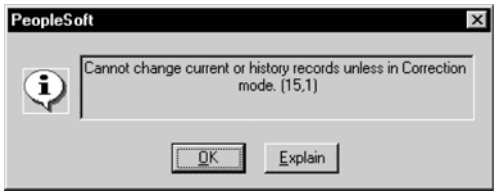


Figure 7.16
Changing data using Update/Display

TIP We can prevent users from correcting current and history rows by disabling their access to the `Correction` action in menu items. This can be accomplished with the help of the Security Administrator tool.

7.2.1 PeopleCode functions for effective-dated processing

Some delivered PeopleCode functions are built into PeopleTools. Most of these functions are used to fetch the effective dates or the row numbers which contain the effective dates from the scroll bar. All these functions work only on effective-dated records. Let us look at some of these functions and how they can be used.

`CurrEffDt` returns the effective date from the current row on the scroll bar. Regardless of where the cursor is on the scroll bar, this function will return the value of the current effective date.

`CurrEffSeq` returns the value of the effective sequence field from the current row on the scroll bar.

`CurrEffRowNum` returns the row number that contains the current effective-dated row on the scroll bar.

`NextEffDt` returns the value of a field on the next effective-dated row. This function takes a fieldname as a parameter.

`NextRelEffDT` returns the value of a related display field from the next effective-dated row. The input parameter to this function is usually a panel field defined as a Display Control item. The output field is defined as a "Related Display" field.

`PriorRelEffDt` returns the value of a "Related Display" field from the prior effective-dated row. The parameters are the same as the `NextRelEffDt` function.

(Please refer to part 3 of this book for more detailed descriptions about these built-in PeopleCode functions.)

NOTE The `Update/Display` action selects only current and future effective dated rows. The `Update/Display All` and `Correction` actions select all rows from the database table.

NOTE The Effective Sequence field is used in conjunction with the Effective Date field. We can use Effective Sequence to distinguish history rows with the same effective date. For example, in PeopleSoft HRMS, JOB record has both EFFDT and EFFSEQ as key fields. When the user enters a promotion and pay rate increase using the same EFFDT for an employee, the EFFSEQ field is used to distinguish the two rows.

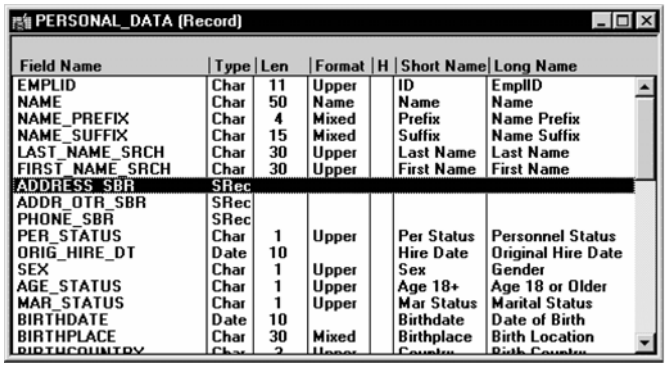
7.3 WORKING WITH SUBPANELS AND SECONDARY PANELS

Subpanels are used to populate repetitive sets of fields using a subrecord. Alternatively, secondary panels are used to organize panel fields based on functionality. Both subpanels and secondary panels are used to organize panel fields and make them easier for input.

7.3.1 Subpanels

Subrecords play a key part in building subpanels. Let's look at a record definition which contains a subrecord, the PERSONAL_DATA record from the PeopleSoft HRMS system (figure 7.17).

Navigation: File →Open →Record →PERSONAL_DATA



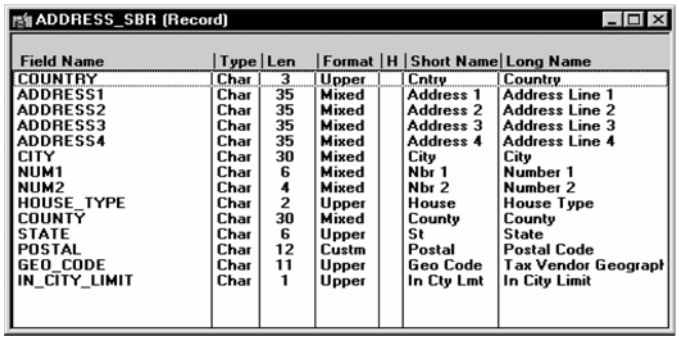
PERSONAL_DATA (Record)

Field Name	Type	Len	Format	H	Short Name	Long Name
EMPLID	Char	11	Upper		ID	EmplID
NAME	Char	50	Name		Name	Name
NAME_PREFIX	Char	4	Mixed		Prefix	Name Prefix
NAME_SUFFIX	Char	15	Mixed		Suffix	Name Suffix
LAST_NAME_SRCH	Char	30	Upper		Last Name	Last Name
FIRST_NAME_SRCH	Char	30	Upper		First Name	First Name
ADDRESS_SBR	SRec					
ADDR_OTR_SBR	SRec					
PHONE_SBR	SRec					
PER_STATUS	Char	1	Upper		Per Status	Personnel Status
ORIG_HIRE_DT	Date	10			Hire Date	Original Hire Date
SEX	Char	1	Upper		Sex	Gender
AGE_STATUS	Char	1	Upper		Age 18+	Age 18 or Older
MAR_STATUS	Char	1	Upper		Mar Status	Marital Status
BIRTHDATE	Date	10			Birthdate	Date of Birth
BIRTHPLACE	Char	30	Mixed		Birthplace	Birth Location
BIRTHCOUNTRY	Char	3	Upper		Country	Birth Country

Figure 7.17
Record definition with subrecords

ADDRESS_SBR is a great example of the use of subrecords and subpanels. Consider now the definition of ADDRESS_SBR subrecord (figure 7.18).

Navigation: File →Open →Record →ADDRESS_SBR



ADDRESS_SBR (Record)

Field Name	Type	Len	Format	H	Short Name	Long Name
COUNTRY	Char	3	Upper		Cntry	Country
ADDRESS1	Char	35	Mixed		Address 1	Address Line 1
ADDRESS2	Char	35	Mixed		Address 2	Address Line 2
ADDRESS3	Char	35	Mixed		Address 3	Address Line 3
ADDRESS4	Char	35	Mixed		Address 4	Address Line 4
CITY	Char	30	Mixed		City	City
NUM1	Char	6	Mixed		Nbr 1	Number 1
NUM2	Char	4	Mixed		Nbr 2	Number 2
HOUSE_TYPE	Char	2	Upper		House	House Type
COUNTY	Char	30	Mixed		County	County
STATE	Char	6	Upper		St	State
POSTAL	Char	12	Custm		Postal	Postal Code
GEO_CODE	Char	11	Upper		Geo Code	Tax Vendor Geograph
IN_CITY_LIMIT	Char	1	Upper		In Cty Lmt	In City Limit

Figure 7.18
Definition of a subrecord

All the fields in ADDRESS_SBR are standard fields that can be used to update address information. Many record definitions in PeopleSoft contain ADDRESS_SBR in its definition. In the database, the subrecord does not actually exist as an SQL table. Only the online application recognizes the subrecord. In the database, the fields from the SubRecord are automatically expanded and stored as individual fields. Because ADDRESS_SBR is used in multiple record definitions, the address fields in all these tables can be updated using the ADDRESS_SBP panel (figure 7.19).

The ADDRESS_SBP subpanel can be placed into any other panel (containing record definitions), using the ADDRESS_SBR subrecord. Notice that the panel field labels are also specified as panel fields. Even though the fields can be the same, the label for these fields can be different based on the context of the application panel. Take a

Navigation: File →Open →Panel →ADDRESS_SBP

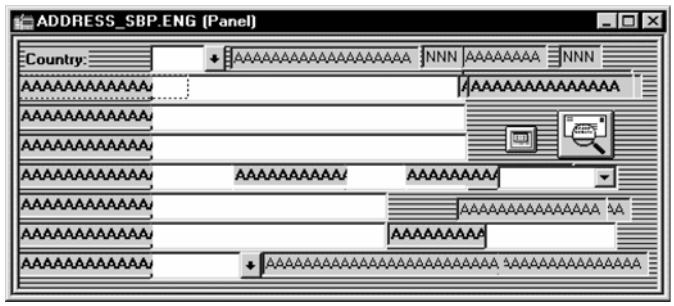


Figure 7.19
ADDRESS_SBP subpanel

look at the PERSONAL_DATA1 panel which has the ADDRESS_SBP subpanel in it (figure 7.20).

Navigation: File →Open →Panel →PERSONAL_DATA1

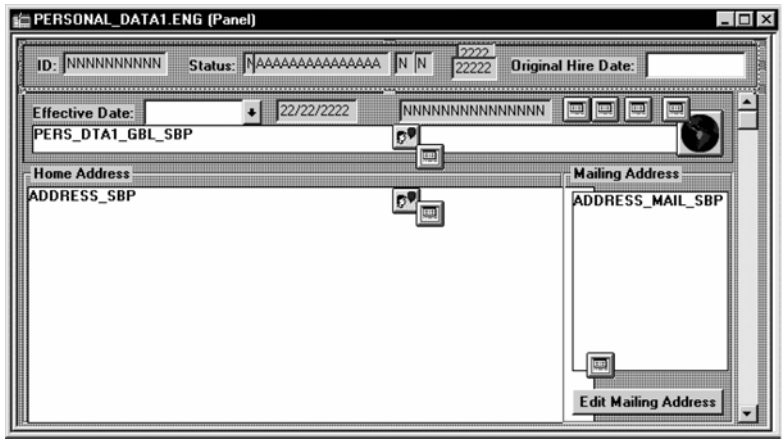


Figure 7.20 Application panel using a subpanel

TIP Record fields can be used as panel field labels. This technique is used to provide context sensitive field labels for Subrecord fields. Subrecord fields are pre-designed in a subpanel and field labels cannot be changed when these subpanels are included in a main panel. Record fields can be used as field labels so that they can be populated based on the context of the main panel (which includes the subpanel in question).

The difficult part in designing a subpanel is the placement of fields on the subpanel. Fields have to suit all the application panels that use them. A potential problem also arises when subpanels must fit into application panels. Usually, fields on the main application panels are rearranged to fit the subpanel properly. The ADDRESS_SBP subpanel in the PERSONAL_DATA1 panel contains all those fields from the ADDRESS_SBR subrecord. At save time, these fields are updated into the PS_PERSONAL_DATA table.

Subpanels can be inserted into an application panel by choosing `Insert/SubPanel` from the Application Designer menu.

The same subpanel, when placed on LOCATION_TABLE1 panel, accesses the address fields from the LOCATION_TBL record. Another popular subpanel in all PeopleSoft applications is the Process Run Control subpanel.

Navigation: File →Open →Panel →PRCSRUNCNTL_SBP

Figure 7.21 Process Run Control subpanel

All Run Control panels, which initiate a batch process, require the operator ID and Run Control ID fields in their record definition. The Process Run Control subpanel helps the Run Control panels by providing these two fields.

7.3.2 Secondary panels

Secondary panels are used to segregate functionally such as fields into a separate panel. Optional fields are separated out into a secondary panel. Secondary panels are accessed using a push button. Both the push button and the secondary panel are placed on the main application panel, one after another. Push button fields are always placed before the secondary panels in the panel layout.

Secondary panels can be inserted into a main application panel by choosing Insert/Secondary Panel from the Application Designer menu. In figure 7.20, the PERSONAL_DATA1 panel also includes secondary panels.

The push button labeled Edit Mailing Address is a derived field which activates the secondary panel. The secondary panel appears as a small hidden icon in the main application panel. Let's look at the definition of this secondary panel (figure 7.22).

Navigation: File →Open →Panel →ADDRESS_OTHER_SEC

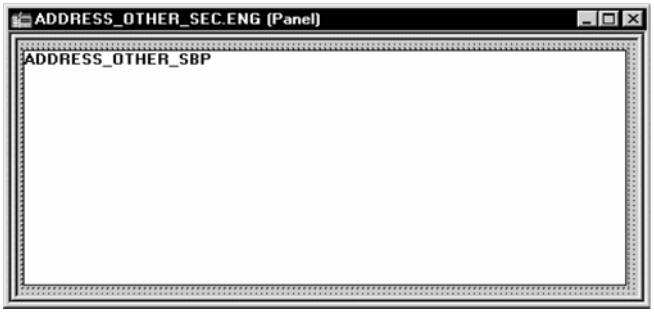


Figure 7.22
Secondary panel definition

TIP Secondary panels have an advantage over subpanels. Secondary panels do not have to be designed to accommodate the main application panel. They are brought up online by pushing a button from the main panel. When secondary panels are brought up, they appear on top of the main panel. They can be closed once data are entered into the fields after focus is transferred back to the main panel.

This secondary panel also uses a subpanel. Subpanels are pre-developed panels which contain certain fields. As soon as the subpanel is placed in an application or a secondary panel, all fields from the subpanel are automatically included inside them. This secondary panel is used to edit the mailing address for an employee. Since address fields are already built into ADDRESS_SBP and ADDRESS_OTHER_SBP subpanels available in the system, this secondary panel makes use of the ADDRESS_OTHER_SBP subpanel.

TIP If a SubPanel design is not suitable to fit the main application panel, the SubPanel can be included in a Secondary Panel and then the Secondary Panel can in turn be included in the main panel.

Let's look at the PeopleCode event which brings up this secondary panel (figure 7.23).

The `DoModal` PeopleCode function is used to bring up the secondary panel online. Instead of cluttering the panels with fields used to enter optional mailing addresses, PeopleSoft has built this secondary panel, which contains these fields. The secondary panel can be summoned on an as-needed basis by activating a push button.

Navigation: File →Open →Record →DERIVED_HR →View →PeopleCode Display

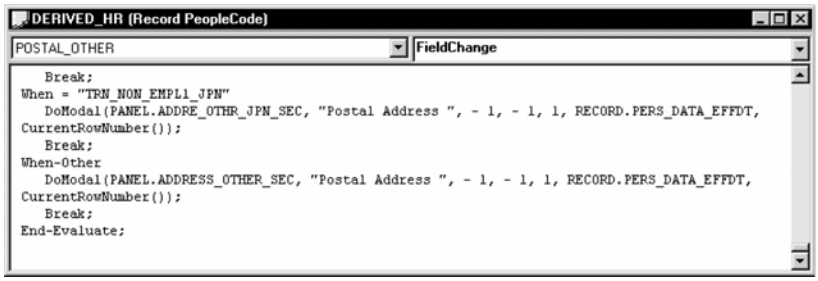


Figure 7.23 PeopleCode program that activates a secondary panel

In figure 7.24, the secondary panel appears on top of the main application panel:

Navigation: Activate Push Button

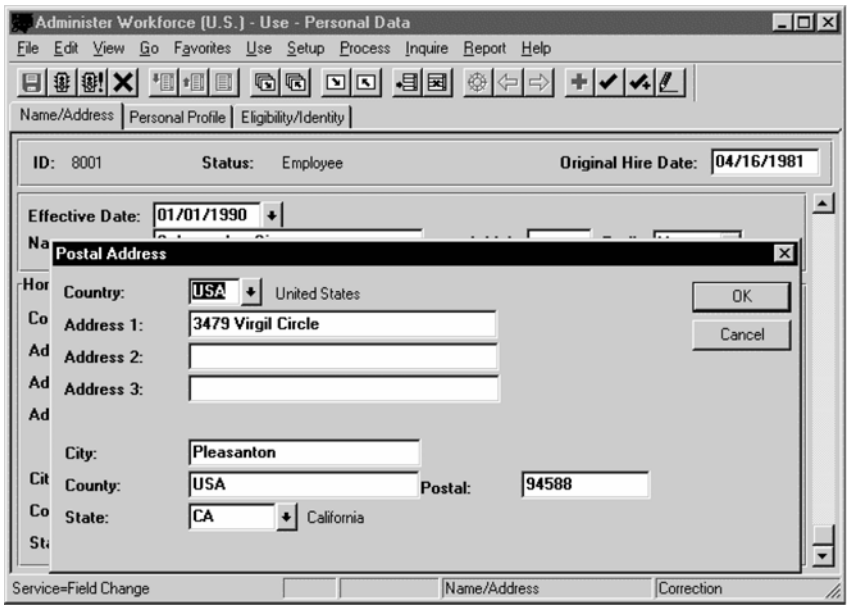


Figure 7.24 Secondary panels on an application panel

7.4 DESIGNING INQUIRY PANELS

Inquiry panels are display panels that cannot update data. Used as organized queries to the database, inquiry panels are easy to build. They consist of several record fields assembled on a panel adhering to panel design rules. Inquiry panels are also used to

by-pass some complex PeopleCode events in record definitions and help in quick inquiry of data. The Application Processor populates information from database tables into the online panel.

In our Problem Tracking application, we designed an inquiry panel which shows us totals by problem status. Let's look at the definition for MY_TRKG_STATUS panel (figure 7.25).

Navigation: File →Open →Panel →MY_TRKG_STATUS

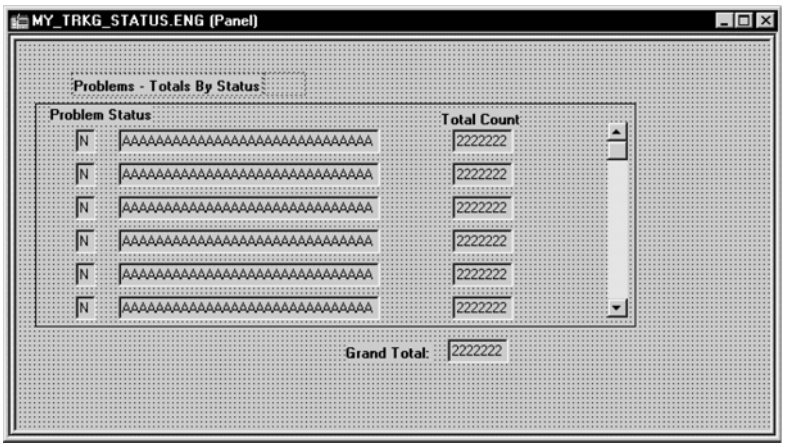


Figure 7.25 Inquiry panel

This simple example of an inquiry panel makes use of an SQL view to display information online. The description field is a related display field from the XLATTABLE. The TOTAL_COUNT field inside the scroll bar is the result of the aggregate `SUM` function used in the SQL view. The grand total field is populated by a simple PeopleCode event adding all the individual totals from inside the scroll bar and displaying the grand total using a derived field.

By building an SQL view definition and using fields from the view definition on the inquiry panel, all PeopleCode events from MY_PROBLEM_TRKG record are bypassed (figure 7.26):

Navigation: File →Open →Record →MY_TRKG_STATUS →File →Object Properties

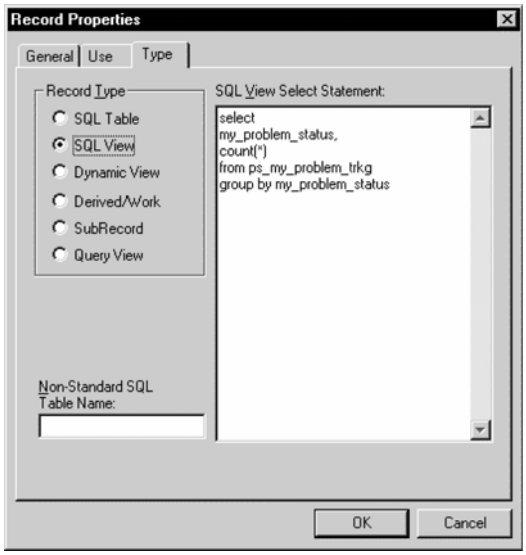


Figure 7.26
SQL view definition

Take a look, too, at the record definition of MY_TRKG_STATUS SQL view (figure 7.27).

Navigation: File →Open →Record →MY_TRKG_STATUS

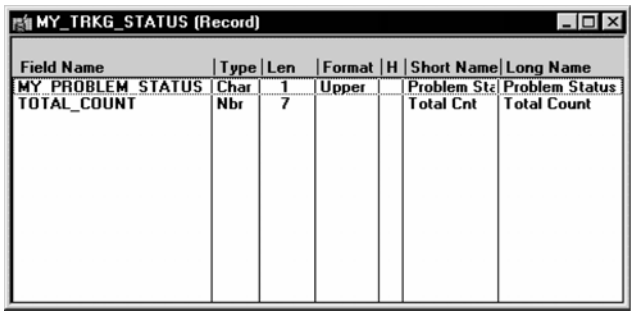


Figure 7.27
Record definition of an SQL view

The inquiry panel has the two fields from the SQL view definition and a related display field from the XLATTABLE. Let's look at the panel field layout and see how the grand total field is derived (figure 7.28).

Notice that the grand total field is a derived field from the MY_DERIVED record definition. As the rows are populated inside the scroll, the totals by individual statuses are added and can be displayed in the grand total field. Now, consider the PeopleCode event from the MY_TRKG_STATUS record definition which sums up the individual totals and

Navigation: Layout →Order (MY_TRKG_STATUS panel is open)

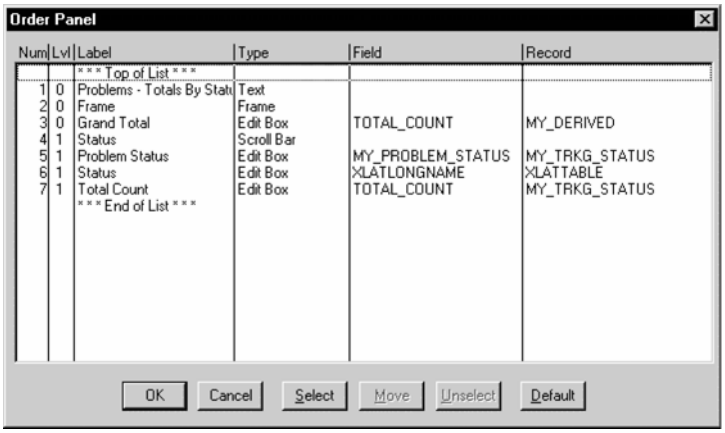


Figure 7.28 Panel layout of an inquiry panel

populates that sum into the TOTAL_COUNT field from the MY_DERIVED record definition (figure 7.29).

Navigation: View →PeopleCode (MY_TRKG_STATUS PeopleCode Display)

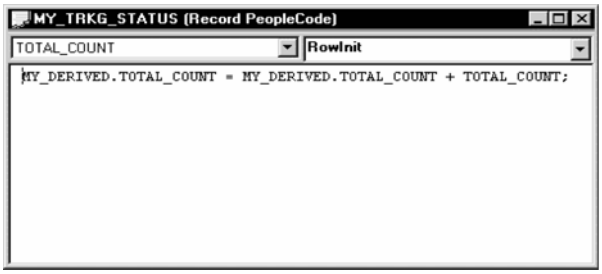


Figure 7.29 PeopleCode event that populates a derived field

In figure 7.30, we see a more complex inquiry panel from the PeopleSoft HRMS system. This panel uses derived fields as switches to change the fields being displayed on the panel. In other words, the panel has key fields from the Job record assembled very close to each other. Based on the choice that the user makes, some fields are hidden and others are shown on the panel. Figure 7.30 shows the definition of this panel.

It may seem that this panel is cluttered with fields, but the four display switches on the top control the display and hiding of fields, using PeopleCode events built behind these display switches. Take a look at the actual online panel (figure 7.31). The online panel appears far more organized than the actual panel design, because certain fields have been hidden based on the choice of switches selected by the user.

Navigation: File →Open →Panel →JOB_SUMMARY

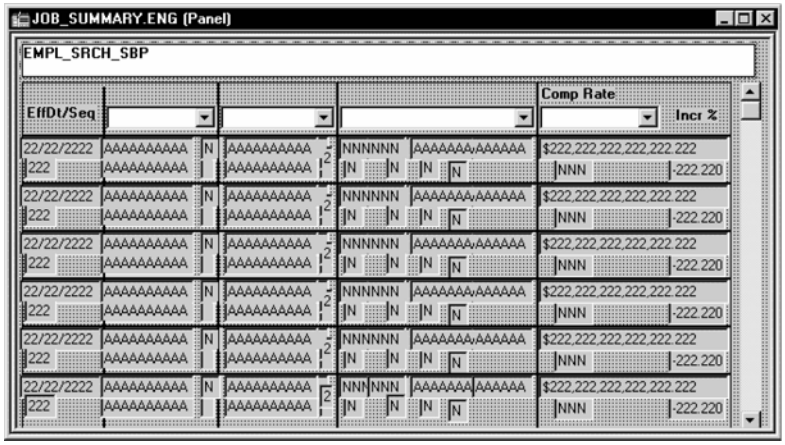


Figure 7.30 Definition of a complex inquiry panel

Navigation: Go →Administer Workforce U.S. →Inquire →Job Summary

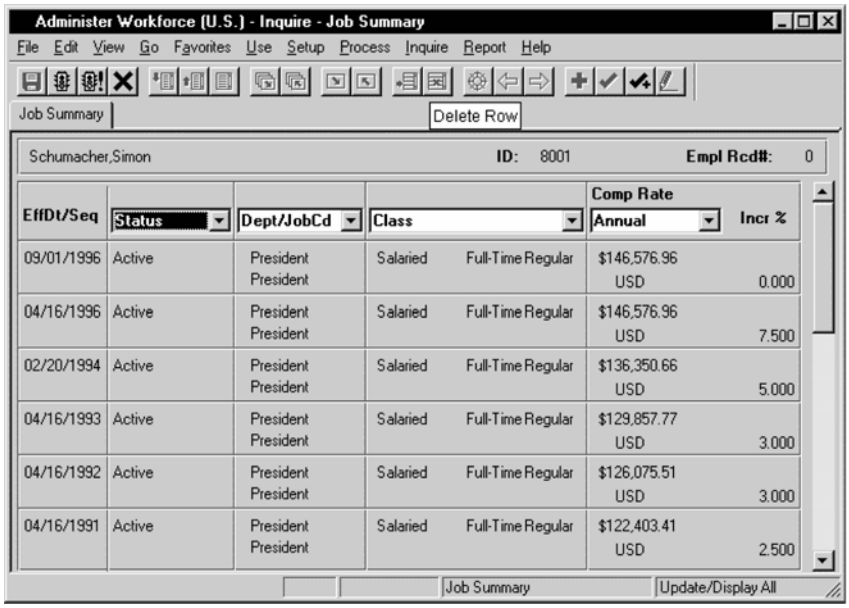


Figure 7.31 Online view of an inquiry panel

The online panel also uses an SQL view definition to display all the fields from the database. The panel uses JOB_VW, an SQL view based on the JOB table. JOB_VW

does not have any of the PeopleCode events from the Job record. The inquiry panel is displayed without processing any PeopleCode event from the Job record. This inquiry panel fetches rows from the database faster than the JOB DATA panels, saving inquiry time.

The same rules apply for scroll bars on an inquiry panel. A scroll bar can contain only fields from one record definition (except for related display and derived fields).

7.5 USING A GRID ON A PANEL

Grids are spreadsheet-like displays on an application panel. As do spreadsheets, grids have cells that can be expanded and contracted. Columns can be frozen to fit more columns into a panel; data from grids can be copied into spreadsheets, and vice versa. Grids are also similar to single level scroll bars. Using grids, we can insert, delete, and change rows.

Grids cannot be used as alternatives to multiple level scroll bars. Basic panel field objects are accommodated on a grid. Objects such as edit boxes, push buttons, check boxes, drop-down list boxes, and long edit boxes are allowed on grids. Note, too, that grids have to be the last control on the panel, and only one grid is allowed in a panel.

In figure 7.32, we see a PeopleSoft HRMS panel that uses a grid for data entry.

Navigation: Go →Compensate Employees →Administer Automated Benefits

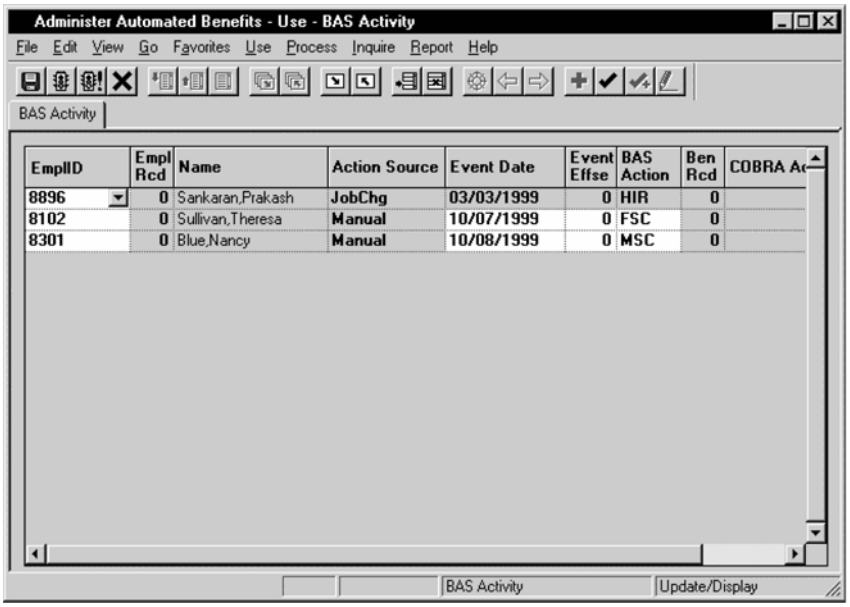


Figure 7.32 Panel with a grid

We can see that some fields here are display fields, others are input fields. The two Manual lines were manually inserted into the grid in a fashion similar to that of `row-insert` function in a scroll bar.

When grids are used, we can:

- sort the grid on any column by clicking on the column heading like a spreadsheet
- copy columns from the grid and paste them into a spreadsheet
- copy columns from a spreadsheet and paste them into the grid
- adjust row height and column width on the grid
- freeze columns on the grid like a spreadsheet

Using the panel from figure 7.32, let's demonstrate all the foregoing features of a grid:

7.5.1 Sorting the grid on its columns

`Employee ID` is the primary key on the main record in this panel. When we bring up the panel, the grid is sorted on the EMPLID field (figure 7.32). After clicking on the column heading for the NAME field (figure 7.33), the grid is sorted on an ascending order of that field. When the column heading is clicked again, the grid will be sorted on a descending order of that field.

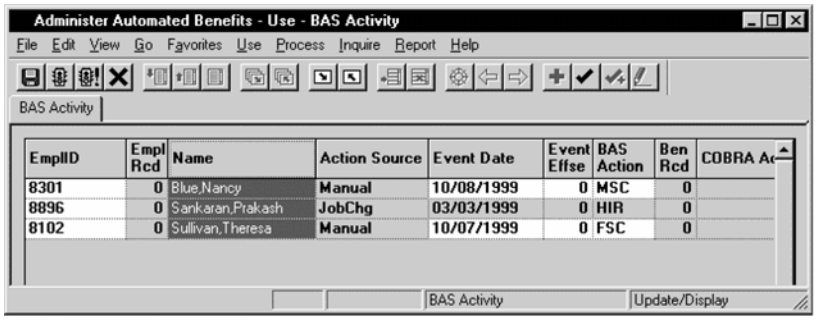


Figure 7.33 Grid sorted on a column

7.5.2 Copy data from grids into spreadsheets

Data from grids can be copied into spreadsheets. When we highlight the cells on the grid and press CTRL-C, the cells can be copied into the Window's clipboard. Then we can paste them into a text editor or into a spreadsheet. This is a useful feature which allows a user to input data into the grid and copy the entire grid into a spreadsheet for a variety of reasons. The user can perform calculations on numeric fields, copy data into spreadsheets for documentation purposes, and so forth. Let's explore this by copying data from the grid into the spreadsheets.

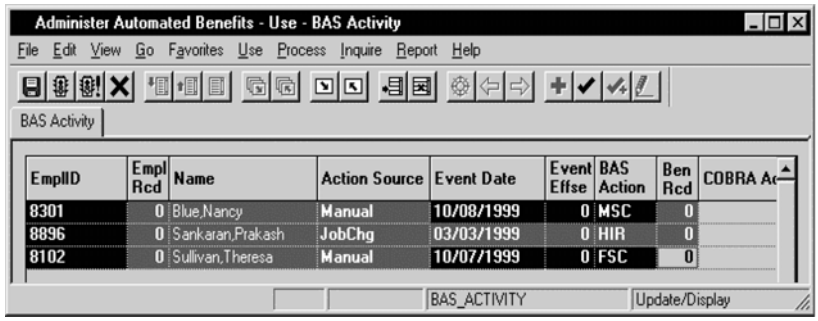


Figure 7.34 Copying data from a grid to the clipboard

Using the mouse—or the shift keys—and highlighting the cells, we copy the cells into the clipboard (figure 7.34). Figure 7.35 illustrates how this data appears when pasted on a spreadsheet.

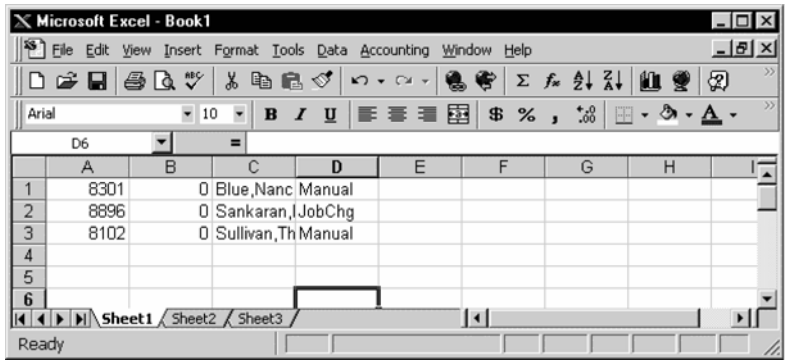


Figure 7.35 Data from grid copied into a spreadsheet

7.5.3 Copy data from spreadsheets into grids

To copy the data from the spreadsheet back to the same grid, we highlight all the cells from the Excel spreadsheet and copy them into the clipboard. Then, we paste right into the grid. Although data can be pasted into one row on the grid, limitations exist on pasting data copied from spreadsheets into grids that have keys. Database key constraints will restrict us from copying multiple rows of data into Grids.

It is possible to copy multiple rows into grids which use derived records. We can create a work grid with derived fields, insert the necessary number of rows into the grid, and paste multiple rows of data into it. Data can be saved in database tables using the `SQLEXEC` PeopleCode function.

7.5.4 Adjust row heights and column widths

By grabbing the edges of a column or a row and dragging them to the appropriate width or height, we can control column and row sizes just as we can in a spreadsheet. In figure 7.36, we can see how the grid appears after adjusting the column width and the row height.

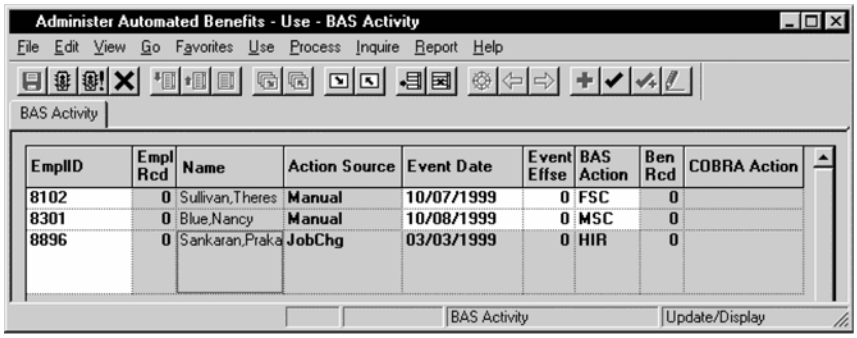


Figure 7.36 Adjust column width and row height on a grid

We increased the row height of the last row on the grid. We also reduced the column width of the NAME column. As you reduce the column width, more columns from the right side appear on the screen.

7.5.5 Freezing columns on a grid

Freezing columns on a grid is not dynamic as on a spreadsheet. The grid must be defined to contain frozen columns, and, unlike in spreadsheets, columns cannot be frozen on the application panel. How do frozen columns appear on the application panel? Figure 7.37 shows our previous grid frozen on the NAME column. When the panel is scrolled to the right, the columns to the right of the frozen column disappear, and the columns hidden on the right side of the panel become visible.

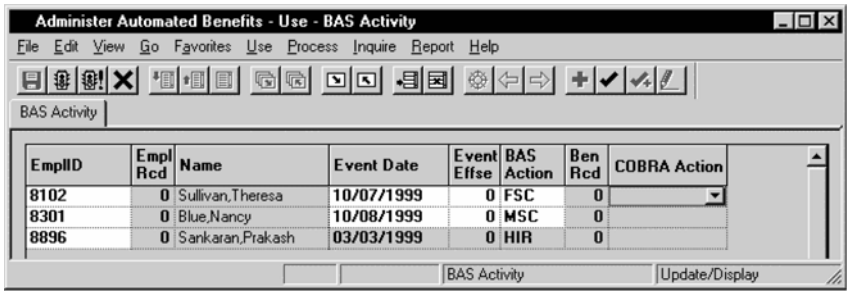


Figure 7.37 Frozen columns on a grid

The Action Source field is hidden and the Cobra Action field appears on the application panel. Previously, this field was not visible, and the whole grid would have scrolled to the left side hiding the EMPLID column. Because the grid was frozen at the NAME column, however, everything to the right of that field scrolls, moving either to the right or the left.

7.5.6 Creating a grid on a panel

Let's take a look at the steps to insert a grid into an application panel. We will present the MY_USER_TBL in a grid-like display, starting by creating a new panel and inserting a grid object into the panel (figure 7.38).

Navigation: File →New →Panel →Insert →Grid

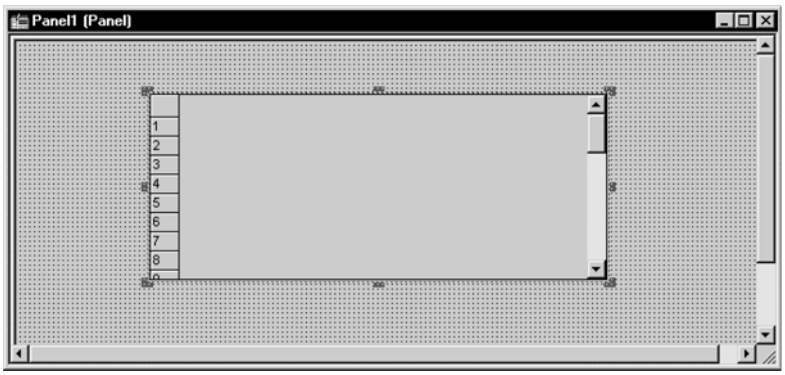


Figure 7.38 Inserting a grid into a new panel

The grid is like any other panel field. We edit the properties for the grid by choosing Panel Field properties like any other panel field. Figure 7.39 illustrates the Grid Properties screen.

We have to define a main record for the grid. Here we define MY_USER_TABLE as the main record. We also define the Occurs Level just as we did on a scroll bar. We insert grids on any level on the scroll bar. In other words, a grid can be placed inside a regular scroll bar.

Under the Columns tab, we add the fields that appear on the grid. We click on the Add button to add columns on the grid. We choose the panel field type and the record field that we want to add to the grid. The properties of each field on the grid are edited using the Panel Field Properties screen.

Under the Labels tab, we add a label to the whole grid. This tab is also used to control the display of row and column headings.

Under the Use tab, we control the properties of the scroll on the grid. We can also attach a pop-up menu to the grid using this tab. Figures 7.40, 7.41, and 7.42 illustrate the other tabs in the Grid Properties screen.

Navigation: Edit →Panel Field Properties (Grid should be highlighted)

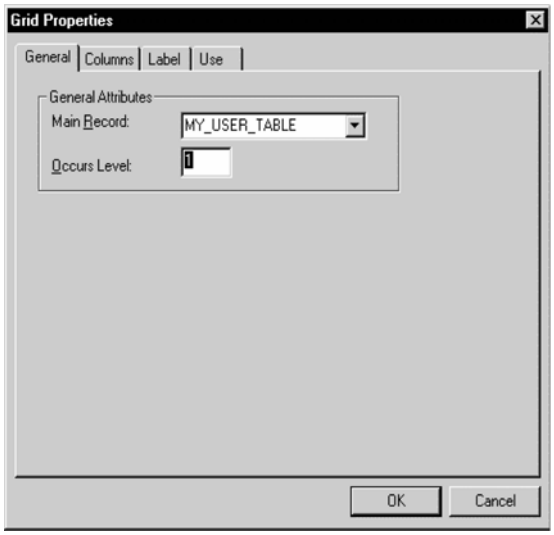


Figure 7.39
Grid Properties—General tab

By clicking on the Add button (figure 7.40), we add any field from MY_USER_TABLE into the grid. As mentioned before, we can use only edit boxes, drop-down list, checkboxes, push button, long edit boxes, and secondary panel field types.

Navigation: Edit →Panel Field Properties (Grid should be highlighted)

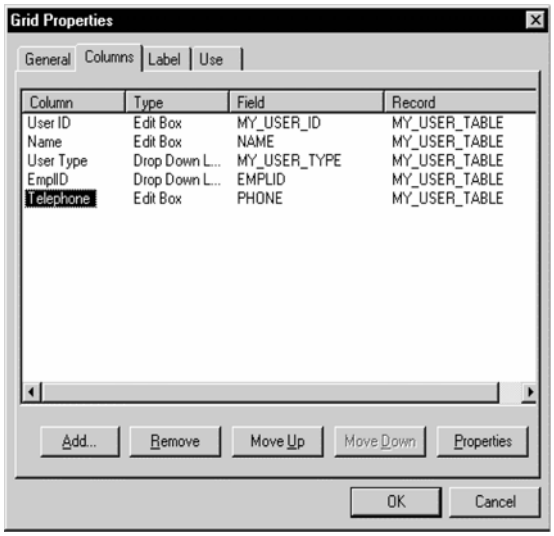


Figure 7.40
Grid Properties—Columns tab

Navigation: Edit →Panel Field Properties (Grid should be highlighted)

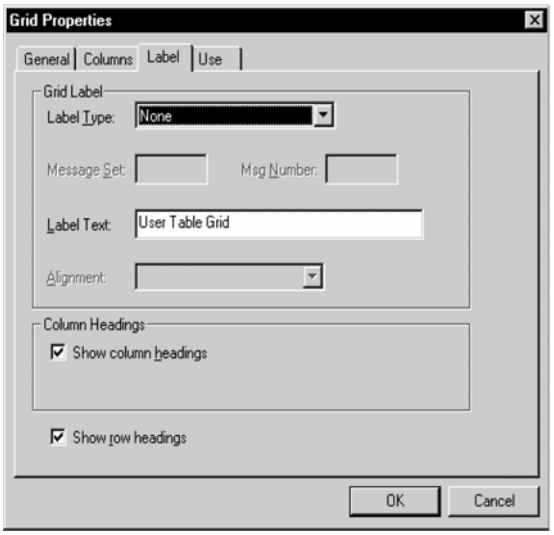


Figure 7.41
Grid Properties—Label tab

Navigation: Edit →Panel Field Properties (Grid should be highlighted)

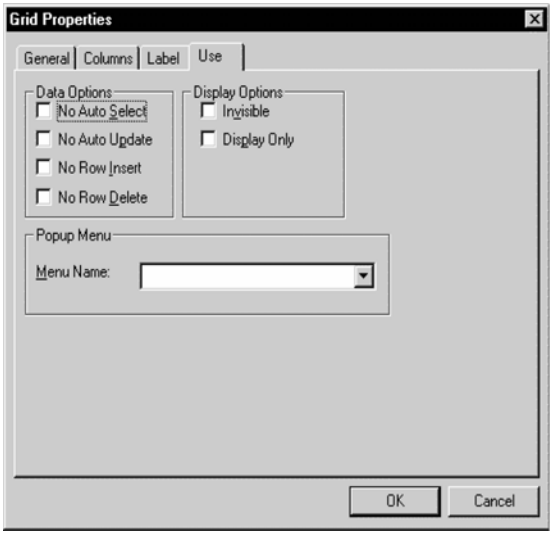


Figure 7.42
Grid Properties—Use tab

By highlighting MY_USER_ID field from the Columns tab in figure 7.40 and clicking on the Properties button, we edit field properties for that individual field within the grid. Here we can specify the panel field properties (as covered in chapter 4).

One key feature, in the Panel Field properties screen is the Freeze Grid Column option (figure 7.43). Choosing this option, we freeze columns on the grid just as we

Navigation: Properties (From Columns tab in Grid Properties screen)

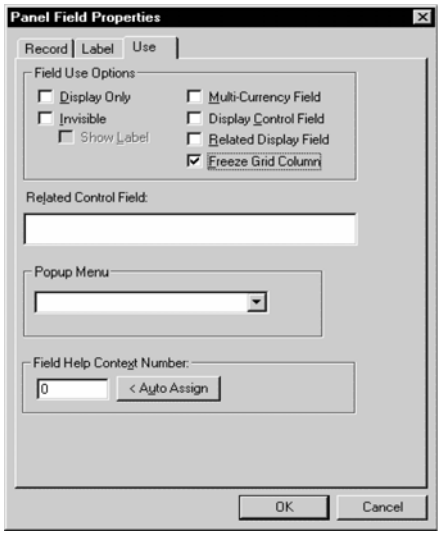


Figure 7.43 Freeze grid column

can on a spreadsheet. All columns to the left of the column chosen for the freeze are frozen as well. We can also remove columns from the grid by choosing the Remove button under the Columns tab (figure 7.40). We order the sequence of the fields on the grid by choosing the Move Up or Move Down buttons from the Columns tab of the Grid Properties screen.

After attaching the grid panel to a panel group, adding a menu item to our Problem Tracking menu, and providing security to the new menu item, we can view the grid panel online. Figure 7.44 shows how the grid appears in the application.

The columns in a grid panel can be resized online. Columns defined as frozen fields do not scroll when the scroll bar is used to move to the right or the left side.

Navigation: Go →Problem Tracking →Setup →User Table Grid

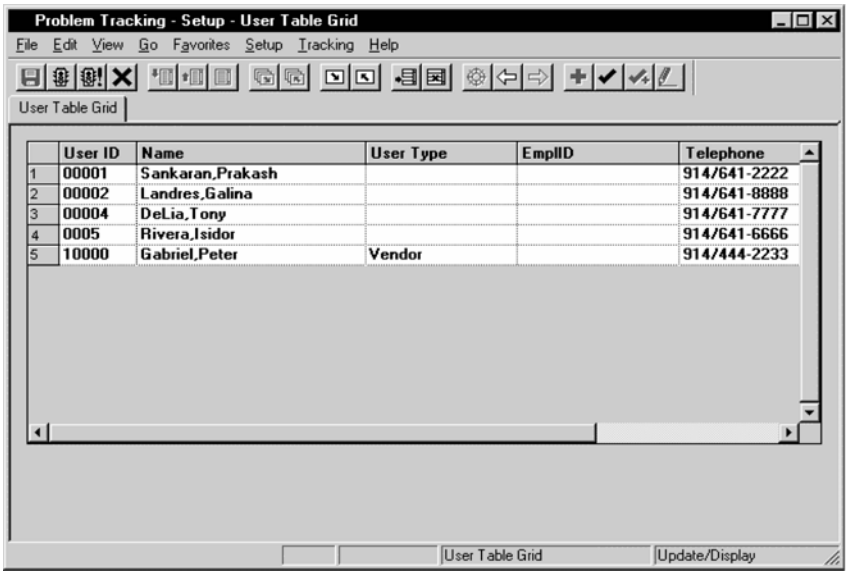


Figure 7.44 Grid application panel

NOTE Only one Grid can be used in an application panel. Grid has to be the last control or field in the panel. Grids are alternatives to a single level scroll bar. Columns and rows in a grid can be resized just as in a spreadsheet. Columns can also be frozen using the Columns tab under the Grid Properties screen.

KEY POINTS

1. Scroll bars are used to display multiple rows of data from the same record on one application panel.
2. Scroll bars are also used to maintain historical data using the EFFDT and EFFSEQ fields.
3. The level above the level one scroll bar is called level 0. Level 0 fields are usually fields from the search dialog box. These fields automatically propagate to child scroll levels to satisfy Parent/Child relationships.
4. The `Update/Display` action displays only current and future effective-dated rows from the database table.
5. The `Update/Display All` and `Correction` actions display all effective-dated rows from the database table.
6. Subpanels and secondary panels are used to organize panel fields. Subpanels can be built using subrecords which contain repetitive fields that can be shared across many panels. Secondary panels separate panel fields by function. Optional fields which are seldom entered can be included into a secondary panel.
7. Inquiry panels can be designed to facilitate faster access to data. PeopleCode programs in records defined as SQL tables can be bypassed by building clone records that are defined as SQL views. The fields from the SQL view can be used in the inquiry panels.
8. A grid can be used to replace a single-level scroll bar on a panel.
9. A grid can be used in a panel to facilitate easier data entry. It also provides the user with capabilities such as resizing row and column size, freezing columns, and copying data into spreadsheets.

CHAPTER 8

Building database objects

8.1 TABLES AND VIEWS IN PEOPLESOFT

PeopleSoft applications are table-based systems. PeopleTools, which runs the online application, is stored in database tables and views. A PeopleSoft application, which runs using PeopleTools, contains three distinct sets of SQL tables and views—Database catalog tables and views, PeopleTools tables and views, as well as Application tables and views.

The tables in each set perform unique functions to run the application. They are also distinguished by the manner in which they are created, modified, deleted, and likewise populated.

8.1.1 Database catalog tables and views

Database catalog tables and views are system tables which store attributes for all PeopleTools and Application tables and views in the database. Database catalog tables vary across database platforms; database catalog views are representations of data stored in the database catalog tables.

The database engine uses data attributes to access and store application data. Database catalog tables store field attributes, table attributes, index attributes, table spaces, view definitions, and so on. Installed when the database is created, database catalog tables and views are updated when the PeopleTools and application tables are created, modified, or deleted. When a PeopleSoft application is installed, both PeopleTools tables/views and Application tables/views are created in the database. Let's look at a few examples of database catalog tables (table 8.1).

Table 8.1 Database catalog table

Database Platform	Database Object	Catalog Table
Oracle	Tables	DBA_TABLES
	Columns	DBA_TAB_COLUMNS
SQLBase	Tables	SYSTABLES
	Columns	SYSCOLUMNS

8.1.2 PeopleTools tables and views

PeopleTools tables and views are part of the PeopleSoft application and are delivered along with the PeopleSoft system. They are application catalog tables which store attributes for fields, records, panels, panel groups, menus, PeopleCode, and other PeopleSoft objects. PeopleTools catalog tables and views are updated when a developer creates, modifies, or deletes a PeopleSoft object. Remember, PeopleSoft objects are building blocks for the online system. The Application Processor assembles these building blocks by accessing data from PeopleTools tables and presents the information in a graphical representation online.

Table 8.2 PeopleTools catalog tables

Objects	PeopleTools Table
Fields	PSDBFIELD
Records	PSRECFIELD, PSRECDEFN, PSINDEXDEFN, PSKEYDEFN, PSRECDDLPARM, PSIDXDDLPARM
Panels	PSPNLFIELD, PSPNLDEFN
Panel Groups	PSPNLGRPDEFN, PSPNLGROUP
Menus	PSMENUDEFN, PSMENUITEM
PeopleCode	PSPCMPROG, PSPROGNAME, PSPCMNAME

These PeopleTools catalog tables listed in table 8.2 are stored in the database as well. Let's look at how these PeopleTools catalog tables are updated.

Suppose a developer creates a record definition by performing the following tasks:

- create the schema for the record
- create fields, if necessary for the record
- define record properties

- define record field properties
- save the record definition
- build the record in the database (tables and views)

When the developer saves the record definition, the definition is stored in PeopleTools catalog tables.

The following PeopleTools tables are updated in this process:

- *PSDBFIELD* is the PeopleTools table that stores field definitions. A row is added into this SQL table each time the developer creates a new field.
- *PSRECDEFN* is the PeopleTools table that stores record definitions. A row is added into this SQL table each time the developer creates a new record.
- *PSRECFIELD* is the PeopleTools table that stores record field definitions. A row is added into this SQL table for each field in the record definition.
- *PSINDEXDEFN* is the PeopleTools table that stores index definitions for SQL tables. A row is added into this SQL table for each index defined for the record.
- *PSKEYDEFN* is the PeopleTools table that stores definitions of columns that are defined as indexed fields. A row is added for each column that is defined as indexed columns for each index.
- *PSRECDDLPARM* is the PeopleTools table that stores database specific parameters for the database table.
- *PSIDXDDLPARM* is the PeopleTools table that stores database specific parameters for the table indexes.

NOTE PeopleTools tables and views are not prefixed with PS_ because PeopleTools tables and views are given a non-standard SQL table name in the Type tab under the record properties. Application tables and views are prefixed with PS_ when the corresponding SQL table or view is created in the database. For example, if the record is named PERSONAL_DATA in PeopleSoft, the corresponding SQL table in the database is called PS_PERSONAL_DATA.

8.1.3 Application tables and views

Users maintain business data in the application. User data are stored in application tables and views. When the user accesses the online application, data from application tables and views are presented online. Application tables and views are created as database objects.

Definitions for application tables and views are stored in PeopleTools catalog tables. The PeopleTools object definitions for application tables and views are called records. Records can also be query views, dynamic views, subrecords, and derived/work records.

Records defined as SQL tables and views are built into the database simply because tables store application data, and views are representations of data. Other PeopleSoft objects—such as panels, panel groups, menus, PeopleCode, and so on—process and present application data online. Every time the users access the online application, these objects are built online. They do not exist in the database.

Records defined as SQL tables permanently store data in the database. Record definitions are created online using PeopleTools. At this point, the record exists in PeopleSoft, but is not yet a database table. As it is, the record definition cannot store any data in the database. When the developer builds the record, a database table which can store data entered using the online application is created. Similarly, records defined as SQL views are created using PeopleTools as well.

The developer must define parameters before records can be built in the database. Records defined as SQL tables contain build parameters used in the build process. Build parameters vary across database platforms.

TIP DDDAUDIT is an SQR process which identifies records defined as tables and views, which do not exist in the database and vice versa.

Let's look at the definition for MY_PROBLEM_TRKG record (figure 8.1).

MY_PROBLEM_TRKG is defined as an SQL table in the application. In the database, the corresponding database table is named PS_MY_PROBLEM_TRKG. When the developer builds the actual SQL table in the database, other related database objects

Navigation: File →Open →Record →MY_PROBLEM_TRKG →File →Object Properties

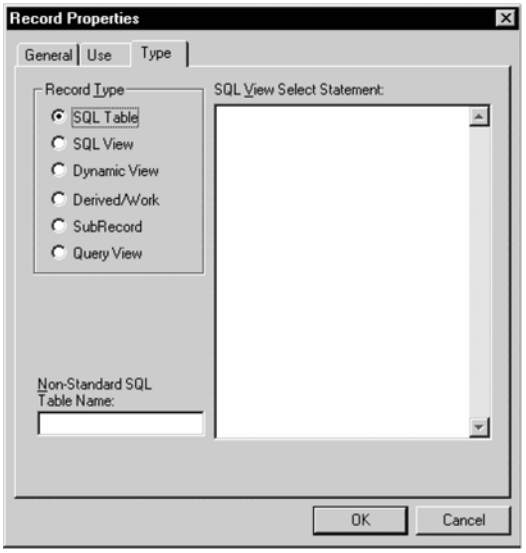


Figure 8.1
Record definition (SQL Table)—Type tab

are created as well. SQL tables also have indexes to facilitate faster access to data stored in them. Indexes are also database objects. When database objects are created, they update database catalog tables.

Now let's take a look at the definition for MY_TRKG_STATUS record (figure 8.2).

Navigation: File →Open →Record →MY_TRKG_STATUS →File →Object Properties

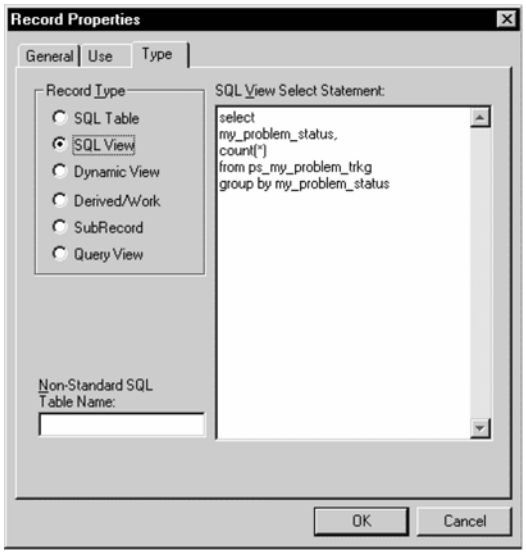


Figure 8.2
Record definition (SQL View)—Type tab

MY_TRKG_STATUS is a record defined as an SQL view. In the database, this object is called PS_MY_TRKG_STATUS. This SQL view represents data from PS_MY_PROBLEM_TRKG table in a Totals format. The SQL view is built using the SQL View `Select` Statement.

Before we describe how application tables and views are built in the database, let's consider the data modeling tools that PeopleSoft delivers as part of PeopleTools.

8.2 DATABASE OBJECT MODELING

PeopleSoft delivers tools which allow a database administrator to define data models. PeopleSoft objects which are stored as database objects, use the defined models as defaults. The database administrator can change the defaults and override parameters based on application needs.

When the PeopleSoft system is installed, these database models are updated to suit the defaults for majority of application tables and indexes. This way, the database administrator has to override parameters only for tables and indexes that cannot use

the default storage parameters. Data models are different across database platforms. PeopleSoft delivers platform-specific parameters based on the platform of installation. If, for example, the installation is an Oracle installation, the modeling tools automatically supply parameters used in an Oracle database.

Let's look at the tools used for data modeling in PeopleSoft. Figures 8.3 and 8.4 illustrate data modeling defaults for SQL tables and indexes in an Oracle platform.

Navigation: Go →PeopleTools →Utilities →Use →DDL Model Defaults

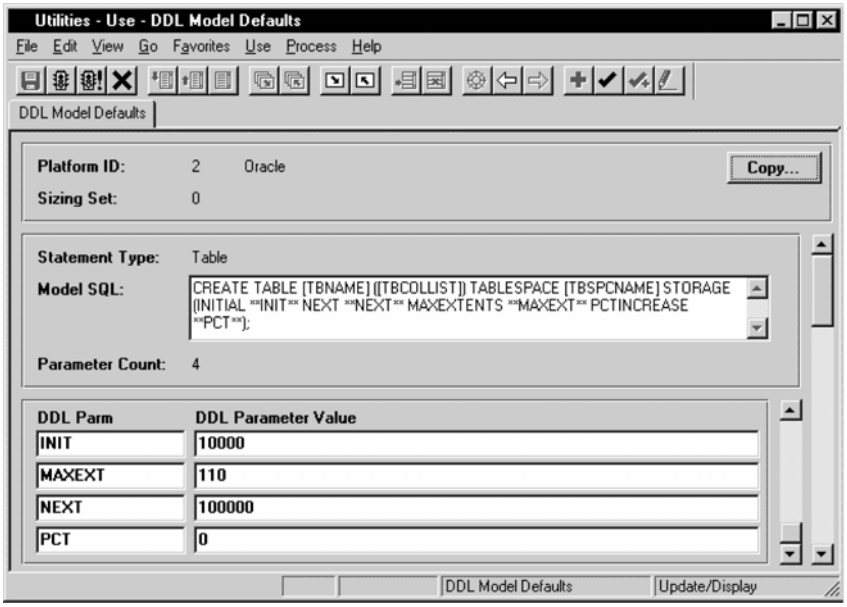


Figure 8.3 DDL model default—table

Platform ID is unique for each database platform. PeopleSoft supplies the following platforms for data modeling:

0 - SQLBase
1 - DB2
2 - Oracle
3 - Informix
4 - DB2/Unix
5 - Allbase
6 - Sybase
7 - Microsoft SQL Server
8 - DB2/AS400

In figure 8.4 the first scroll bar contains data modeling parameters for *all* database objects. The second scroll bar contains individual parameters used for modeling *each* database object. These parameters come predefined, and the DDL parameter value can be changed for each parameter.

Navigation: Go →PeopleTools →Utilities →Use →DDL Model Defaults

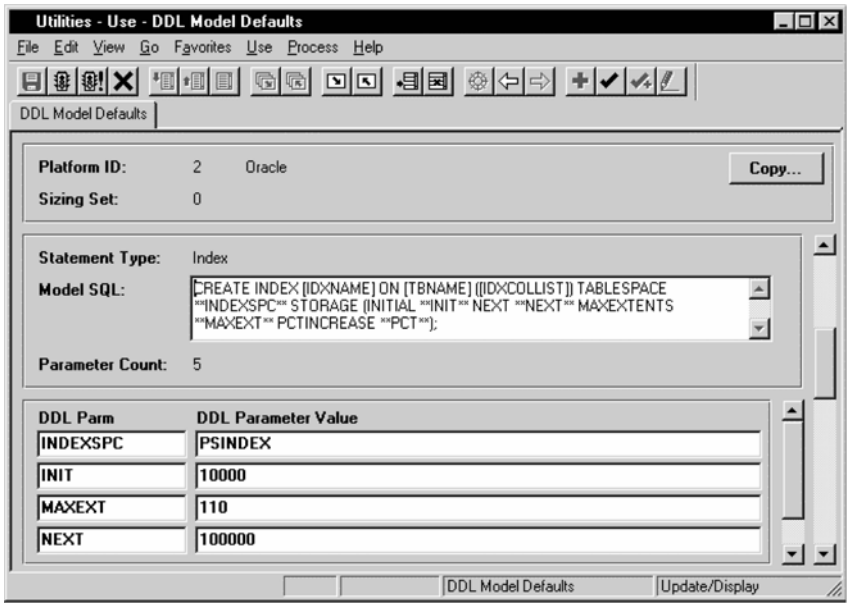


Figure 8.4 DDL model default—index

Data modeling parameters can be overridden for each record definition using the Data Administration option in Application Designer. The database administrator has to estimate data storage parameters for individual application tables and indexes and, if default parameters are not suitable, override them.

8.3 BUILDING DATABASE TABLES AND VIEWS

Record definitions defined as SQL tables or views can be built in the database using the Application Designer. A record definition must be built in the PeopleSoft system first before it can be built as a database object.

Record definitions are defined as SQL tables or views through the Use tab in the Record Properties screen, and attributes defined before an SQL table or view can be built in the database. Let's use the MY_PROBLEM_TRKG record from our Problem Tracking application to understand this progression (figure 8.5).

Navigation: File →Open →Record →MY_PROBLEM_TRKG →File →Object Properties

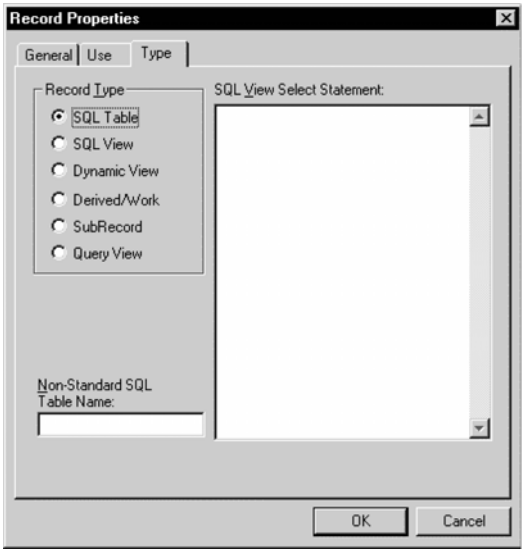


Figure 8.5
Record Properties—Use tab

First we list the steps required to build a PeopleSoft record definition in the database:

- define the record definition type
- define the database keys
- define DDL parameters for the table
- define DDL parameters for indexes
- build the object in the database

8.3.1 Define the record definition type

As we can see in figure 8.5, the Record definition must be defined as an SQL table or an SQL view. We do this through the Use tab in the Record Properties screen. When the record definition is defined as an SQL view, the record definition can be built in the database at any time without any fear of losing data because SQL views are just representations of data from SQL tables. They do not store actual data in the database.

8.3.2 Define the database keys

Our next step is to define the database keys for the SQL table or view. We do this using the Application Designer tool. Remember, the record definition must be open to perform this step (figure 8.6).

When the record definition is first built using fields from the PeopleSoft system, the developer should have an idea of the fields that will be used to build the database index. Database keys perform best when they are placed one after another. For this

Navigation: File →Open →Record →MY_PROBLEM_TRKG

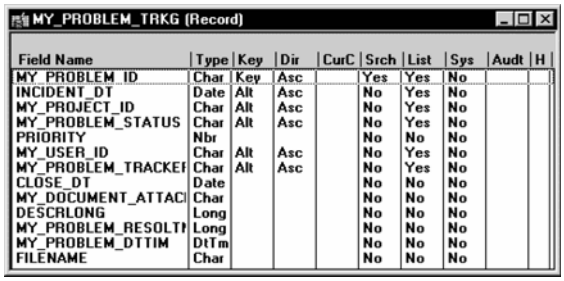


Figure 8.6
Record definition—Use display

reason, we place the database keys in sequence in the record definition. By highlighting each field and choosing Edit/Record field Properties from the Application Designer menu, we can define a record field as the database key (figure 8.7).

Navigation: Highlight Record Field →Edit →Record Field Properties

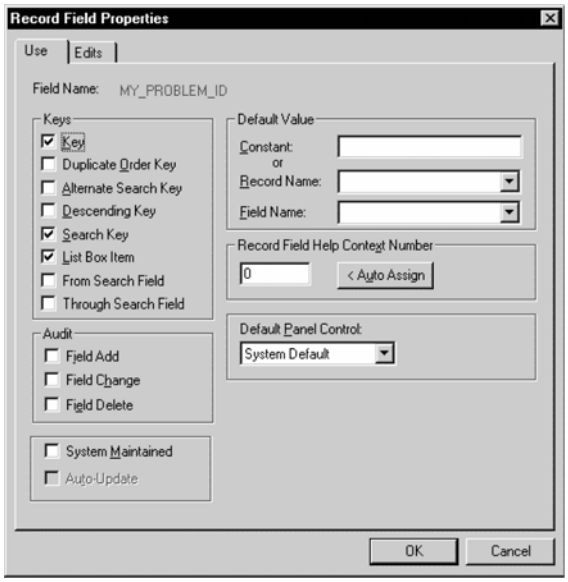


Figure 8.7
Record Field Properties

Record fields can be defined as database keys by clicking on the Key checkbox. The other options are Duplicate Order Key for non-unique indexes, Alternate Search Key, and Descending Key. Alternate Search Keys are also used to create non-unique indexes in the database. They appear in the online application in list boxes as well. Descending Keys are simply database keys presented in a descending order. They can be part of either a unique or a non-unique index. All the record fields, which are

database keys can be defined as such by highlighting the appropriate field and choosing "Edit →Record Field Properties" from the Application Designer menu.

8.3.3 Define DDL parameters for the table

Data Definition Language (DDL) defines the parameters needed to create the object in the database. Parameters such as storage parameters, data increments, and table space parameters are defined in this step. DDL parameters vary according to database platforms. We will use Oracle as the database platform for our purposes. Figure 8.8 illustrates the Maintain Record DDL screen where DDL parameters are entered.

Navigation: Tools →Data Administration →Record DDL (Record Definition is open)

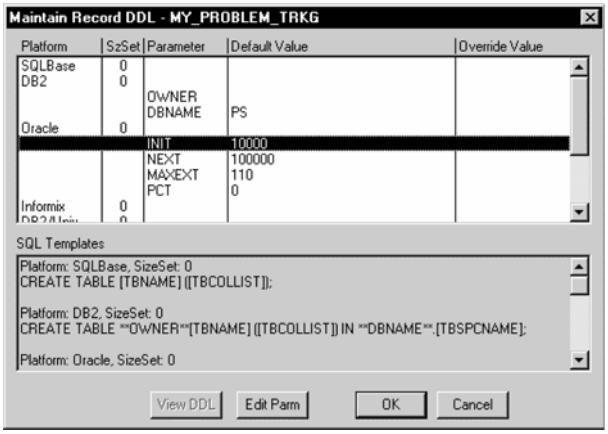


Figure 8.8
Record DDL Parameters

Navigation: Double Click on a DDL Parameter (From Maintain Record DDL screen

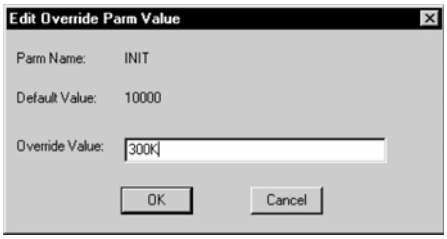


Figure 8.9 Override DDL parameter

Oracle database requires parameters such as `Initial Extent`, `Next Extent`, `Maximum Extents` and `Percentage of Increase`. They can be defined by double-clicking on each of those parameters under the Oracle section (figure 8.8). The default values seen in figure 8.8 are defined in the DDL Model Defaults panel under the Utilities menu. The defaults are not, however, always suitable for all tables in the database. We can enter an override to these parameters before the object is created in the database.

All DDL parameters can be overridden by entering the override value, choosing the OK button, and saving the record definition (figure 8.9). DDL parameters are

saved in the PSRECDDLPARM PeopleTools catalog table. As we said, DDL parameters vary across database platforms, so only parameters applicable to your database should be changed before the object is created.

8.3.4 Define DDL parameters for indexes

Index DDL parameters can be changed as illustrated in figure 8.10.

Navigation: Tools →Data Administration →Indexes (Record Definition is open)

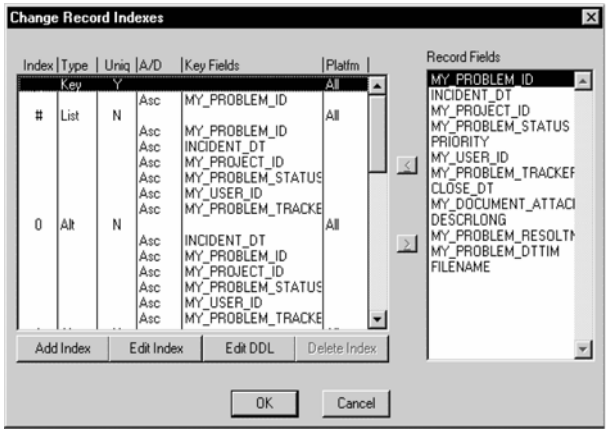


Figure 8.10
Change index DDL parameters

Based on the key defined for the record definition, PeopleTools determines the number of database indexes that must be created in the database. Primary indexes are given the same name as the table name. Other indexes are named with a numerical sequence added. In our example, the primary index for MY_PROBLEM_TRKG table is named PS_MY_PROBLEM_TRKG. The other indexes are named PS0MY_PROBLEM_TRKG, PS1MY_PROBLEM_TRKG, and so on.

By choosing the Edit DDL button, we can override DDL parameters for each of these indexes. Before the Edit DDL button is pushed, each index is highlighted. Index DDL parameters also vary across database platforms (figure 8.11). The Edit Index button can be used to change the index uniqueness. The index can also be turned off on some database platforms using the Edit Index button.

Navigation: Edit DDL (From Change Record Indexes screen)

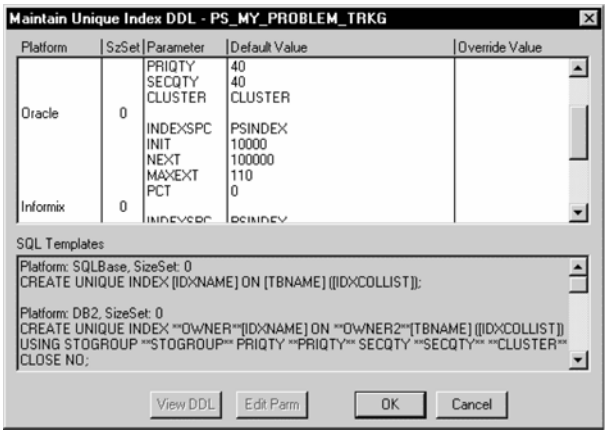


Figure 8.11
Index DDL parameters

Navigation: View DDL (From Index DDLParameters Screen)

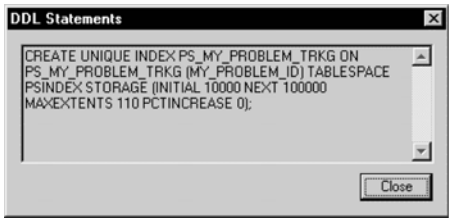


Figure 8.12 View DDL

The parameters in figure 8.11 are the same as the DDL parameters for tables—with one exception. The table spaces for indexes are overridden here. Each of these parameters can be overridden either by double-clicking on the parameter or highlighting the parameter and choosing the Edit Parm button. We can use the View DDL button to look at the actual DDL statement that will be used to create the index in the database (figure 8.12).

8.3.5 Build the object in the database

Once we specify the record definition as an SQL table or view, we must build the record in the database. Since we have defined the database keys and DDL parameters, it's time to build the record in the database. Before we can do so, we still have to define the table space parameter for an SQL table. This parameter is not required in all database platforms.

The table space parameter can be changed as illustrated in figure 8.13.

All tables built under the table space chosen are displayed in the list. Before we build the object in the database, we must save the record definition by clicking on the Save icon from the Application Designer menu.

We build the current open object by choosing Build →Current Object from the Application Designer menu. Two options appear on the Build Object screen. They are Build Options and Build Execute Options. Build Options is used to control the

types of database objects built; Build Execute Options controls how the database objects are built.

Now let's consider the options—and the reasons we'd choose them—under Build and Build ExecuteOptions (figure 8.14).

Navigation: Tools →Data Administration →Set Table Space (Record Definition is open)

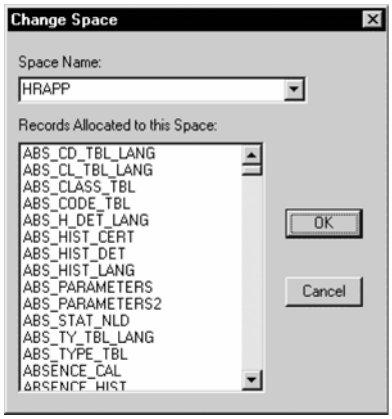


Figure 8.13 Set table space

Navigation: Build →Current Object (Record Definition is open)

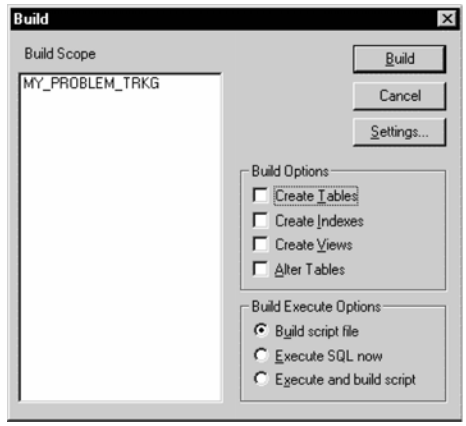


Figure 8.14 Build objects

Build options

- *Create Tables* Choose this option to build all records defined as SQL tables in the Build Scope list.
- *Create Indexes* Choose this option to build all indexes for SQL tables in the Build Scope list.
- *Create Views* Choose this option to build all records defined as SQL views in the Build Scope list.
- *Alter Tables* Choose this option to alter schema for records defined as SQL tables in the Build Scope list.

TIP When the Alter Tables option is chosen, only those SQL tables that require an alter in the Build Scope list will be altered. Tables which require alteration are determined by comparing the record and database definitions for the tables.

Build Execute options

- *Build Script File* Choose this option to build a script file before executing the SQL script to create or alter the database object.
- *Execute SQL Now* Choose this option to execute DDL SQL statements and create or alter the database object immediately. This option may not be available for SQL Alters in some database platforms.
- *Execute and Build Script* Choose this option if you want to execute and build the DDL SQL statements at the same time.

TIP It is prudent to always build a script file before execution. The script files can be used for review before execution. The script files can also be used as change documents.

Figure 8.14 shows the Settings button used to control recreate options, script creation options, and logging options. Figures 8.15 through 8.18 illustrate all four tabs in the Build Settings screen.

Under the Create tab, (figure 8.15) we either choose to recreate the object if it already exists or skip re-creation. This option is applicable to objects already present in the database. Use Table Creation Options for SQL tables and View Creation Options for SQL views.

Navigation: Settings (From the Build Object screen)

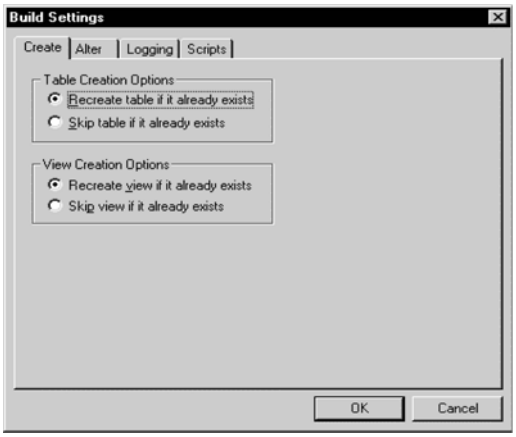


Figure 8.15
Build Settings—Create tab

Under the Alter tab (figure 8.16), we enter settings specific to altering SQL tables.

Navigation: Settings (From the Build Object screen)

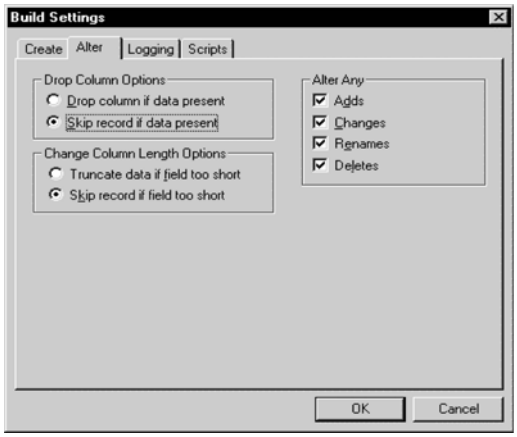


Figure 8.16
Build Settings—Alter Tab

Drop Column Options

- *Drop column if data present* Choose this option if you want to delete a column as part of SQL Alter, even if the column contains data in it.
- *Skip record if data present* Choose this option if you want to skip altering a table when data are present in a column and the column is being dropped.

Change Column Length Options

- *Truncate data if field too short* Choose this option to truncate data when a field length is changed, and, the data in the column are larger than the new column length.
- *Skip record if field too short* Choose this option if you want to skip the SQL Alter for the record when column lengths are changed, and data in the column are larger than the new column length.

Alter Any

- *Adds* Choose this option to add new columns during SQL Alters.
- *Changes* Choose this option to change column lengths during SQL Alters.
- *Renames* Choose this option to rename columns during SQL Alters.
- *Deletes* Choose this option to drop columns during SQL Alters.

During SQL Alter, the Build Settings specific to the Alter process is verified to determine whether to proceed to alter a record or to skip the record.

NOTE Alter Any settings vary across database platforms. Some database platforms do not allow changing of column lengths without re-creation of tables.

Under the Logging tab, we specify options for logging the results from the Build process (figure 8.17).

Navigation: Settings (From the Build Object screen)

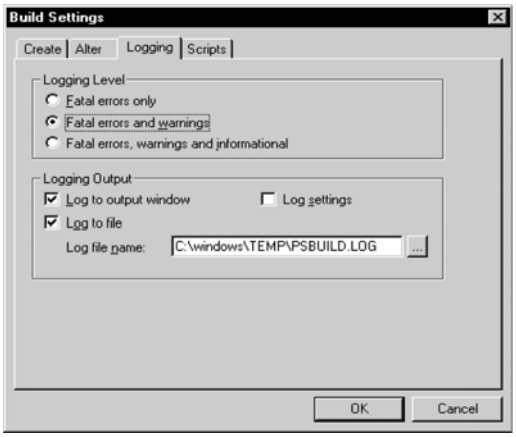


Figure 8.17
Build Settings—Logging tab

Logging level

- *Fatal Errors Only* Choose this option when you want only the fatal errors to be logged during the process.
- *Fatal Errors and Warnings* Choose this option when you want both fatal errors and warnings to be logged during the Build process.
- *Fatal errors, warnings, and informational* Choose this option to log all results during the Build process.

Logging output

- *Logging to Output Window* Choose this option if you want the log to be displayed on an output window.
- *Log to File* Choose this option to log files to a text file. You can specify the full path and the file name for the log file in the Log file name box.
- *Log settings* Choose this option to log all settings at the time the Build process was executed.

Under the Scripts tab (figure 8.18), we specify options to build a script file.

Navigation: Settings (From the Build Object screen)

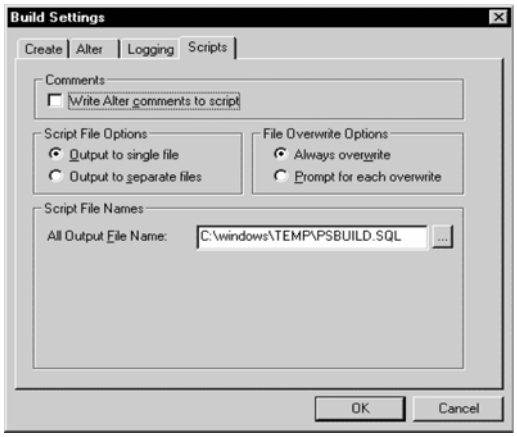


Figure 8.18
Build Settings—Scripts tab

Write Alter comments to script

Choose this checkbox to see Alter comments in the script file. During the Build process, information is written about changes to individual fields right above the actual Alter SQL statement. This checkbox is useful especially when we need to alter a number of columns in the same SQL table.

Script File Options

- *Output to single file* Choose this option if you want to build a single script file for SQL Table Creates, SQL Table Alters, SQL View Creates, and Index Creates.
- *Output to separate files* Choose this option if you want to create separate files for SQL Table Creates, SQL Table Alters, SQL View Creates, and Index Creates. When this option is chosen, four boxes appear under the Script File Names option to accommodate separate filenames.

File Overwrite Options

- *Always overwrite* Choose this option if you want to overwrite existing Build scripts with the same name.
- *Prompt for each overwrite* Choose this option if you want to be prompted during the build process, before scripts are overwritten.

TIP Because SQL views do not store actual data, they can be re-created whenever necessary.

NOTE PeopleSoft performs SQL Alters by re-creation in some database platforms. During this process, PeopleSoft creates a table called PS_1 used to temporarily hold data from the SQL table being altered. The SQL Alter process may fail if the table being altered consumes considerable storage space. Review the `Build` script to ensure that the SQL table PS_1 can hold data from the SQL table being altered.

TIP Record DDL parameters have to be constantly evaluated and updated as data grow in the database. It is prudent to keep record DDL parameters in PeopleSoft in sync with the storage parameters for the corresponding SQL tables in the database.

KEY POINTS

1. Records defined as SQL tables or SQL views are also database objects and have to be created in the database.
2. Table and Index DDL parameters have to be set before the object can be built in the database.
3. When altering tables, we recommend that you build a script file. The script file has to be reviewed before execution in the database.
4. SQL views can be re-created at any time.
5. DDL Model defaults are used as default DDL parameters to build objects in the database.

CHAPTER 9

PeopleSoft Application Processor

In the previous chapters, we described concepts behind the development of a PeopleSoft application panel. We also explained how security is provided to users to access application panels. This chapter primarily focuses on how PeopleTools processes an application panel.

Application Processor organizes the numerous individual processes that occur from the time a panel group is requested to the time the panel group is saved. It is important to understand these individual processes and the sequence in which the Application Processor organizes them. This knowledge helps us design and develop objects more intuitively to suit business needs.

In each stage of the Application Processor (figure 9.1) a number of PeopleCode events are executed. In this chapter, we will walk through the Application Processor stages using the Track Problems menu item from our Problem Tracking application as

an example. As we do, the sections will switch back and forth between `Add` and `Update/Display` modes. The reader will need to correlate screens in the correct order to follow an entire mode from start to finish. (Part 3 describes in detail the individual PeopleCode events as well as more about sequences in which these events are executed.)

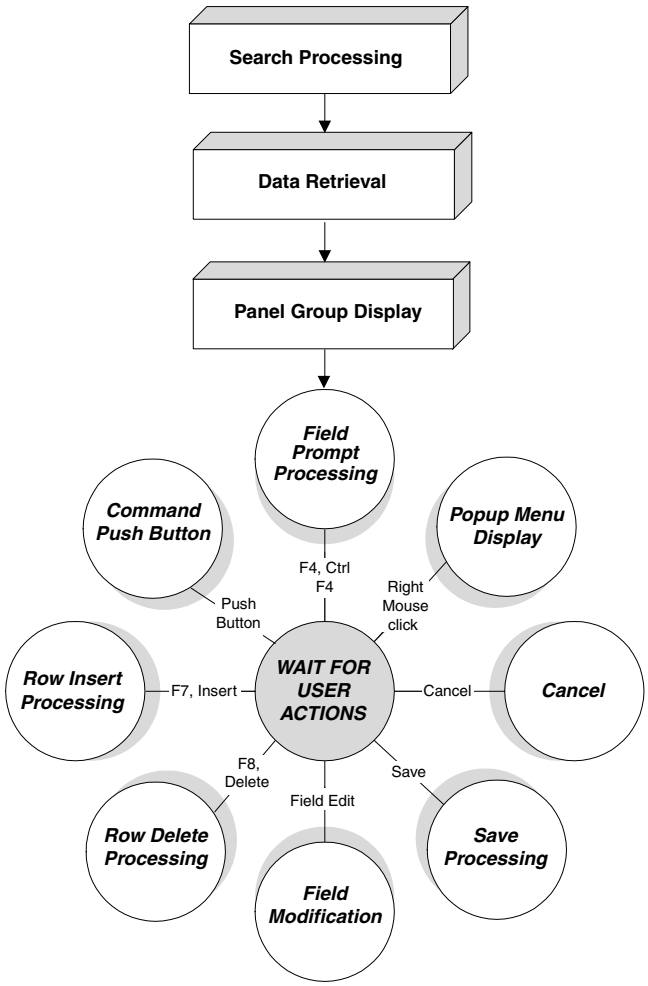


Figure 9.1
Stages in application processing

NOTE The key to application processing is the panel group object. The panel group definition provides vital information throughout all stages to process the application panel.

9.1 SEARCH PROCESSING

Search Processing takes place when the user chooses the menu item that accesses the application panel group. This stage begins by building the search dialog box used to access the panel group and ends after fields for data retrieval are populated and saved into the search fields. The Search Processing stage can be further divided into multiple steps. Let's walk through all possible steps that can occur during this stage.

- determine mode of access
- retrieve panel group definition
- determine search fields
- populate and display search record fields
- edit user inputs into search dialog fields
- populate search buffer for data retrieval

9.1.1 Determine mode of access

The Application Processor determines the mode in which the user accesses the panel group. All authorized modes necessary to access the panel group are pre-determined when the application menu is brought up. Users can view only authorized panel group actions when they access the application menu. The Application Processor determines authorized panel group actions from a PeopleTools catalog table called PSAUTHITEM. Let's look at our example and find out what the authorized actions are for the menu item.

In figure 9.2, this user has two different actions/modes authorized for this menu item. The panel group definition screen will show all the authorized actions for the menu item. Based on the action chosen by the user, the Application Processor determines the record used for search processing.

9.1.2 Retrieve panel group definition

Search records are attached to panel group definitions. Therefore, the Application Processor has to retrieve the panel group definition to determine the search record. PSPNLGRPDEFN is the PeopleTools catalog table which contains the search records for a panel group. We can define two different search records for a panel group: a search record used in `Add` mode and a search record for all other modes. Figure 9.3 shows the panel group properties screen. Here we can see how search records and authorized actions are attached to panel groups.

The Panel Group Properties screen contains the two search records and the authorized actions available for the panel group. When Add search record is left blank, the regular search record is used for all authorized actions. `Add` and `Update/Display` are the two authorized actions available for this panel group. A panel group, however, can have more authorized actions than a user can see when the actual application menu item is accessed. Security to the menu item and corresponding actions are provided using the Security Administrator screen. In the example, we access this

Navigation: Go →Problem Tracking →Tracking →Track Problems

Figure 9.2 Authorized action for a panel group

Navigation: File →Object Properties (MY_PROBLEM_TRKG panel group is opened first)

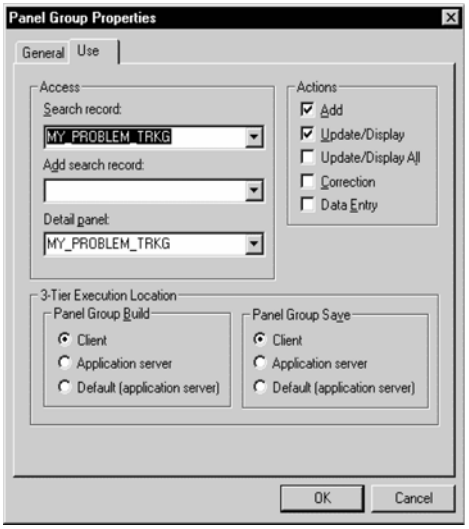


Figure 9.3
Panel Group Properties—Use tab

menu item using the PS operator that belongs to the ALLPANLS operator class. Figure 9.4 shows the Security Administrator screen where security is provided to ALLPANLS to access this menu item.

Navigation: File →Open →ALLPANLS (From Security Administrator screen)
Double-click on PROBLEM_TRACKING line item under the Menu Items tab

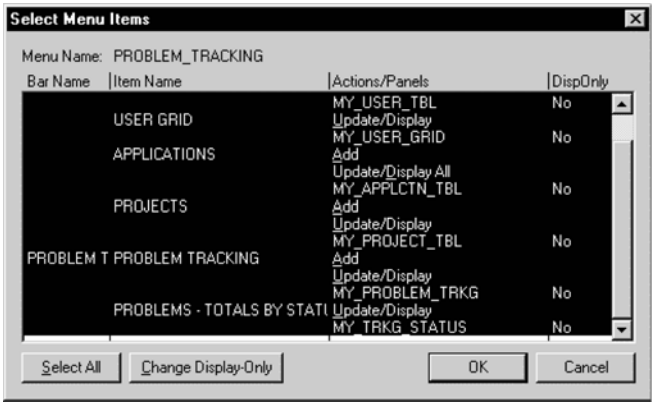


Figure 9.4 Authorized menu items and actions

All authorized menu items are highlighted in figure 9.4. In our example, we can either choose Add or Update/Display actions. Next, we will see how the Application Processor reacts when either action is selected.

9.1.3 Determine search fields

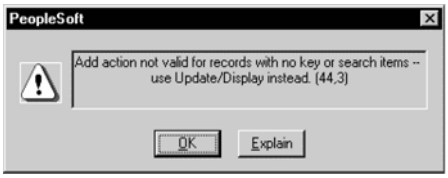


Figure 9.5 Search records with no search keys

The Application Processor determines the search fields that should appear on the input dialog box. Fields defined as search keys in the search record are assembled on the input dialog box. Fields defined as alternate search keys in the search record are also assembled on the input dialog box in all modes except Add mode. When no search keys are defined, the Application Processor proceeds directly to the data retrieval stage after verifying that the user is accessing the application panel in Add mode. When a mode other than Add is selected, the error message in figure 9.5 is issued.

In PeopleSoft HRMS, the INSTALLATION record can be used as the search record to access the panel group directly, without an input dialog box. As we saw in the previous error message, we cannot use Add mode to access panel groups which contain search records with no search keys.

TIP Search records that do not have search keys can be processed using `Update/Display` mode only. This is useful when you want to bring up an application panel without an input dialog box.

Before we continue with our application panel example, let's look at the definition for MY_PROBLEM_TRKG record.

Navigation: File →Open →Record →MY_PROBLEM_TRKG →View →Use Display

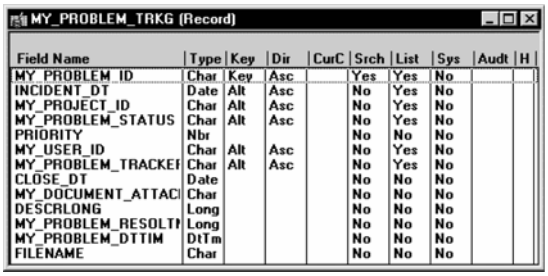


Figure 9.6
Search key definition for a search record

Navigation: Go →Problem Tracking →Tracking →Track Problems →Add

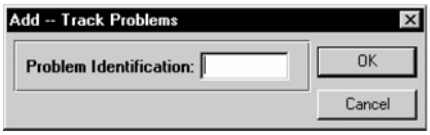


Figure 9.7 Search fields in Add mode

In our record definition, we have one field defined as the search key and five fields defined as alternate search keys. The panel processor fetches the search record definition in order to determine the search keys that will be presented on the input dialog box. In figure 9.7, we can see which fields are brought up on an input dialog box when the menu item is accessed in `Add` mode.

In `Add` mode, all fields defined as search keys are presented to the user in the input dialog box. MY_PROBLEM_ID is the only field defined as a search key. Now let us access the menu item using `Update/Display` mode (figure 9.8).

In addition to search fields, alternate search fields are also presented on the input dialog box in `Update/Display` mode. We notice another difference in the input dialog box between `Add` and `Update/Display` modes: a list box is provided to display search results, once search field values are saved.

Navigation: Go →Problem Tracking →Tracking →Track Problems →Update/Display

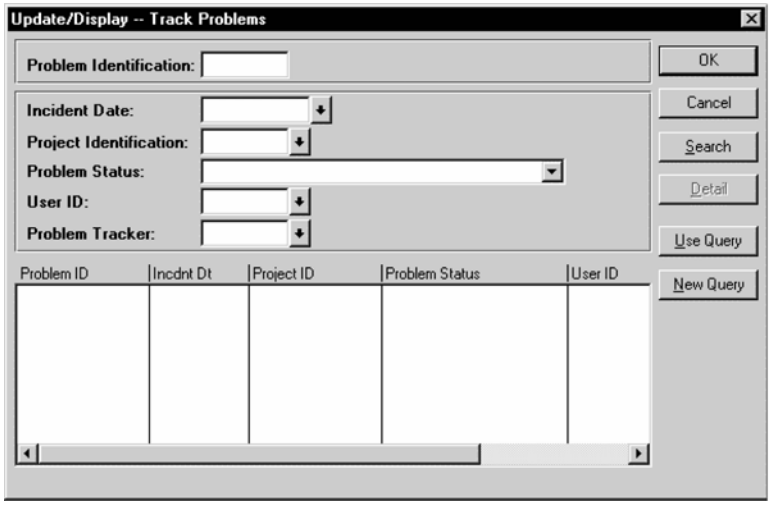


Figure 9.8 Search fields using Update/Display

9.1.4 Populate and display search fields

Before the Application Processor brings up the search/input dialog box, it executes certain PeopleCode events, which populate default values into the search fields. Any default values that are attached to fields in search records are used.

In Add mode `FieldDefault`, `RowInit`, and `SearchInit` PeopleCode events are executed (in that order) for search and alternate search fields. In other modes, `SearchInit` PeopleCode events are executed, and values are populated into search fields. We can attach PeopleCode to these events on the search record to see how they populate search fields before displaying them.

Figure 9.9 illustrates how PeopleCode is added to the `FieldDefault` event for MY_PROBLEM_ID field.

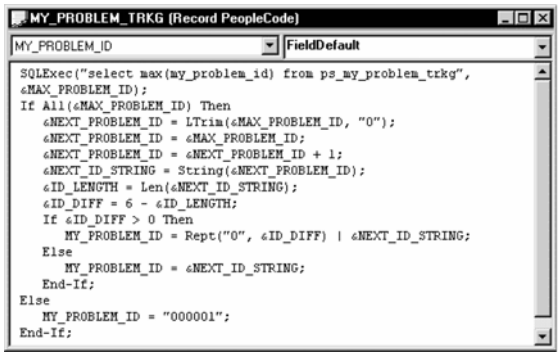


Figure 9.9
FieldDefault PeopleCode event on a search field

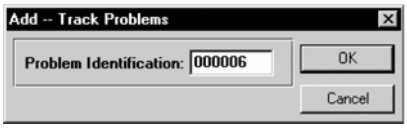


Figure 9.10 FieldDefault PeopleCode event in Add mode

This PeopleCode increments the MY_PROBLEM_ID field automatically. We only want to increment MY_PROBLEM_ID field in `Add` mode. `FieldDefault` is executed on a search record only in the `Add` mode. We do not want to default anything to MY_PROBLEM_ID in other modes. We use the `Update/Display` mode to update problem IDs which have been already created using the application. Let us see how the Application Processor executes this PeopleCode event and displays it on the input dialog box (figure 9.10).

We see that `FieldDefault` was executed and the next problem ID is now displayed on the input dialog box. `FieldDefault` is an iterative event that is constantly processed. `FieldDefault` PeopleCode event executes any time the field is blank. `RowInit` PeopleCode event executes any time the field is displayed either on the input dialog box or on the panel.

9.1.5 Edit search fields

The next step in search processing occurs when the user enters or overrides values in the search fields. The Application Processor performs internal field format-checking upon data entry into search fields. For example, when the user enters characters into fields defined as numeric, the Application Processor issues a message to the user. When search fields are defined as required fields on the search record, the user must enter values into such fields before saving the input. The following types of edits take place when data are entered into search fields:

- field format edits
- required field edits
- field modification PeopleCode edits (Field Edit)
- search/save PeopleCode edits (Save Edit, Search Save)

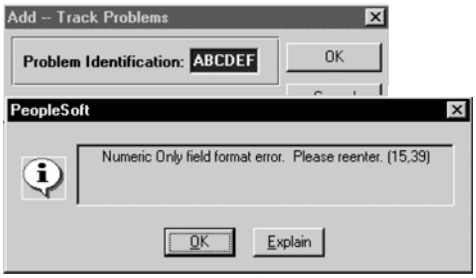


Figure 9.11 Field format edits

We will illustrate a few types of PeopleCode edits in this section. We'll also show you all the non-PeopleCode edits that the Application Processor validates.

Figure 9.11 illustrates how field formats are verified by the Application Processor. Similarly, other format types like date, time, name, and so on, are edited.

In figure 9.12, the Application Processor issues a message when a required field is blanked out. All search key fields presented on the input dialog box are usually defined as required fields. This is especially useful when the search field appears on the panel as a display field. However, when the field is displayed as an input field or when

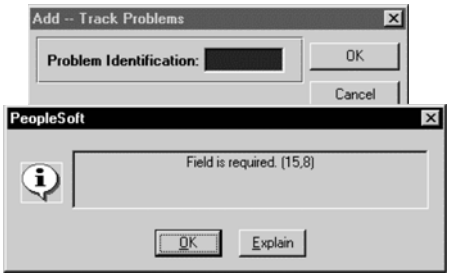


Figure 9.12 Required field edit

a PeopleCode event populates the field, a required field edit is not necessary.

Again, certain PeopleCode events are executed in this step.

In `Add`, when the user enters values into the search fields, `FieldEdit` and `FieldChange` PeopleCode events from search fields are executed. When the user saves the input by clicking on the OK button, `SaveEdit` and `SearchSave` PeopleCode events are executed from search fields.

TIP Search fields that are not input fields on the panel can be defined as Required fields. This prevents the user from saving the panel with empty keys in `Add` mode.

In `Update/Display`, when the user enters values into search fields, no PeopleCode events are executed. However, when the user saves the input, `SearchSave` PeopleCode event is executed. In `Update/Display` mode, events are processed for both search fields and alternate search fields.

Suppose we want to issue a warning message when the user tries to override the problem ID assigned by the system. How can we do this? We can add a `SearchSave` PeopleCode to one of the search fields. In this case, the only search field in our search record is the MY_PROBLEM_ID field. Let us take a look at the PeopleCode we can write to perform this function (figure 9.13).

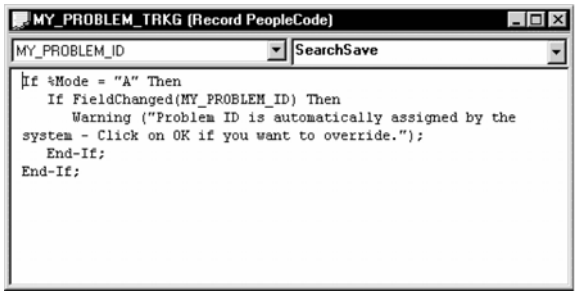


Figure 9.13
SearchSave PeopleCode

Because `SearchSave` PeopleCode event is executed in all modes, we use the `%Mode` system variable to execute this PeopleCode event in `Add` mode only. This PeopleCode event will issue a warning message when the user tries to change the problem ID assigned by the `FieldDefault` PeopleCode event.

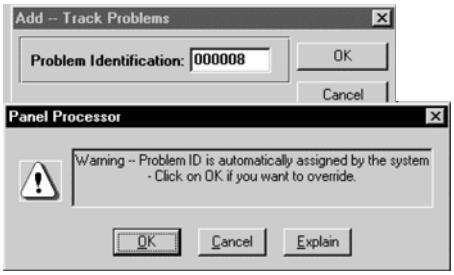


Figure 9.14 SearchSave PeopleCode event in Add mode

In figure 9.14, a warning message is provided with options to continue or to cancel. When the user chooses the OK button, the Application Processor proceeds to the next step for data retrieval. When the user chooses the Cancel button, the Application Processor brings the user back to the input dialog box for re-entry.

In chapter 13, "PeopleCode and the Application Processor," PeopleCode events in search processing are discussed in detail.

9.2 DATA RETRIEVAL

Data retrieval starts when the user populates the search dialog box and clicks OK. At this point, all edits, default processing, and search save processing has taken place, and the Application Processor uses the search keys to retrieve data.

What happens when the Search record does not have any search keys defined? In that case, all rows from the database are retrieved for panel group display. An example of this is the INSTALLATION table panel group in all PeopleSoft applications. The INSTALLATION table only has one row in it with no database keys defined.

The Application Processor performs the following operations during the data retrieval stage:

- verifies mode with data from search record
- prepares fields for the list box based on search keys
- prepares a list of panels that builds the panel group
- prepares a list of tables and views necessary to display the panel group in `Update/Display`, `Update/Display All`, and `Correction` modes
- retrieves data from the database for the complete list of tables and views

9.2.1 Verify mode with data from search record

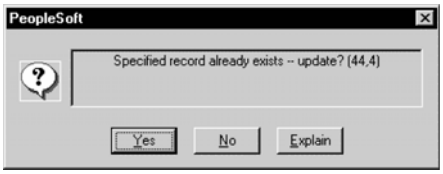


Figure 9.15 Validate mode with data in the database

The Application Processor validates the menu action (`Add`, `Update/Display`, and so on) that the user chooses with data from the search record. If the user chooses `Add` mode and the search record already has data matching the search fields, the Application Processor issues a message to that effect. If we choose `Add` mode and then choose `000001` as the problem ID, the Application Processor checks whether the search record has data matching those keys. In this case, the search record is MY_PROBLEM_TRKG. In the database, this table has a row that matches that problem ID. The Application Processor issues a message as illustrated in figure 9.15.

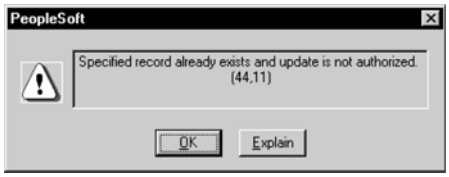


Figure 9.16 Update action not authorized

The message, however, enables the user to bring up the panel group in `Update/Display` mode. If the user is not authorized to choose `Update/Display` mode for the panel group, the Application Processor again issues an error message (figure 9.16).

In modes other than `Add`, the Application Processor proceeds directly to the next step to prepare list box items.

9.2.2 Prepare the list box

A list box is the result of search processing and data retrieval. All rows in the search record—in other words, the underlying database table or view—are presented on the list box. The list box is provided only when the Data Retrieval finds more than one row that matches the search fields. The Application Processor provides a list box when more than one row is found matching search fields; a partial search results in multiple rows; or no input is supplied for search.

We should be aware that when discussing search fields, we imply Primary Search Fields and Alternate Search Fields. The data retrieval stage attempts to match rows using both types. The results on the list box enable us to choose the data we want to view or update. Search fields and alternate search keys are automatically defined as list box items, but we can override this by turning off the definition for the list box. Any field from the search record except for Long Edit Boxes can be defined as a list box item.

In figure 9.17, all fields marked as list box items appear on the list box.

NOTE Search records can either be SQL tables or views. When search records are SQL views, the Application Processor accesses data from tables that were used to build the views.

Navigation: File →Open →Record →MY_PROBLEM_TRKG

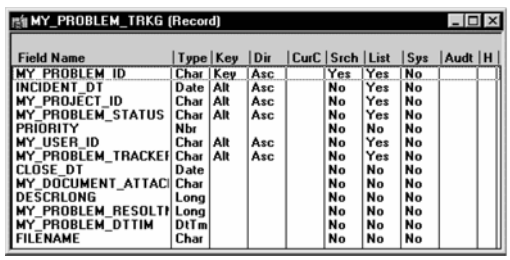


Figure 9.17
Definition of search record

Figure 9.18 illustrates a list box with all rows in the database table. We did not provide any input on the search dialog box. All rows from MY_PROBLM_TRKG table are displayed in the list box, a useful feature when the user is not sure what to enter for input.

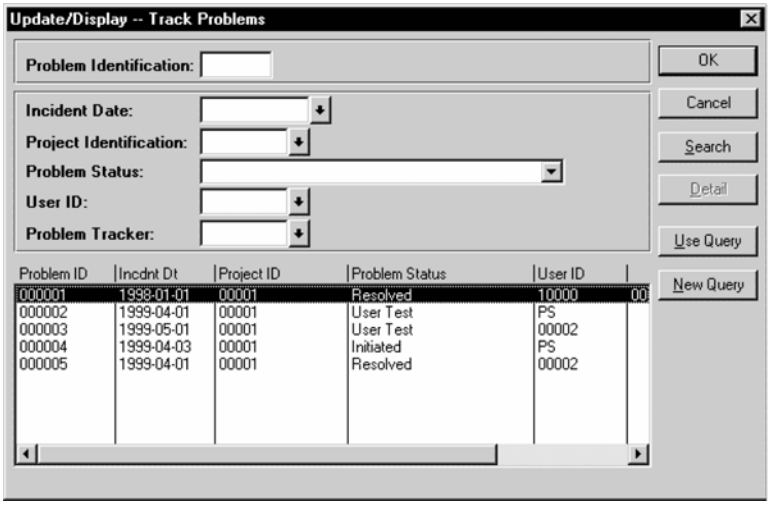


Figure 9.18
List box display with no input

Now, let's provide input on the input dialog box and see how the Application Processor selects rows in the list box (figure 9.19).

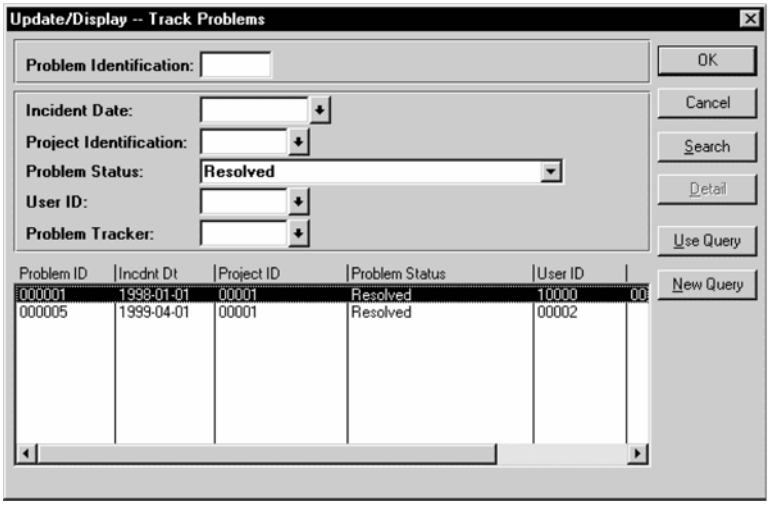


Figure 9.19
List box display with full key input

The example lists resolved incidents. Notice how the Application Processor uses the Problem Status field as input and displays all rows which match the status. We can also provide partial input in a search field. The Application Processor matches partial input and displays rows which match the partial input.

To perform this task, the Application Processor reads the record definition, and it retrieves information from PeopleSoft catalog tables, PSRECDEFN and PSRECFIELD. These catalog tables contain details on search fields and list box fields. The user can choose any line item from the list box and then view data on the panel.

Two push buttons—the OK and Search push buttons—trigger the Application Processor to populate a list box. When the OK button is chosen and a unique match exists, the Application Processor proceeds directly to displaying the panel. When the Search button is chosen and a unique match exists, the Application Processor displays the list box instead. Let us look at this process in figure 9.20.

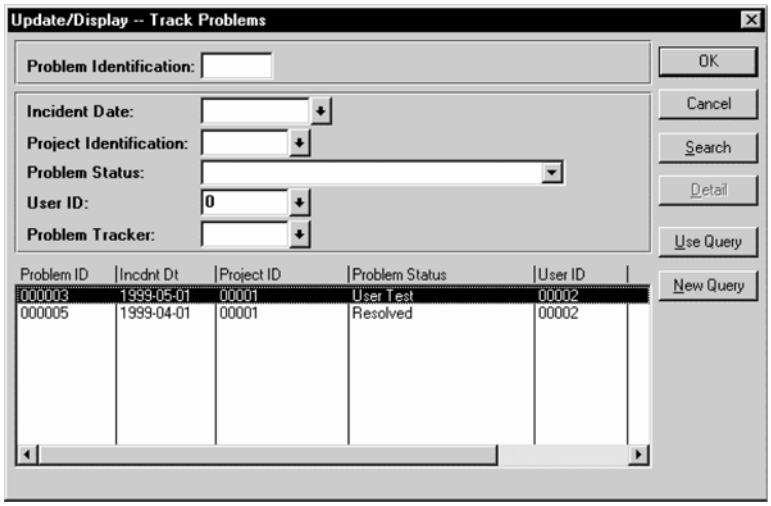


Figure 9.20 List box display with partial key input

In the example, we just provide a "0" in the User ID field, and the Application Processor finds all the rows that contained a User ID prefixed with a "0."

9.2.3 Prepare a list of panels

The Application Processor must still retrieve data to display on the panel. In order to do so, the Application Processor must access the panel group definition and prepare a list of panels. The Application Processor retrieves this information from a PeopleSoft catalog table called PSPNLGROUP. Remember, one or more panels may be in a panel group.

In this step, the Application Processor determines the order in which panels are displayed, the field layouts in a panel, the control display fields, the related display

fields, the secondary panels, the subpanels, and so on. All these related objects are attached to the panel definition. The PeopleSoft catalog table which stores this definition is called PSPNLFIELD.

In our example, the panel group MY_PROBLEM_TRKG contains only one panel. Figure 9.21 shows the number of panels in the panel group.

Navigation: File →Open →Panel Group →MY_PROBLEM_TRKG

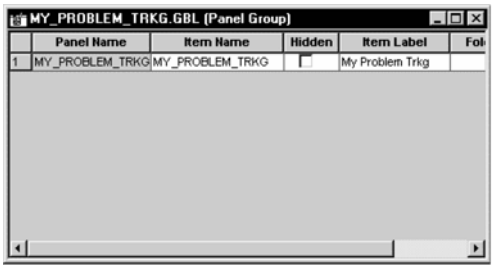


Figure 9.21
Panel group definition

After the Application Processor has prepared a list of panels and related objects, it proceeds to the next step.

9.2.4 Prepare a list of records and fields

The Application Processor starts preparing the records needed to build the panels. The records can be SQL tables, SQL views, or derived records.

In `Add` mode, the Application Processor retrieves less data from the database when compared to other modes. This is because, in `Add` mode the Application Processor invokes a new panel group. In `Add` mode, too, the Application Processor may need to retrieve values for related display fields, which are descriptions for search key fields.

In `Update/Display`, the Application Processor has to retrieve data for tables and views on the panel.

Derived records do not exist in the database. They are work records either populated by record field defaults or PeopleCode events. We can get a list of the records that MY_PROBLEM_TRKG panel contains by looking at the panel field layout. The panel field layout contains records and fields that make a panel.

In figure 9.22, we can see that MY_PROBLEM_TRKG panel displays fields from SQL tables and derived records. Some of these fields are related display fields which show descriptions for control display fields.

Navigation: File →Open →Panel →MY_PROBLEM_TRKG →Layout →Order

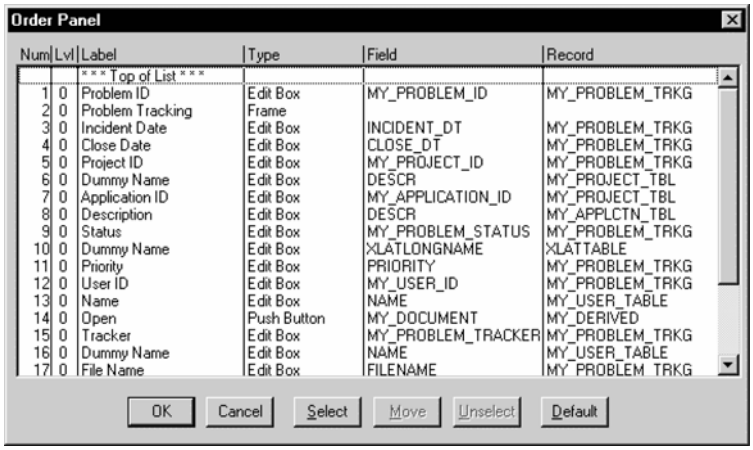


Figure 9.22
Panel field layout

9.2.5 Retrieves data from the database

In our next step, the Application Processor begins retrieving data for the list of tables and views on the panel. In `Add`, the Application Processor does not retrieve any data for our example. This is because the Application Processor displays a new panel and no related display descriptions are necessary on this new panel.

In `Update/Display`, the Application Processor retrieves data from MY_PROBLEM_TRKG table because, in modes other than `Add`, the Application Processor has to retrieve existing data for viewing and update from the database.

In `Add`, the table MY_PROBLEM_TRKG is accessed from the database.

In `Update/Display`, the following tables are accessed from the database:

- MY_PROBLEM_TRKG
- MY_PROJECT_TBL
- MY_APPLCTN_TBL
- XLATTABLE
- MY_USER_TABLE

The tables MY_PROJECT_TBL, MY_APPLCTN_TBL, XLATTABLE, and MY_USER_TABLE are all accessed to retrieve information for related display fields. In `Add`, because the control display fields do not have any values and are new, values for these related display fields are not necessary.

Let us take a look at the panel display in the `Add` (figure 9.23) and `Update/Display` (figure 9.24) modes to better understand this concept.

All panel fields are new here and do not have any values. The Problem ID field is populated because it was entered using the search dialog box.

Navigation: Go →Problem Tracking →Tracking →Track Problems →Add

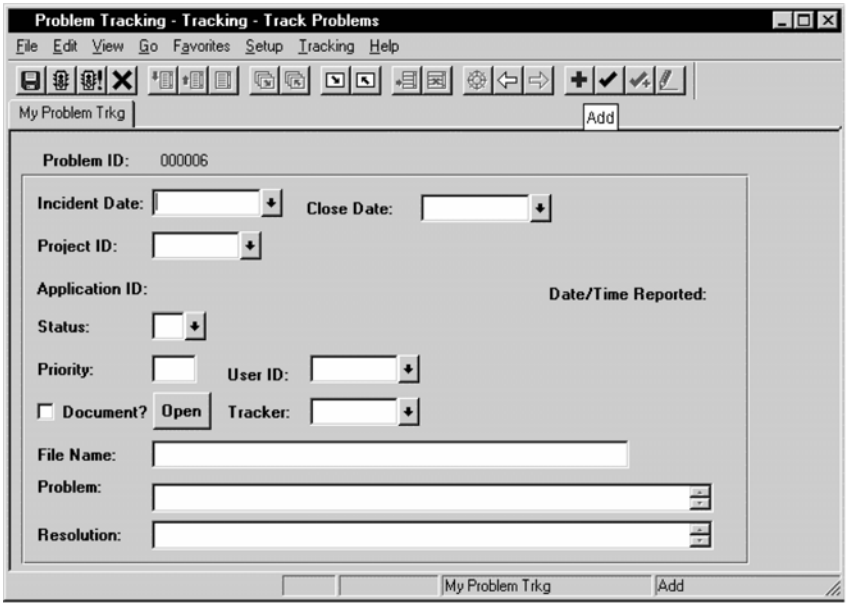


Figure 9.23 Application panel group in Add mode

In figure 9.24, notice some of the differences in the `Update/Display` mode. The data row that matches the Problem ID, `000001` is retrieved from the database. Data for all related display fields are also retrieved from the database. Project Description, Application Description, Status Description, User Name, and Tracker Name are some of the related display fields in the panel.

In `Add`, none of the related display fields have corresponding control display fields. However, if the values from search fields have corresponding related display fields, data for these fields are retrieved in `Add`.

The Application Processor uses search key values to retrieve information from MY_PROBLEM_TRKG table. So it uses MY_PROBLEM_ID field as the key field to retrieve data from this table. Information is retrieved from MY_PROJECT_TBL using MY_PROJECT_ID field from MY_PROBLEM_TRKG table. The Application Processor constructs SQL statements based on these parameters. In the database, table indexes are built using search fields defined in record definitions. The Application Processor can retrieve information from database tables and views more efficiently when key fields are available. Then, the Application Processor proceeds to the panel group display stage.

Navigation: Go →Problem Tracking →Tracking →Track Problems →Update/Display

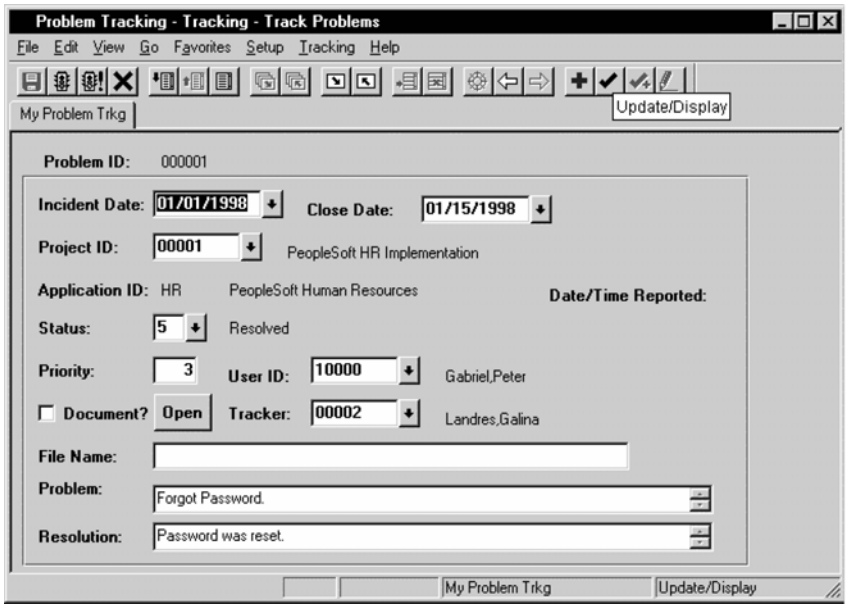


Figure 9.24 Application panel group in Update/Display mode

9.3 PANEL GROUP DISPLAY

After the Application Processor has collected data for panel group display, it performs row select and default processing on fields in the panel group. PeopleCode events are used to perform row select and default processing before the final data buffer is prepared for panel group display. The Application Processor then begins displaying panel fields by populating input fields, display fields, related display fields, and so forth. The Application Processor also has to determine the field labels and field layout before it can display the panel group. During the panel group display stage, the Application Processor performs row select and default processing (Iterative), displays panel group, and waits for user action.

9.3.1 RowSelect processing

In the `Add` mode, the Application Processor does not perform any row select processing. Row select processing discards rows from the buffer based on `RowSelect` PeopleCode event. In `Add`, no rows are selected or retrieved from the database.

In `Update/Display`, the Application Processor performs row select processing using the `RowSelect` PeopleCode event. After data retrieval, the Application Processor discards rows from the panel buffer. Row select processing is only performed on

panels with scroll bars. During panel group display, the Application Processor discards the rows from the buffer based on PeopleCode logic and will not display any discarded rows of data on the panel. The `RowSelect` PeopleCode event can be used to discard rows from the buffer and stop loading data into the panel.

NOTE Row select processing is only performed on panels with scroll bars. The `RowSelect` PeopleCode event is used to discard rows from scroll bar buffers before the final panel group display. Data discarded using row select processing is not available in the buffer.

9.3.2 Default processing (iterative)

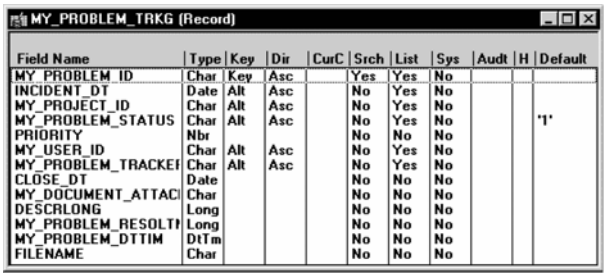
Default processing is performed using the `FieldDefault` and `FieldFormula` PeopleCode events. All fields that are not related display fields go through default processing which occurs both in `Add` and `Update/Display` modes. Default processing is an iterative process which constantly checks for blank fields in the panel group. A PeopleCode event may blank out a field that has field level default assigned to it. In this situation, the Application Processor performs default processing for the blank field. Any time the value of a field in the panel group changes, default processing is performed for other blank fields in the panel group.

Default processing is also processed using default values defined in record field properties. First, all defaults attached to record fields are processed, then `FieldDefault` and `FieldFormula` PeopleCode events are processed.

The Application Processor performs default processing on panel fields when the panel field is blank after data retrieval, the panel field is blanked out by a PeopleCode event, or the user blanks out a panel field.

Chapter 13, "PeopleCode and the Application Processor," covers default processing PeopleCode events as well as the sequence in which they are executed in greater depth. In this section, we want to illustrate how record field level defaults are processed. First, let's look at how default values appear on the record definition screen (figure 9.25).

Navigation: File →Open →Record →MY_PROBLEM_TRKG →View →Use Display


MY_PROBLEM_TRKG (Record)

Field Name	Type	Key	Dir	CurC	Srch	List	Sys	Audt	H	Default
MY_PROBLEM_ID	Char	Key	Asc		Yes	Yes	No			
INCIDENT_DT	Date	Alt	Asc		No	Yes	No			
MY_PROJECT_ID	Char	Alt	Asc		No	Yes	No			
MY_PROBLEM_STATUS	Char	Alt	Asc		No	Yes	No			'1'
PRIORITY	Nbr				No	No	No			
MY_USER_ID	Char	Alt	Asc		No	Yes	No			
MY_PROBLEM_TRACKEI	Char	Alt	Asc		No	Yes	No			
CLOSE_DT	Date				No	No	No			
MY_DOCUMENT_ATTACI	Char				No	No	No			
DESCRLONG	Long				No	No	No			
MY_PROBLEM_RESOLTI	Long				No	No	No			
MY_PROBLEM_DTTIM	DtTm				No	No	No			
FILENAME	Char				No	No	No			

Figure 9.25
Record definition—field defaults

If MY_PROBLEM_STATUS field is blank at the time the panel group is displayed, the default value defined in that record field is used to populate the field. Record field defaults are used in all modes as long as the field is blank. Let us look at the application panel in the `Add` mode to see how default values are used during display (figure 9.26).

Navigation: Go →Problem Tracking →Tracking →Track Problems

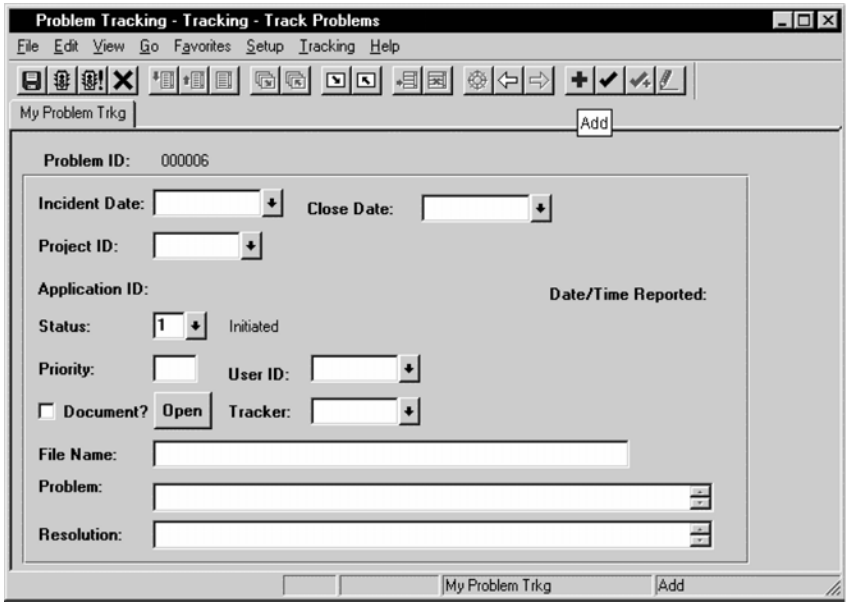


Figure 9.26 Record field default processing

Similarly, in modes other than `Add`, record field defaults are processed as well. Usually, however, when the application panel is processed in modes other than `Add`, field values from the database are retrieved and displayed. In instances where these fields are blank, record field defaults are used as display values.

9.3.3 Display panel group

During a panel group display, the Application Processor assembles fields in the order in which they are defined in the panel field layout. In this process, the Application Processor also displays field labels for panel fields. After performing all necessary items for display, the Application Processor then executes the `RowInit` PeopleCode event that may change the value of a panel field.

We also saw that default processing occurs when a panel field is blanked out. The `RowInit` PeopleCode event can potentially blank out a panel field, and the Application

Processor immediately performs default processing. Let us take a look at the panel field layout one more time to see the order in which fields are displayed (figure 9.27).

Navigation: File →Open →Panel →MY_PROBLEM_TRKG →Layout →Order

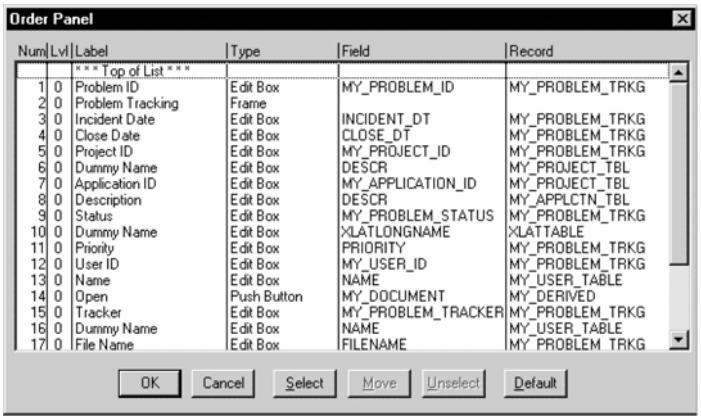


Figure 9.27
Panel field layout

Usually, search fields appear first in the order in which fields are displayed. Alternate search keys can be placed anywhere in the panel layout.

TIP Display control fields are always placed before their corresponding related display fields. In order for the Application Processor to successfully retrieve values for related display fields, values in the corresponding display control fields must be populated first.

After all fields are displayed, `RowInit` PeopleCode events are executed. At this point, all derived fields are also populated with PeopleCode events. After data retrieval and row select processing, derived fields are populated along with all other fields in the panel. Derived fields also go through default processing and display processing PeopleCode events.

TIP All default processing and display processing PeopleCode events from panel fields are processed during panel display. Even if a field is not present in the panel, PeopleCode events are triggered from that field. One exception to this rule is fields from derived records. PeopleCode is processed from derived fields only when they are present in the panel. For this reason, the same derived record can be shared across multiple panel sessions without interfering with each other.

In instances where multiple panels are present in a panel group, each panel is displayed in sequence within the panel group. The focus is on the panel the user chooses while accessing the panel group. The Application Processor now waits for user action.

9.4 DATA ENTRY OR INQUIRY

The Application Processor displays field values in the panel group and waits for user action. User actions can be adding, updating, or deleting data and saving them to the database. Some panel groups consist of inquiry panels that do not require any data entry. After the panel group is displayed, the user can perform the following list of actions in the data entry or inquiry stage:

- field modifications
- `rowInsert`
- `rowDelete`
- prompt processing
- command or push buttons
- pop-up menus
- save processing
- Cancel

9.4.1 Field modification

The user can modify any input field on the panel. When the user modifies the value on a panel field, the Application Processor performs field edit processing. The following processes are executed when a user modifies a panel field:

- internal PeopleTools edits
- field edit
- field change
- default processing

PeopleTools edits the value entered in the field with the field format defined for the field and it issues a message to the user. For example, when the user enters a non-numeric character into a field that can accept only numeric values, the Application Processor issues a message (figure 9.28).

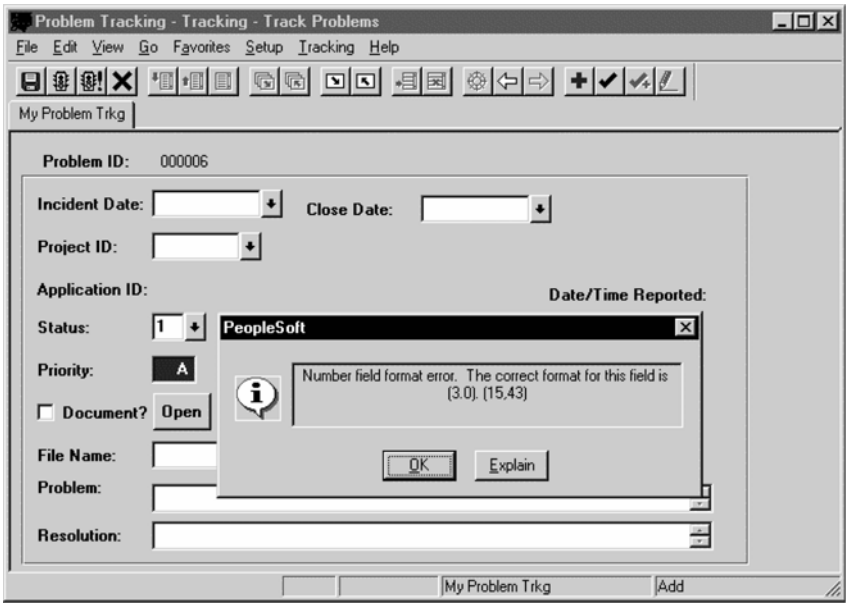


Figure 9.28 Field format edits

Similarly, the Application Processor performs other types of edits by comparing the values entered with record field attributes. Let us look at the record field attributes for the INCIDENT_DT field (figure 9.29).

Figure 9.29
Record Field Properties

The INCIDENT_DT field is edited and defined as a required field. The Reasonable Date checkbox is enabled as well. Let us see how the Application Processor behaves when the value entered into this field fails the edit (figure 9.30).

Because we entered a past date, the Application Processor issues a warning message indicating that the date is out of range. The Application Processor issues this

Figure 9.30 Reasonable Date checkbox

message when the date is over thirty days in the past or future and the reasonable date check is enabled. The following types of PeopleTools edits are available in PeopleSoft:

- Field Format
- Required Field
- Reasonable Date
- Prompt Table Edits
- Yes/No Table Edits
- Translate Table Edits

All these edits are performed using record field attributes. The Application Processor retrieves information on record field attributes from PeopleSoft catalog tables PSRECFIELD and PSDBFIELD.

`FieldEdit` and `FieldChange` PeopleCode events are also processed once the Application Processor successfully processes the PeopleTools edits. Chapter 13, "PeopleCode and the Application Processor," explains these PeopleCode events in a thorough fashion.

And, as we discussed earlier, any time a panel field is changed, the Application Processor performs default processing.

9.4.2 RowInsert

We use the `RowInsert` function on panels that have scroll bars. We can either press the F7 key or choose the row insert icon from the application menu. When a row is inserted into a scroll bar, the following events take place:

- A new row is inserted in the current scroll area and all dependent scroll areas.
- The `RowInsert` PeopleCode event is executed.
- Default processing occurs for fields on the new scroll row.
- The `RowInit` PeopleCode events are processed for fields on the new scroll row.

The Application Processor automatically inserts a row into the current scroll bar and child scroll bars. After the new rows are created, `RowInsert` event is executed from all fields in the scroll bars. As previously discussed, the Application Processor performs iterative default processing on all fields in the scroll bar. Default values are used only when the fields are blank or zero for numeric fields. Finally, `RowInit` is triggered from all the fields on the scroll bar.

When the primary record on the scroll bar is effective-dated, the Application Processor copies values of fields from the current row to the new row. (`Current` does not imply the current effective-dated row. It means the row from which the user chooses to perform the row insert.) This feature is built internally into PeopleTools to accommodate maintenance of history data in PeopleSoft. This enables the user to override only the fields that are different on the new row.

TIP When the primary record on the scroll bar is effective-dated during `RowInsert`, the Application Processor automatically copies data into the new row.

TIP We can disable the `RowInsert` function on a scroll bar by editing scroll bar properties in panels.

9.4.3 RowDelete

We can use the `RowDelete` function on panels that have scroll bars. This function is used to delete rows from the scroll bar. The `RowDelete` function can be performed by either pressing the F8 Key or by choosing the row delete icon from the application menu. When the `RowDelete` function is chosen, the `RowDelete` PeopleCode event is executed from fields on the scroll area; a warning message is issued to verify the `RowDelete`; and rows are deleted from the current scroll area and all child scroll areas.

The Application Processor executes the `RowDelete` PeopleCode event from fields on the scroll area. This PeopleCode event can be used to verify whether the user can delete this row, and an error or warning message can be issued to either prevent or warn the user.

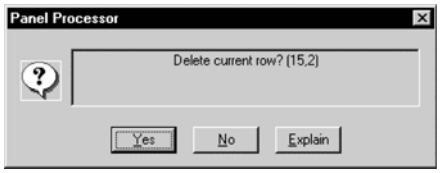


Figure 9.31 PeopleTools message during RowDelete

By default, the Application Processor automatically issues the message in figure 9.31 when a row is deleted from the scroll bar.

If the user chooses to proceed with `RowDelete`, the Application Processor deletes rows from the current scroll area as well as all dependent scroll areas. It is important to note that the rows are not physically deleted from the database until the panel is saved. The deleted row is still available for access until save time.

TIP Scroll bar functions such as `TotalRowCount` and `ActiveRowCount` treat rows marked as deleted in scroll bars differently. The `TotalRowCount` function adds all rows including deleted rows on the scroll bar. The `ActiveRowCount` function adds all rows except the deleted rows.

TIP We can disable the `RowDelete` function on a scroll bar by editing scroll bar properties in panels.

9.4.4 Prompt processing

A prompt is a list of valid values for a Panel field. Prompt processing is activated under record field properties. When the user presses the F4 Key or clicks on the drop-down list, the Application Processor provides a list of valid values for the panel field. Prompt processing is performed using keys from the prompt table.

For the prompt to be processed correctly, key fields from the prompt table have to be present on the panel group. Let's take a look at how prompt processing works in the MY_PROBLEM_TRKG panel (figure 9.32).

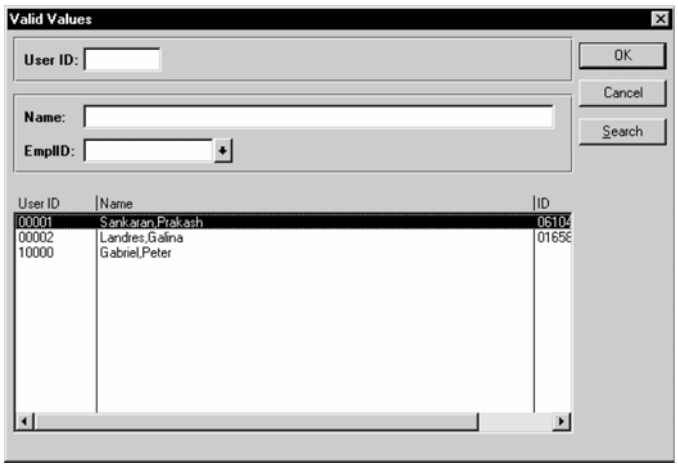


Figure 9.32 Prompt processing

Here, we activated the prompt on the User ID field, and the Application Processor produced a list of valid users from MY_USER_TABLE. MY_USER_TABLE is the prompt record for the MY_USER_ID field on the MY_PROBLEM_TRKG record.

Observe how the Application Processor displays all list box fields from the Prompt record in the prompt list. (To learn more about prompt records and prompt processing refer to chapter 6.)

9.4.5 Command or push buttons

When command or push buttons are pressed, the Application Processor executes the `FieldChange` PeopleCode event for the panel field. Push buttons can be defined as a command button, a process, or a secondary panel.

When the push button is defined as a command button, the Application Processor executes `FieldChange` PeopleCode event from the record field. When the push button is defined as a process, the batch process attached to the push button definition is run. When the push button is defined as a secondary panel, the secondary panel attached to the push button definition is brought up. Consider the example from our application panel shown in figure 9.33.

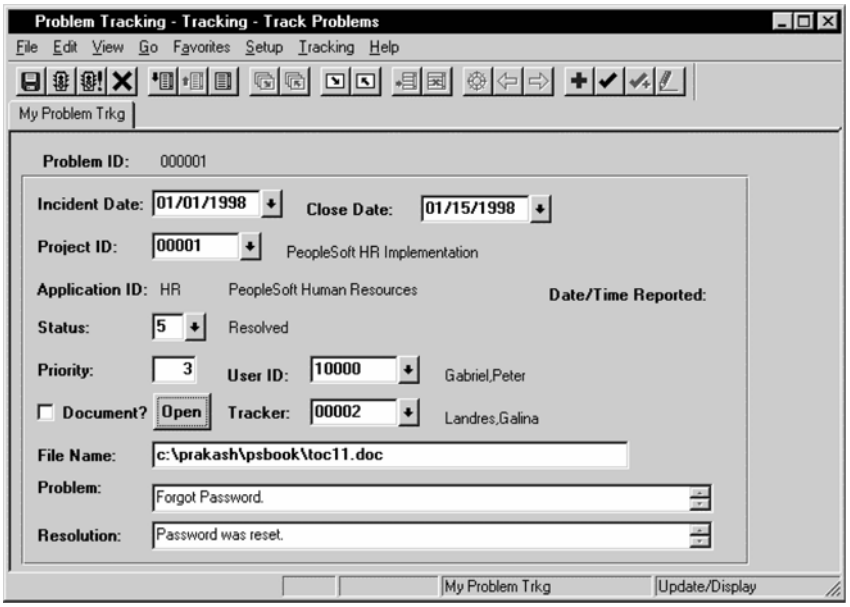


Figure 9.33 Push buttons on application panels

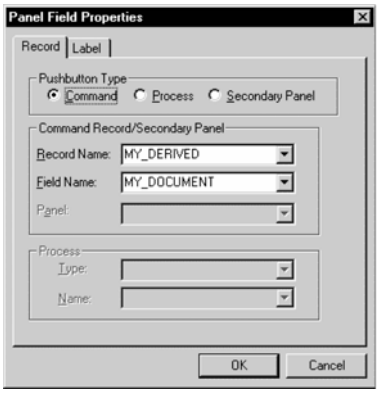


Figure 9.34 Push Button Properties

When the Open push button is activated, the `FieldChange` PeopleCode event is processed. Let's look at the definition for the push button and the `FieldChange` PeopleCode event attached to it (figures 9.34 and 9.35).

The push button is defined as a command button, and the field MY_DOCUMENT is from MY_DERIVED record. The `FieldChange` PeopleCode event from this record field opens a Microsoft Word document with the filename as the parameter. The Application Processor automatically executes the `FieldChange` PeopleCode event to bring up the Word document. (To learn more about push buttons, refer to chapter 6.)

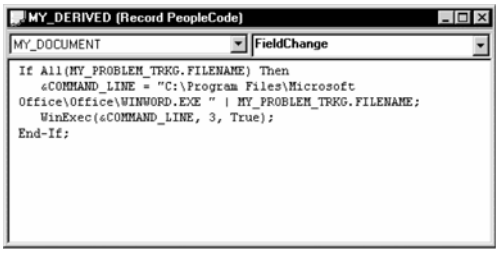


Figure 9.35 FieldChange PeopleCode for the push button

9.4.6 Pop-up menus

We can attach pop-up menus to a panel field to bring up context-sensitive information. Two types of pop-up menus can be brought up from a panel field: the standard pop-up menu and the developer-defined pop-up menu.

The Application Processor executes the `PrePopup` PeopleCode event when the user activates a developer-defined pop-up menu. Pop-up menus are activated either by right-clicking on a panel field or using the SHIFT-F10 Key.

In figure 9.36, we can see how a standard pop-up menu is displayed by the Application Processor. We can use any field in our application panel for this purpose.

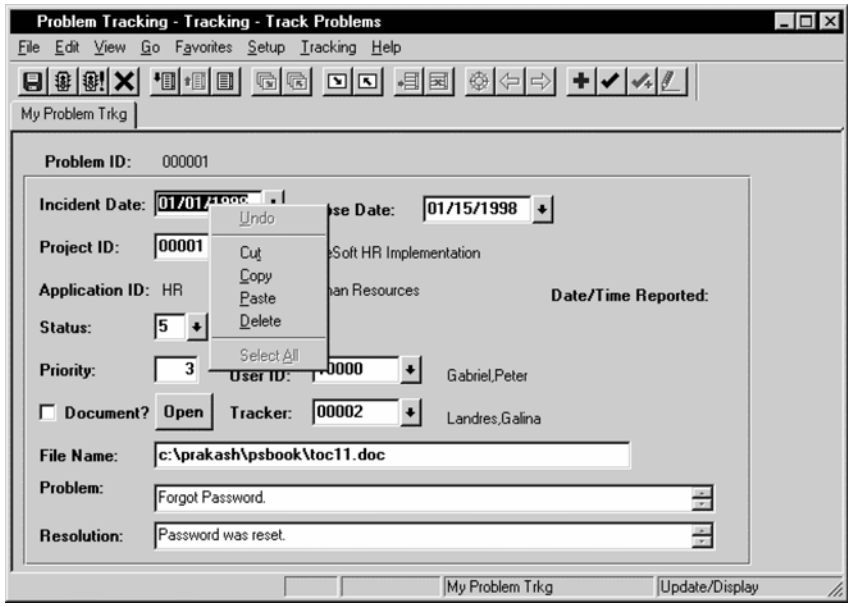


Figure 9.36 Standard pop-up menu for a panel field

9.4.7 Save processing

When the user chooses to save the panel group by pressing on the Save button, the Application Processor begins the save processing stage. We start with any one of the following actions:.

- choose File →Save from the application menu
- press ENTER using the keyboard
- choose the Save button

When the user performs the following actions, the Application Processor issues a message to the user, as illustrated in figure 9.37.

- choose Next in List
- choose Previous in List
- choose List
- choose another panel group
- choose another application menu without starting a new window
- close the application panel window

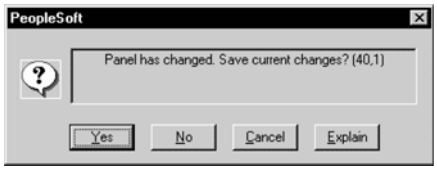


Figure 9.37 Save message

The message in figure 9.37 is just a reminder to the user that data has been changed in the application panel and the changes are not saved yet. When the user chooses "Yes" to proceed with the save, the save processing stage begins. When the user chooses "No," the Application Processor transfers the focus back to the application panel. When the user chooses "Cancel," the Application Processor cancels the whole panel group, and all changes are lost.

Let's go back to our panel group example where the user has made some changes and now chooses to save the panel group. During the save processing stage, the Application Processor triggers the following events:

- executes the `SaveEdit` PeopleCode event
- executes the `SavePreChg` PeopleCode event
- executes the `WorkFlow` PeopleCode event
- updates the database with changes made in the panel group
- executes `SavePostChg` PeopleCode event
- saves the data to the database

Execute SaveEdit PeopleCode event

The Application Processor executes `SaveEdit` PeopleCode events from all records in the panel group except records which contain related display fields. Let us look at the panel field layout for MY_PROBLEM_TRKG panel to identify related display and non-related display fields (figure 9.38).

Navigation: Layout →Order (MY_PROBLEM_TRKG panel is open)

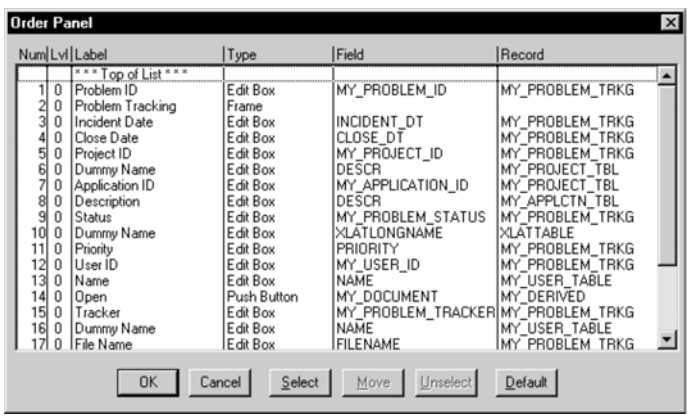


Figure 9.38 Panel fields layout

In figure 9.38, we notice that fields exist from more than one record definition. MY_PROJECT_TBL, MY_APPLCTN_TBL, XLATTABLE, and MY_USER_TABLE are records that contain the related display fields in the panel. The only other record that does not contain related display fields is the MY_PROBLEM_TRKG record.

Figure 9.39 Error message

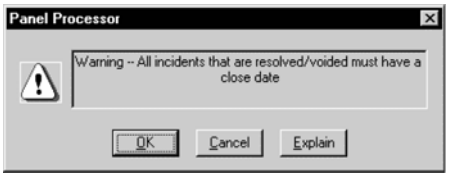


Figure 9.40 Warning message

Only the `SaveEdit` PeopleCode events from MY_PROBLEM_TRKG record are going to be executed by the Application Processor. `SaveEdit` PeopleCode events are used to validate user data entry before saving them to the database. Error and warning messages can be added to `SaveEdit` PeopleCode events to either prevent or warn the user after validation. Let's go back to our example and look at the difference between an error message and a warning message (figures 9.39 and 9.40).

An error message prevents the user from saving the panel group. The Application Processor then returns the focus to the application panel to allow the user to rectify the error. A warning message allows the user to override the validation and save the panel group. The user can choose the OK button to override the message and save the panel group. On the other hand, the user can choose the Cancel button to get back to the panel group and correct the problem causing the Application Processor to issue the warning message. `SaveEdit` events are processed in the order in which fields are laid out in the record definition. (Part 3 will explain more about the usage of `SaveEdit` PeopleCode events.)

TIP `SaveEdit` events are processed in the order in which fields are laid out in the record definition. `SaveEdit` events in records that are present in panels with scroll bars are processed starting from level zero.

Execute SavePreChg PeopleCode event

The Application Processor executes all `SavePreChg` PeopleCode events from records in the application panel with the exception of records which contain related display fields. `SavePreChg` PeopleCode events are used to process data after user input—before the data are saved to the database. A perfect use for `SavePreChg` PeopleCode event would be to populate certain fields which are not input fields on the panel, based on other fields which *are* input fields. The Application Processor executes the `SavePreChg` PeopleCode events in the order in which the fields are laid out in the record definition. The Application Processor errors out with a runtime

error when an error or warning message is used in SavePreChg PeopleCode events. (Part 3 explains SavePreChg PeopleCode events in greater detail.)

TIP An error or warning message can be used only in SaveEdit PeopleCode event during save processing.

Execute WorkFlow PeopleCode event

After all panel fields are populated for saving, the Application Processor then executes WorkFlow PeopleCode events. WorkFlow PeopleCode events are used to trigger business events which either update a work list or execute a message agent. WorkFlow PeopleCode events are executed in the order in which fields are laid out in the record definition. (Again see part 3 for more discussion on WorkFlow PeopleCode events.)

Update the database with changes

The Application Processor then builds SQL scripts to update database tables based on changes made to the panel. Some database platforms allow updates to SQL views that in turn update the underlying database tables from which the views were built. It is important to note that the data are not saved in the database yet. In other words, the Application Processor does not perform any commits to the database in this step. In the Add mode, the Application Processor determines that it must perform an SQL insert. In other modes, the Application Processor determines that an update is necessary. In panel groups with scroll bars, the Application Processor performs an SQL insert for rows added into the scroll bar and SQL deletes for rows deleted from the scroll bar.

TIP Rows that are deleted from the scroll bar are available in the panel buffer until the Application Processor issues an SQL commit to the database.

Execute SavePostChg PeopleCode event

The Application Processor executes all SavePostChg PeopleCode events from records in the panel group. SavePostChg PeopleCode events are not executed from records which contain related display fields in the panel group. SavePostChg PeopleCode events can be used to perform updates to other related application tables based on information entered in the panel group. As with the preceeding PeopleSoft events, SavePostChg PeopleCode events are executed in the order in which fields are laid out in the record definition. (See part 3 for more information about SavePostChg PeopleCode events.)

Saves the data to the database

The Application Processor is now ready to save the changes permanently to the database. The Application Processor issues an SQL commit to the database to accomplish this. At this time, all rows deleted from scroll bars are no longer available in the panel buffer. The Application Processor, after successfully saving data, returns the focus back to the panel group in a static state.

NOTE The Application Processor can error out with a runtime error after `SaveEdit` PeopleCode events are executed. Runtime errors can occur as a result of errors in SQLExec statements, database index constraints, data storage parameters, network connection loss, and so forth.

Navigation: File →Object Properties (Panel Group is open)

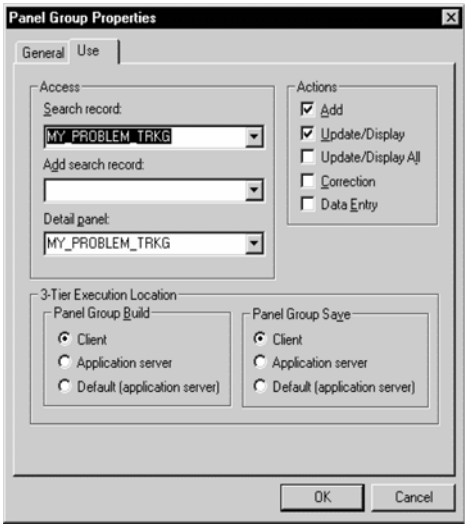


Figure 9.41 Panel Group build and save parameters

The panel group definition contains two parameters which help the Application Processor determine both where to build the panel group and where to perform save processing (figure 9.41).

Figure 9.41 illustrates the two parameters in the panel group definition which the Application Processor uses to build and save the panel group. When Client is chosen either for Panel Group Build or Panel Group Save, the Application Processor is going to process both stages on the client workstation. When Application Server is chosen, the Application Processor is going to process both stages in the machine which hosts the Application Server. The Application Server is physically closer to where data reside than the client workstation.

During save processing, all PeopleCode events except for the `SaveEdit` PeopleCode event can be executed on the Application Server.

In a three-tier installation which runs using a Tuxedo Application Server configuration, it is prudent to process both these stages close to the database server. (For a detailed description on two-tier and three-tier installations, refer to chapter 1.)

9.4.8 Cancel

Finally, the user can choose to cancel all changes made to the fields in the panel group by simply choosing the Cancel button from the application menu. The user can also choose "No" to the message illustrated in figure 9.37 to cancel all changes.

KEY POINTS

1. The Application Processor organizes numerous individual steps from the time the user accesses an application panel to the time the user saves the changes to the database.
2. The Application Processor stages can be divided into search processing, data retrieval, panel group display, and data entry/inquiry stages.
3. Search records that are attached to panel groups are used during the search processing stage.
4. The Application Processor executes a number of PeopleCode events during all stages.
5. The Application Processor retrieves less data from the database in `Add` and `Data Entry` modes when compared to all other modes.
6. Panel field definitions and layouts are used to retrieve and display fields in the panel group.
7. The Application Processor performs default processing, which is an iterative process performed during all stages of the panel group session.
8. During the data entry/inquiry stage, the Application Processor performs validations, executes field modification PeopleCode events and row modification PeopleCode events, and performs save processing.
9. Save processing starts when the user chooses to save the panel group session.
10. In a three-tier installation, the Application Processor looks at the `Panel Build` parameter and the `Panel Save` parameter in the panel group definition in order to determine where to build and save the panel group.

CHAPTER 10

Application Designer—PeopleSoft 8

PeopleSoft 8 contains exciting new features that enhance the types of end users who can access a PeopleSoft application. One such feature is the Internet Client. Almost all the new features that are available in the Application Designer tool are focused to service the Internet Client. While the Application Designer tool in this new release is similar to that in release 7.5, some enhancements are available in PeopleSoft 8. Let's have a look at what's new in PeopleSoft 8.

10.1 DEVELOPMENT OBJECTS

A number of new objects can be developed using the Application Designer tool in PeopleSoft 8. Objects such as the Application Engine Program and Approval Rule Sets, developed using a tool menu outside of the Application Designer tool in release 7, are now integrated with the Application Designer tool. The following is a list of objects that can be developed using the Application Designer. New objects in PeopleSoft 8 are mentioned in *Italics*:

- Activity
- Application Engine Program
- Approval Rule Set
- *Business Component*
- *Business Interlink*
- *File Layout*
- *HTML Definition*
- *Image*
- *Message Definition*
- *Message Channel*
- *Message Node*
- *SQL*
- *Style Sheet*

10.1.1 Application Engine program

Application Engine is a PeopleTools object that provides an alternative to using COBOL and SQR for batch applications. Application Engine is now integrated within the Application Designer tool and can be upgraded between databases. To learn more about new features in Application Engine in release 8, see chapter 45 in this book.

10.1.2 Business components

Business components are individual business transactions. Business transactions such as Purchase Orders, New Hires, and Journal Entries can be encapsulated into a business component. Business Components are single instances of a panel group session and can be invoked from external applications as well as from PeopleCode. The whole idea behind business component definitions is to provide external access to PeopleSoft applications. Business Components expand the possibilities of the user types who can access the system.

Business component definitions are divided into three parts:

- Search Keys are from the search records defined in panel groups.
- Methods are processes that take place when the business component is invoked.
- Properties refers to a single instance of data in the panel group. For example, data stored in a field in the panel group constitutes a property.

Figure 10.1 illustrates a business component definition in PeopleSoft 8.

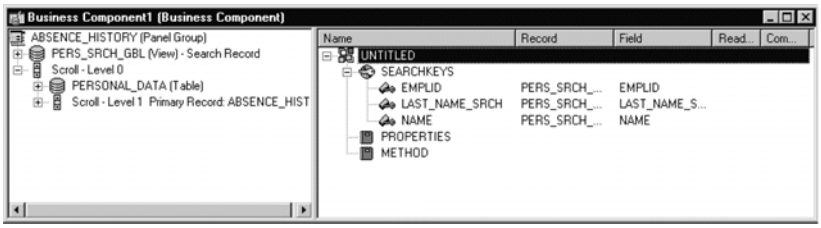


Figure 10.1 Business component definition

The business component API is used by external applications to access the PeopleSoft business component and supports the following environments:

- Microsoft COM (Visual Basic)
- PeopleCode

Figure 10.2 is an illustration from PeopleBooks 8 on business component architecture.

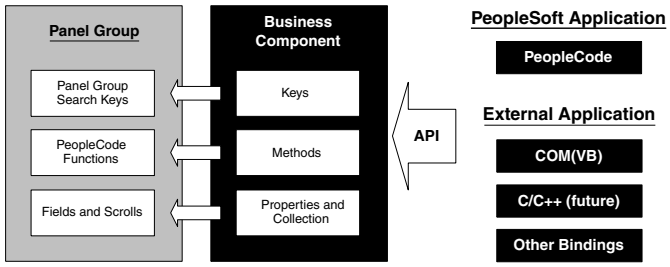


Figure 10.2 Business Component architecture

10.1.3 Business interlink

A business interlink allows you to integrate your software system with PeopleSoft. Business Interlink is a gateway to your software system from PeopleSoft. You can invoke a business interlink from your PeopleSoft application. The external software system is integrated by using a business interlink plug-in which provides a framework to that system. The business interlink plug-in allows you to invoke the external software system using PeopleCode.

Basically, the business interlink plug-in accesses the PeopleSoft Application Server, which then initiates a business interlink object which resides in the PeopleSoft application. The business interlink plug-in can be located in the same machine as the

PeopleSoft Application Server to enable access to software systems within your company. On the other hand, the business interlink plug-in can be located outside the firewall on a web server to access third-party software applications from within PeopleSoft.

The business interlink plug-in exposes the external software system to PeopleSoft as sets of input/output transactions or data classes for query and updates. For example, your PeopleSoft Payroll application can transact with your Oracle General Ledger system to feed in journal entries using data classes which represent the data structure in the external database. Similarly, your PeopleSoft Benefits application can feed enrollment transactions to vendors which manage employee benefits. Figure 10.3 illustrates the business interlink architecture with the plug-in located on a web server.

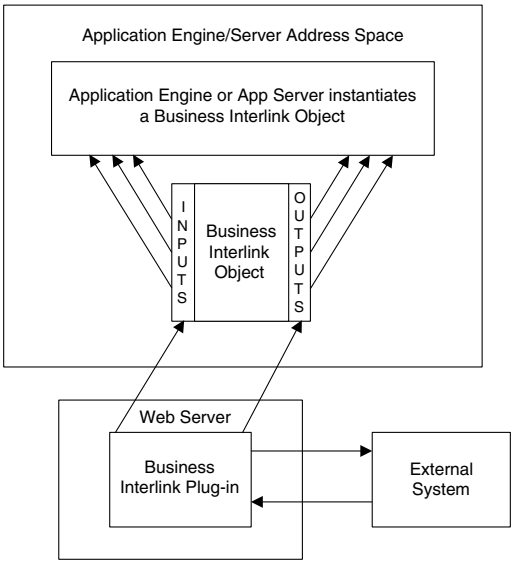


Figure 10.3
Business interlink architecture—plug-in located on the web server

10.1.4 File Layout

File Layout maps fields in a file. It basically describes the location of columns in a file. Using the file layout definition, PeopleCode can either read from or write into a file. Figure 10.4 illustrates a simple file layout definition in PeopleSoft 8.

One or more record definitions may be used in building a file layout. Fields from the record definitions are automatically expanded into the file layout. The record fields are only used as templates for columns in the file. No correlation exists between record fields and the file layout. Changes to record field attributes will not be reflected in the file layout.

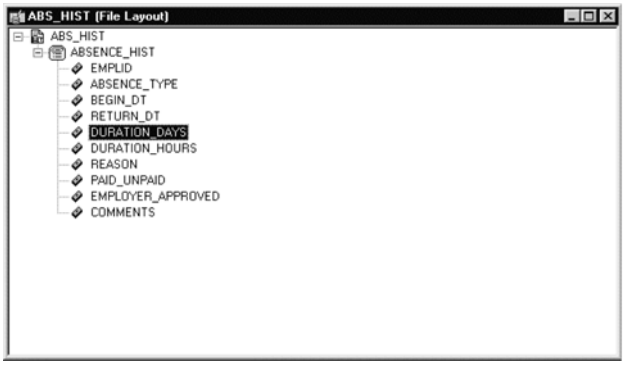


Figure 10.4
File layout definition

10.1.5 HTML definitions

HTML area control can be inserted into any PeopleSoft panel. It can be viewed only from an Internet Client. The Internet Client automatically converts the panel into HTML tags. The HTML area control can be inserted into any level in the panel as a rectangular box which can be reshaped and resized.

An HTML area can be populated using static texts and images or dynamic record fields. The HTML area control is different from other Internet Client controls. While the Internet Client translates other controls such as check boxes, prompt lists, and so forth. into HTML tags in the HTML area control, the developer writes HTML code used as is during runtime, but HTML tags <body>, <frame>, <frameset>, <form>, <head>, <html>, <meta>, and <title> are not supported in the HTML area control.

10.1.6 Image definition

Images are converted into image definitions and stored in a repository. In PeopleSoft 7.5, images were associated to panels and defined as panel fields. In PeopleSoft 8, image files are converted into image definitions and stored within PeopleTools as objects.

The following image types can be stored as image definitions:.

- BMP—Bitmap
- DIB—Device independent Bitmap
- JPEG
- GIF—Only for Internet Client

10.1.7 Message definition

Message definition is used in the Application Messaging system in PeopleSoft. Message definitions store information on how a single message is passed using the Application Messaging system. Messages are objects formatted in XML. In PeopleSoft 8, messages are used as a single unit of transaction in the Publish and Subscribe process.

10.1.8 Message channel definition

Message channels group individual message definitions and organize their transmission. They route messages between nodes across your network and they define how each single message definition in the group should process.

10.1.9 Message node definition

Message nodes are physical systems connected to the messaging network. Subscribers subscribe to a Message node, which can be an Application Server or a database, and the Application Server or the database publishes a message, which is transmitted to the subscriber using message channel definitions. Message channel definitions include information on how to route messages from the publishing node to the subscribing node. Figure 10.5 illustrates how messages are published and subscribed using these three new object types.

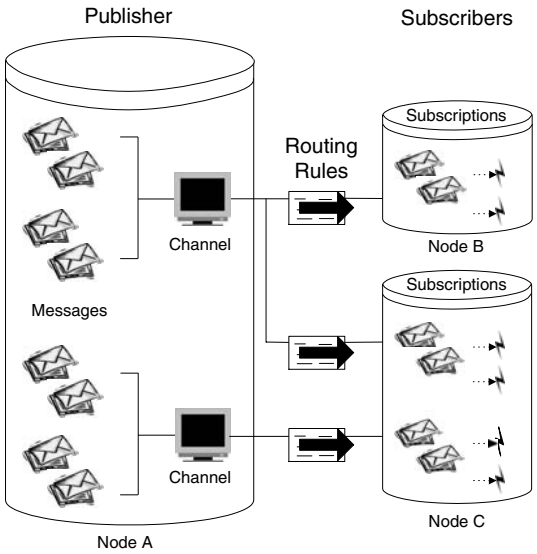


Figure 10.5
Publishing and subscribing messages—application messaging

10.1.10 SQL definition

We can store SQL statements in PeopleTools using the Application Designer. An SQL definition can be a section of an entire SQL statement that you want to reuse or multiple SQL statements. The SQL definition can be accessed using PeopleCode SQLClass. Unlike `SQLEXEC` functions, an SQLClass allows selection of all rows using an SQL statement. You can also create an SQL definition using PeopleCode, then store it in PeopleTools. The SQL definition contains the SQL statement, the database type, and an effective date.

PeopleCode functions associated with the SQL definitions are `CreateSQL`, `DeleteSQL`, `FetchSQL`, and `GetSQL`.

10.1.11 Style sheet

Web development needs style attributes beyond foreground, background, and fonts. In PeopleSoft 8, a new development object, the Style Sheet, can be created using the Application Designer. A style sheet is a collection of styles that can be used on a webpage. A style sheet can be displayed only on the Internet Client and is attached to the properties of a PeopleSoft Panel definition. PSSTYLEDEF is the default style sheet that is delivered in PeopleSoft 8. This default style sheet already contains classes that define style attributes in a webpage.

10.2 OTHER FEATURES

Other new features are available in PeopleSoft 8 under the Application Designer tool. They can be categorized into:

- general environment
- field definitions
- record definitions
- panel definitions
- panel groups

10.2.1 General environment

- The PeopleCode Debugger can be accessed using the new Debug menu from the Application Designer tool.
- Language translations can be performed from the Tools menu using the Translate menu item.
- A new Internet Options menu item is available under the View menu from the Application Designer tool.
- Additional password controls are available in PeopleSoft 8. Minimum length, password expiry, and special character requirements are some features which control user passwords.

10.2.2 Field definitions

- Multiple labels can be attached to field definitions in PeopleSoft 8. A default label can be indicated in the field definition. Any one of the labels can be used as the record field label or panel field label.
- Numeric fields can accommodate thirty-one decimals in PeopleSoft 8.

10.2.3 Record definitions

- You can sort fields within a record definition by clicking on the column headings.

- Subrecords attached to record definitions can be expanded and viewed.
- A new record type called the Temporary Table is now available in PeopleSoft 8.

10.2.4 Panel definitions

- Under the panel field layout screen, scroll levels are now displayed. Control Display and Related Display fields can also be seen in the panel field layout screen.
- Required fields appear with an '*' at runtime in panels.
- Multiple grids can be included in the same panel in PeopleSoft 8.
- Grids can be hidden or shown using a PeopleCode object.

10.2.5 Panel group definitions

- Panel groups are displayed in a tabbed interface in the object workspace.
- Two new PeopleCode events associated with panel groups—the `PreBuild` and `PostBuild` PeopleCode events—are available in PeopleSoft 8.
- A new option is available to disable saving the panel group in PeopleSoft 8.
- Internet Client attributes are available in the panel group definition to support the new Internet Client.

PART 3

PeopleCode: an in-depth look

One of the most powerful PeopleTools is the proprietary language called PeopleCode, which is used in conjunction with the Application Processor to control an application's behavior. The fundamental elements in the PeopleCode language are similar to those found in other programming languages such as SQR, Visual Basic, and PowerBuilder. In addition, PeopleCode has an extensive set of functions and syntax conventions designed specifically for PeopleSoft object types such as scroll bars. Some readers may be familiar with event-driven languages like Visual Basic where code is attached to action events such as mouse-clicking or tabbing from one field to another. As we discovered in part 2, the Application Processor differs in that it generally maintains events at the record and field levels. For example, when a panel is initially populated, the fields may be initialized through PeopleCode designated in the `RowInit` event. When an attempt is made to save the information on a panel, the Application Processor calls any `SaveEdit` PeopleCode that may exist. Since all the events and execution of PeopleCode are regulated by the Application Processor, the developer is free to concentrate on the functional aspects of the program. PeopleCode language elements, syntax, variables, and field references will be covered along with further explanation of event processing and program execution flow. To demonstrate a few advanced features of PeopleCode, we continue to examine and enhance our Problem Tracking application as well as present other pertinent examples. One of the most difficult PeopleCode techniques to master is working with scroll bars. To emphasize scroll handling, we create a slick little application which links employees to operator classes/locations. We also cover topics such as function libraries, error handling, debugging, and embedded SQL. The section concludes with an overview of new PeopleCode features in release 8.0.

CHAPTER 11

Introduction to PeopleCode

Using Application Designer, we can build records and panel functionality that are quite eclectic. The PeopleCode language offers a wide range of features including the ability to manipulate variables, panels, and scrollbars. PeopleCode is accessed from the Application Designer, which contains the PeopleCode editor.

11.1 WHAT IS PEOPLECODE?

PeopleCode is a PeopleSoft proprietary programming language used with PeopleTools applications. It is an interpreted scripting language and works as part of Application Designer in conjunction with the Application Processor. PeopleCode programs are linked to applications through record fields or menu items.

Let's assume that an application is distributed to various clients. Some of these clients would like to fully utilize the application, and others require limited functionality or fewer fields on panels than what would normally be displayed. One option would be to develop two panels: one panel can contain all the delivered fields; and the other panel would contain a limited number of fields. Maintaining two panels would not be difficult, but if a multitude of panels existed requiring similar designs, our panel maintenance would increase. This would also impact future upgrades, menu maintenance, and security—the potential for a tumultuous undertaking!

If the objective is to be more efficient within certain standards and utilize the Application Designer environment to a fuller extent, PeopleCode can be used to hide fields on the same panel. This approach saves the replication of panels. We can identify the user in an efficient manner so that the PeopleCode can operate on the correct fields by using operator classes, panel groups, menus, or language code. Once this information is known to the program, the manipulation of the PeopleTools environment can be done by hiding fields on panels and setting varying default values and edit controls using PeopleCode. Although several panels may not be required, the trade-off is additional PeopleCode that can, however, be localized to specific records and events.

As we've said, PeopleCode is a programming language which enables developers to extend the functionality of PeopleTools applications. An application can be refined or made more efficient through the use of PeopleCode. These refinements can take many forms including:

- hiding and un-hiding values on a panel
- defaulting values based on some common identifier
- editing values entered on a panel
- submitting jobs to the Process Scheduler
- enabling/disabling menu items
- calling functions from various events. (possibly leading to code efficiency and reusability)

Familiarity with structured programming languages such as C++, Visual Basic, SQR, and knowledge of relational data base concepts and SQL provide a foundation for a better understanding of PeopleCode. PeopleCode's syntax is similar to some structured languages but the events, rules, and general behavior of PeopleCode are directly linked to the Application Designer and PeopleTools environment. Panels, menus, records, and fields are all interwoven into the application of the PeopleCode language.

PeopleCode programs are joined with record fields and triggered by events such as record initialization, tabbing out of a field, saving a record, or other events generated by the Application Processor. Think of the Application Processor as a traffic officer who coordinates the efforts between the end-user, panels, PeopleCode, and other PeopleTools objects. Variables appearing on panels and work fields may be initialized or manipulated using PeopleCode. Other features include message functions used to communicate with the end user as well as features which handle panel scroll bars and their data contents.

11.2 PEOPLECODE EVENTS

In PeopleTools 7 every PeopleCode program is associated with both objects and events. PeopleCode events occur when specific actions take place. These actions can take place when a panel is initially displayed or when the value of a field on a panel is changed. Two types of PeopleCode exist within Application Designer. The first type is tied to events and occurs within record fields. The other type of PeopleCode is tied to menu items. These are referred to as record PeopleCode and menu PeopleCode, respectively. Collectively, we can refer to both types as PeopleCode event sets.

11.2.1 Record PeopleCode events

PeopleTools terminology defines a record as an SQL table, a view, or a derived/work record. Fields are a subset of records and they contain over ninety percent of PeopleCode. Each field can have an event, and each event can trigger PeopleCode. Some code is more efficient when contained in specific events; however, the same code placed into other events can generate runtime errors.

The illustration in figure 11.1 demonstrates the relationship between records, fields, and their corresponding PeopleCode events.

As figure 11.1 illustrates, records are comprised of fields. A record can contain one to many fields. Each field contained in a record has events and may have PeopleCode linked to one or more events. Events enable the developer to interact with the PeopleTools environment at key points during a working session. These events can include PeopleCode to initialize values and perform specific actions when a row is selected or when data are changed on a panel. PeopleCode can also be used to interact with events during panel save operations. We can categorize events into those that occur before data are retrieved from a record, during the maintenance phase when data are changed, and at the time data are saved to a record.

Record PeopleCode events include:

- `FieldDefault`
- `FieldEdit`
- `FieldChange`
- `FieldFormula`
- `RowInit`

- RowSelect
- RowInsert
- RowDelete
- PrePopup
- SaveEdit
- SavePreChg
- WorkFlow
- SavePostChg
- SearchInit
- SearchSave

These events occur during actions related to fields. PeopleCode can be attached to these events and can range from simple to complex. Hypothetically, PeopleCode can be inserted into each of the events for a given field. Not every PeopleCode event plays a role, so it may not be relevant or necessary to insert PeopleCode into each

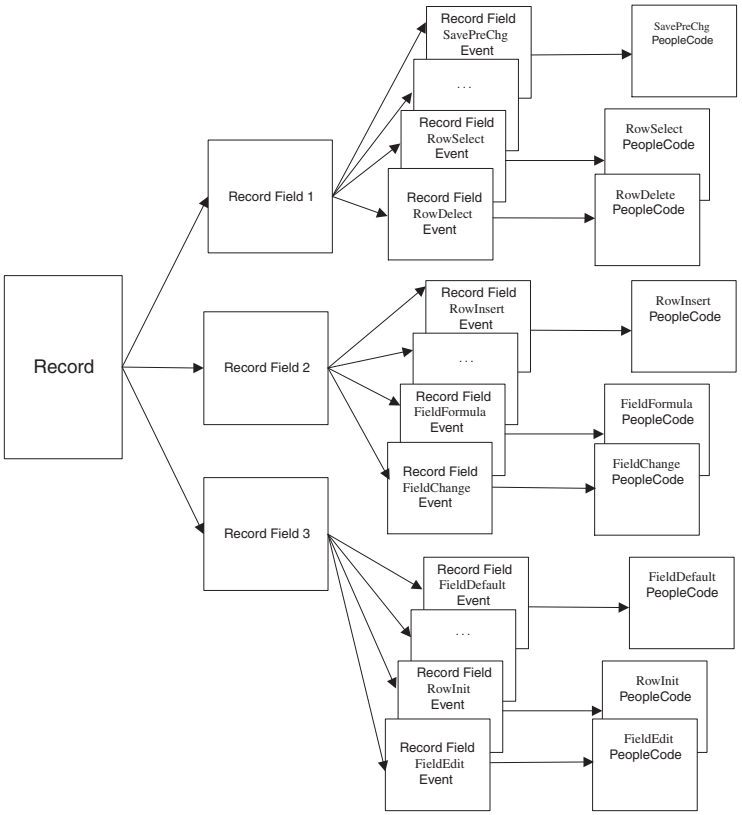


Figure 11.1 Relationship between records, fields, and PeopleCode events

event. Some PeopleCode events are triggered based on factors such as scroll bars and pop-up menus that appear on a panel. For example, in the Problem Tracking application, the panel MY_PROBLEM_TRKG does not contain a scroll bar. If we insert `RowInsert` or `RowDelete` PeopleCode, the programs in the events are never executed because `RowInsert` and `RowDelete` are related to scroll bar actions.

11.2.2 Menu PeopleCode events

Only one Menu PeopleCode event exists:

- `ItemSelected`

Menu PeopleCode is linked to the selection of a menu item from a standard or pop-up menu item.

11.3 USING APPLICATION DESIGNER TO DEVELOP PEOPLECODE

The Application Designer enables the insertion or editing of PeopleCode from points within the project workspace. PeopleCode can be accessed:

- from a record field definition
- by double-clicking on the lightening bolt "⚡" in the project workspace
- from a menu item
- within the panel definition

PeopleCode is commonly added and modified from the Application Designer through record field definitions. To add record PeopleCode to the Problem Tracking application, the first step required is to open the project definition from the Application Designer.

The Problem Tracking application, when viewed from the project workspace, contains objects such as menus, panel groups, records, and fields (figure 11.2). When we view projects and more specifically objects such as records, we see they contain a "+" on the left side. Clicking on the "+" for records displays all the records contained in the project. A subsequent click to a record such as MY_PROBLEM_TRKG displays the fields contained in the record. Figure 11.3 ❶, illustrates the record fields for MY_PROBLEM_TRKG record in MY_PROJECT.

Navigation: File →Open →Project →My Project

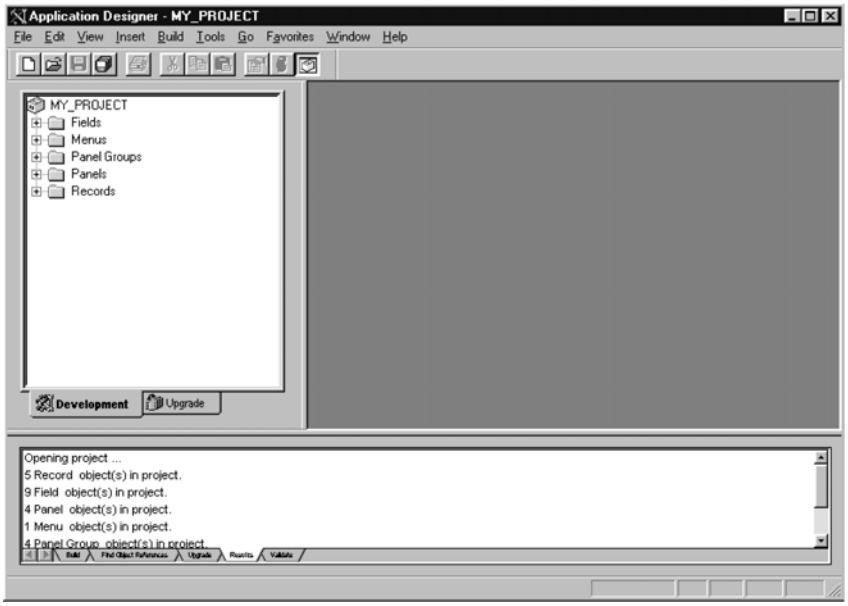


Figure 11.2 Initial project workspace

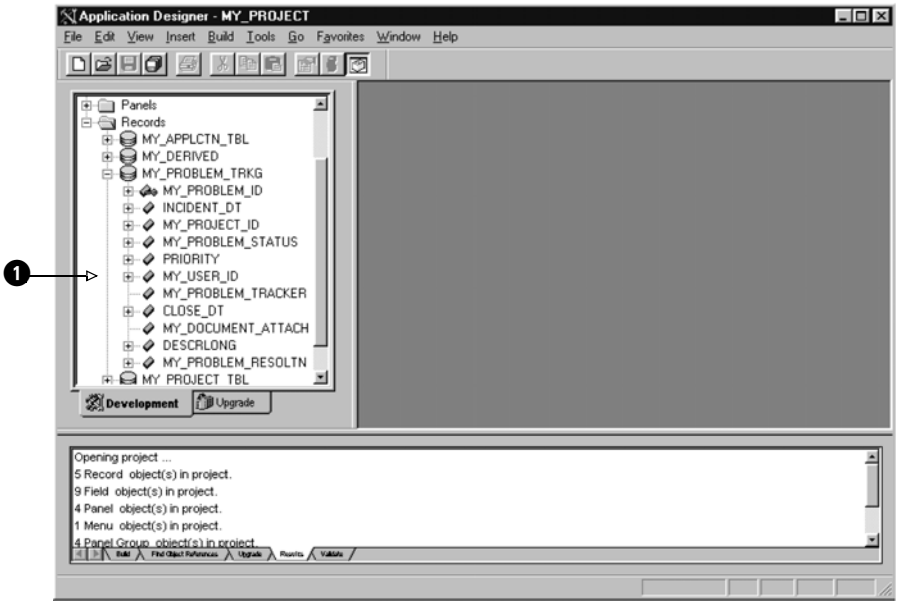


Figure 11.3 Record fields in the project workspace

The "+" next to the record field indicates that PeopleCode exists for that field. In figure 11.3 the record field MY_PROBLEM_TRKG.MY_USER_ID contains PeopleCode as indicated by the "+." When we click on the "+" for MY_USER_ID, any PeopleCode associated with the field is identified by the ⚡ symbol for that event. The example in figure 11.4, ❶ indicates that the field MY_USER_ID contains several record PeopleCode events.

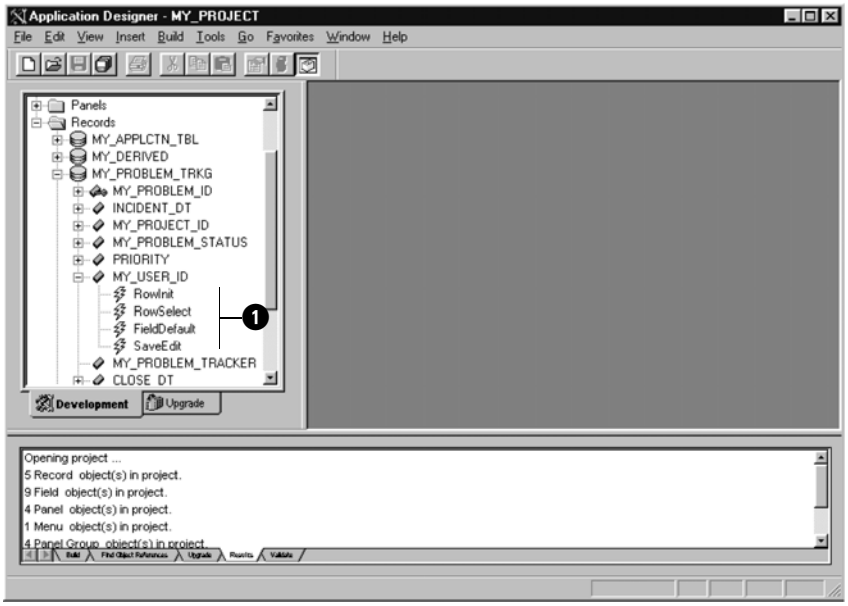


Figure 11.4 Record fields with PeopleCode

To view the record definition, double-click on the record name. We can also click on the desired record name within the workspace and, using the right mouse button, click on View Definition. An alternative method of viewing the record definition can be accomplished by double-clicking on the appropriate record name, which then enters the field display mode. After the record definition is displayed, we can view any associated PeopleCode from the PeopleCode display panel that displays the fields and corresponding events.

Another manner by which we can identify where PeopleCode exists for a record is to use the PeopleCode Display toolbar icon as illustrated by ❶ in figure 11.5. When viewing the PeopleCode Display panel, any field containing PeopleCode will have "Yes" displayed at the intersection of the fieldname and event.

Navigation: View →PeopleCode Display

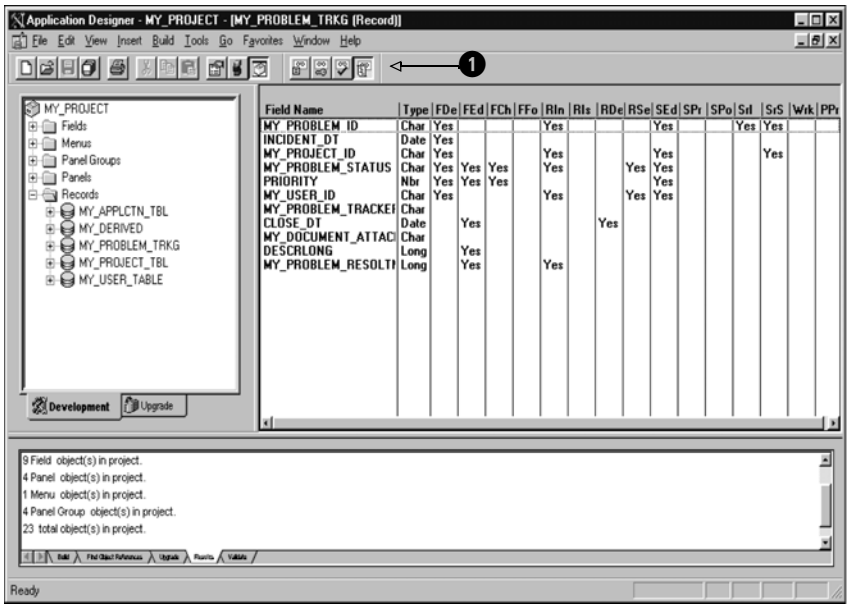


Figure 11.5 PeopleCode display view

To access PeopleCode through a panel definition, click on the field to which the code is to be added or modified. As illustrated in figure 11.6 the right mouse button is used to view PeopleCode for the Close Date field.

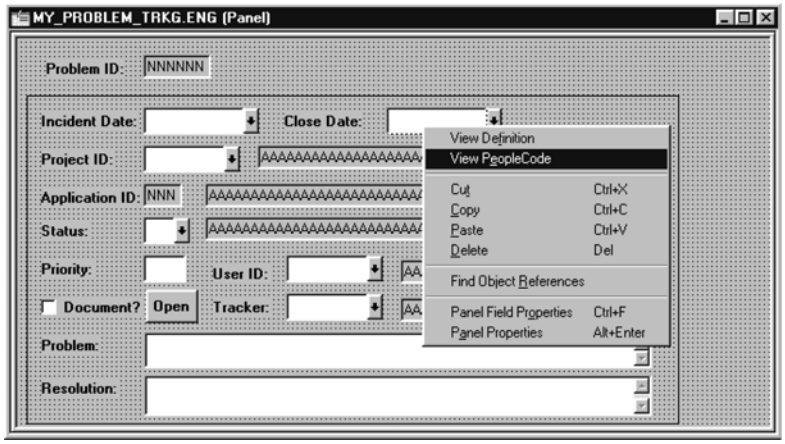


Figure 11.6 Accessing PeopleCode through a panel definition

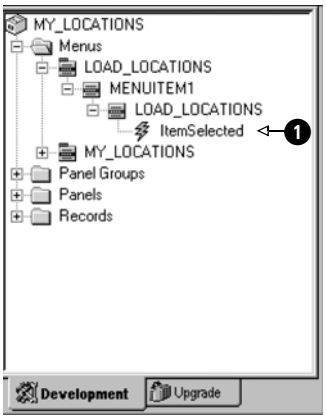


Figure 11.7 Accessing Menu PeopleCode

Menu PeopleCode can be accessed in a similar manner to record PeopleCode from the menu definition. Figure 11.7, ❶ illustrates the menu used for a small application used to link security operator classes and locations. Menu PeopleCode can be accessed using the right mouse button on the menu item or by double-clicking on the "⚡".

With a basic understanding of PeopleCode and how to access it, we are now ready to become familiar with the syntax, the rules, and the statements required to successfully implement applications that utilize PeopleCode.

KEY POINTS

1. PeopleCode is a programming language, which enables developers to extend the functionality of PeopleTools applications.
2. The Application Designer is used to insert or update PeopleCode statements.
3. PeopleCode events enable the developer to interact with the PeopleTools environment at key points during a work session.
4. Events occur during actions related to record fields. PeopleCode attached to all or some of these events is referred to as record PeopleCode.
5. Menu PeopleCode consists of the `ItemSelected` event and is linked to the selection of an item from a menu.

CHAPTER 12

PeopleCode language elements

The objective of this chapter is to introduce some of the fundamentals and basic building blocks of the PeopleCode language. In addition to the language constructs, the chapter illustrates how PeopleCode can be used to build simple programs using the materials presented in part 2.

12.1 PEOPLECODE AND RECORD FIELDS

In part 2, a record and it's associated fields were identified and built. The section "Creating a PeopleSoft panel definition" illustrates how fields are added to a panel with basic prompts and PeopleTools edits. Beyond the range of these edits is where knowledge and application of PeopleCode becomes a key element in the development and implementation of PeopleSoft applications. PeopleCode is linked to record fields, unlike fields in Application Designer that have the same characteristics, regardless of the records on which they exist. When a field named MY_PROBLEM_STATUS is defined, the field may appear on several different records. When the characteristics of the field are changed, the change is reflected throughout the database. If the field description or data type is changed, the modification is reflected on every table and panel containing the field MY_PROBLEM_STATUS. PeopleCode, on the other hand, is linked to fields through the record field definition. As an example, let's assume we've added PeopleCode to the `RowInit` event of the MY_PROBLEM_TRKG record, MY_USER_ID field. Using dot notation, it can also be specified as MY_PROBLEM_TRKG.MY_USER_ID.RowInit. The field MY_USER_ID also exists on the record MY_DERIVED; however MY_DERIVED.MY_USER_ID.RowInit does not contain this PeopleCode. Alternatively, when the length of the field MY_USER_ID is changed, the new length is reflected in both the MY_PROBLEM_TRKG and MY_USER_ID records.

12.2 PEOPLECODE EDITOR

In chapter 11 we discussed how PeopleCode can be viewed from a record field definition: by double-clicking on the lightening bolt "  " in the project workspace, from a menu item, or within the panel definition. The PeopleCode editor provides a facility to insert and maintain PeopleCode. Language statements, comments, and expressions entered are saved into the PeopleTools system table PSPCMPROG. This record is linked to the record containing record field definitions, PSRECFIELD. The PeopleCode editor performs syntax checking during save. Explicit syntax checking can be performed using a handy feature that validates syntax, available with release 7. Validate syntax is represented by the toolbar button  and verifies syntax without having to enable the save button. Records and fields used as bind variables in embedded SQL and scroll functions are also validated at this time. The syntax check edits a large percentage of code, but some rules in PeopleCode however are not strictly enforced. These errors can "slip by" and result in runtime error messages which can halt the processing of a panel—for example, an error might be a reference to a non-existing record enclosed within the quotes of a SQLExec statement or an invalid usage of scroll levels.

Navigation: Tools →Options →PeopleCode

Figure 12.1 Customization of PeopleCode editor font settings

Additional PeopleCode editor features include:

- customization of font settings, which can be changed by selecting Options →PeopleCode from the menu (figure 12.1)
- integration with Application Designer toolbar, menus, and accessibility via a pop-up window
- support for drag-and-drop PeopleCode text between independent programs

Let's view the PeopleCode for MY_PROBLEM_TRKG.MY_USER_ID.RowInit as illustrated in figure 12.2.

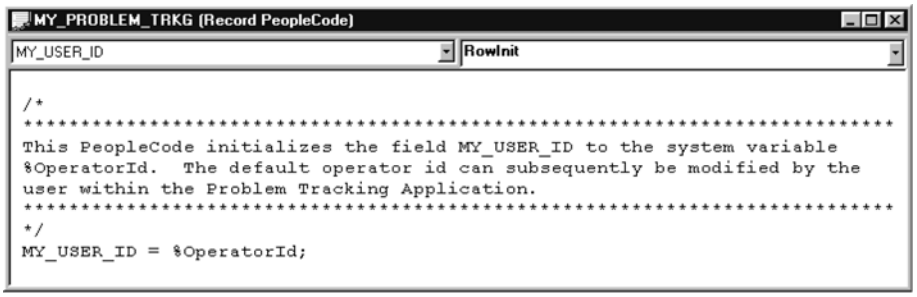


Figure 12.2 PeopleCode characteristics

The illustration in figure 12.2 identifies some PeopleCode characteristics:

- The header area identifies the record name and identifies this as PeopleCode attached to the MY_PROBLEM_TRKG record. In PeopleTools there are two types of PeopleCode: record and menu. The example identifies record PeopleCode.
- The area below the header displays the fieldname (MY_USER_ID) and the type of PeopleCode event (`RowInit`).
- Finally, the actual PeopleCode statements.

When new PeopleCode is added to or removed from a record, the record definition must be saved prior to exiting the PeopleTools session. A record definition does not have to be saved when existing PeopleCode is changed. The PeopleCode itself is saved to the appropriate PeopleCode tools tables.

12.3 PEOPLECODE COMMENTS

Figure 12.2 contains an example of PeopleCode statements. The first several lines contain comments. Comments provide an excellent method of internal code documentation. Information such as the date, the author, and the purpose of the code can be detailed in comments. Comments can exist anywhere in a PeopleCode program. They are not executed and can also be helpful during upgrades. If all our code were documented with a common text literal such as company name, we could conceivably search PeopleCode for the string and identify customizations. Two methods of inserting comments into a PeopleCode program are available. One form is to enclose comments with a leading /* and trailing */. This can be useful when testing code or debugging. It can also be helpful when one wishes to prevent PeopleSoft-delivered code from executing, but does not want to delete the code. The `REM` statement can also be used for comments. The `REM` or `REMark` statement is terminated with a semicolon (;). Comments beginning with a /* generate an error message when an attempt to validate syntax or a save operation is performed and the closing */ statement is missing. A misplaced semicolon (;) in a `REM` statement does not generate a message. It is possible that comments as well as PeopleCode statements are both treated as comments, because the scope of the `REM` statement terminates with the semicolon. Figure 12.3 shows some examples of valid comments and others that are not coded correctly. In the example, `&STRING_FIELD2` is preceded by a `REM` statement. The second `REM` is not terminated by a semi-colon and, therefore, `&STRING_FIELD2` initialization is treated as part of the comments.

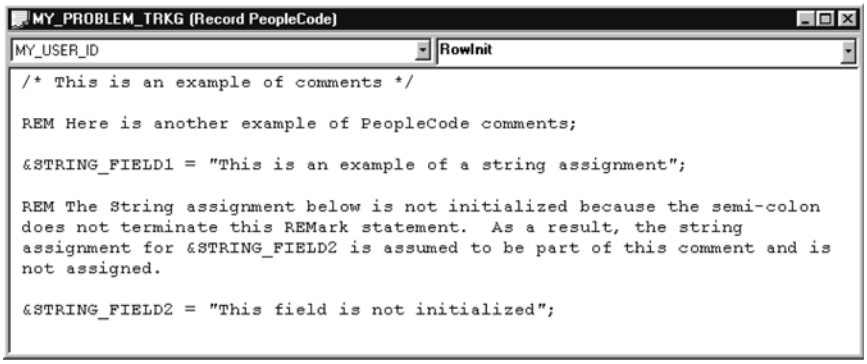


Figure 12.3 PeopleCode comments

12.4 DATA TYPES

The PeopleCode language contains several data types that permit operations on all categories of database fields. The examples that follow initialize variables. A

PeopleCode temporary variable is preceded by an ampersand ("&"), which assumes the data type of the assignment statement.

STRING	Strings are made up of any combination of characters that can be numeric, alphabetic, or special characters. A string can be initialized by enclosing the combination of characters in either single (') or double (") quotation marks. When one type of delimiter, either a single quote or double quote, is part of the string, it can be enclosed by the other delimiter type. Our first code example below is an example of single and double quotation marks used in string data types. PeopleCode functions exist, which operate on strings. Some are used to convert strings to numbers, combine or concatenate two or more strings, and trim a string either on the right or left. String functions will be reviewed as we progress through the chapters.
NUMBER	A number is any decimal value including integers and decimal points. A large percentage of the data types used in PeopleCode are either of the string or numeric type. As with strings, there are many functions that operate on the Number data type. Our second example below is an example of numbers in PeopleCode.
DATE	Dates are stored internally in the YYYY-MM-DD format. It is important to note that each platform and type of database stores dates differently, based on how the date is used. The Application Processor converts dates when they are loaded into a PeopleCode program. However, functions such as SQLExec, which operate directly on database tables and fields do not automatically convert dates. There are a number of built-in and Meta-SQL functions that work with dates and can be used to convert the formats.
DATETIME	A DateTime data type contains a date and a time expressed as YYYY-MM-DD-HH.MI.SS.SSSSSS. PeopleCode functions such as DatePart and TimePart extract the date or time portion. Functions such as DateTime6 and DateTimeValue generate a DATETIME data type.
TIME	Expressed as HHMISS. Functions involving TIME data type return numeric values expressed as seconds and sub-seconds.
BOOLEAN	A Boolean data type can have one of two values, TRUE or FALSE.
OBJECT	This data type is used with functions such as CreateObject. Word documents or Excel spreadsheets are examples of OLE objects.
ANY	This data type may contain any of the other data types. A field defined as ANY takes on the characteristics of the field with which it is initialized. A variable having no explicit data type declaration is ANY by default. PeopleCode determines the data type based on circumstances. It is therefore possible to have an undeclared field that can be used interchangeably by various data types and functions. The third example below provides us with a flavor of how ANY data types can be used.

The following is an example of single and double quotation marks used in string data types:

```
/* Here is an example of a string */
&STRING_FIELD1 = "This is a basic PeopleCode string";
&STRING_FIELD2 = "This is a string that contains a 'single quote'  delimiter
character";
&STRING_FIELD3 = "This string is enclosed within single quotes and contains
a string ""enclosed in double quotes""";
```

Where possible, enclose strings in double quotation marks and represent any embedded strings using single quotes.

Below, we can see examples of numbers in PeopleCode:

```
/* Examples of Numbers in PeopleCode */
&NUMBER_FIELD1 = 12345;
&NUMBER_FIELD2 = 123.45;
&NUMBER_FIELD3 = 0.12345;
&NUMBER_FIELD4 = - 123.45;
```

Let's look at an example of ANY data type:

```
/* Definition of an ANY data type */
Local any &WORK_FIELD;

/* ANY data type being set to a STRING */
&WORK_FIELD = " ";

REM  &Work_Field now becomes a NUMBER;
&WORK_FIELD = &NUMBER_FIELD * 100;

REM This concatenates two strings. &WORK_FIELD is a string again;
&WORK_FIELD = &STRING_FIELD1 | &STRING_FIELD2;

REM What datatype will the following statement return ? ;
&WORK_FIELD = %Date - &MY_BIRTHDATE;
/* If you guessed Number, you are correct. A number expressed as days is the
result. However, the ANY data type can also contain a date.   Below,
&WORK_FIELD is set to the current date;*/
&WORK_FIELD = %Date;
```

12.5 PEOPLECODE DATA ELEMENTS

12.5.1 Record field references

As we've already discussed in part 2, records are comprised of fields. How a field is referenced, retrieved, and initialized is of particular importance when the fieldname exists on various records accessible to a PeopleCode program. The syntax of a record field is as follows:

```
[RecordName].FieldName
```

Three components make up a record field name:

RecordName

RecordName is required when the PeopleCode program is in a record field other than the record where the PeopleCode resides. The name can contain from one to fifteen characters. At the database level, PeopleTools prefixes application tables with PS_. Within Application Designer however, the PS_ prefix is not used. When a record is opened or referred to in a panel, the prefix must be omitted. If specified, the presence of the prefix actually results in an 'Invalid record name' error message when used on a panel. The exception is the SQLExec statement that contains embedded SQL statements.

Period separator

The period is used when the record name prefix is required. When a PeopleCode program refers to a field on the same record as the program, the record name prefix is not required and is actually removed by the PeopleCode editor.

Fieldname

Both Fieldname and RecordName are not case sensitive and are converted to UPPERCASE by the PeopleCode editor. A fieldname can consist of one to eighteen characters and can include special characters such as @, _, $ and #. When used in field names, some of these special characters such as the # may be platform dependent. The RecordName.FieldName convention becomes more important when using specific functions that require an explicit RecordName.FieldName definition. Functions that use RecordName.FieldName are described throughout this book and in appendix E. More specific examples include:

```
FetchValue
FieldChanged
GetStoredFormat
Gray
Hide
PriorValue
SetCursorPos
SetDefault
SetDefaultAll
SetDefaultNext
SetDefaultPrior
SortScroll
SQLExec
Transfer
Ungray
Unhide
UpdateValue
```

12.5.2 Temporary variables

Temporary variables use the prefix '&' as part of their naming convention. Names can range from one to seventeen characters in length and consist of letters A–Z (a–z), digits 0–9 and special characters @, _, $ and #. Record field names can consist of one to eighteen characters, one more than variables because of the '&' prefix. The following illustrates the use of temporary variables and field names:

```
/* These are PeopleCode temporary Variables */
&STRING_FIELD1 = "This is a string temporary variable";

/* This is a numeric temporary variable */
&NUMBER_FIELD1 = 100;

&SPECIAL_2@10# = "This field name contains special characters";
```

TIP Although special characters can be included in record field names and variable names, they are not recommended except for the underscore character ('_').

12.5.3 Constants

Within PeopleCode, numeric, string, and Boolean constants are available. Numeric constants may be expressed as any number and can be included in assignments such as

```
&NUMBER_FIELD1 = 12345;
&NUMBER_FIELD2 = 123.45;
&NUMBER_FIELD13 = - 123.45;
&NUMBER_FIELD14 = 0.12345;
&NUMBER_FIELD5 = - 0.12345;
```

String constants are enclosed within single or double quotation marks. To enclose a single quotation as part of a string, we can place double quotation marks around it. Alternatively, we can enclose double quotations within single quotation marks:

```
&STRING1 = "Example of a basic string";
&STRING2 = "This is 'One string within another string'";
&STRING3 = "This is ""Another string within another""";
```

Boolean constants can be represented as `True` or `False`.

The following example verifies the Boolean return value issued by a function call:

```
If &RETURN_VALUE = True Then
   SetDefault(MY_PROBLEM_RESOLTN);
   SetDefault(CLOSE_DT);
End-If;
```

An alternative method of verifying a Boolean value is to omit the comparison operator:

```
If &RETURN_VALUE Then
   SetDefault(MY_PROBLEM_RESOLTN);
   SetDefault(CLOSE_DT);
End-If;
```

When &RETURN_VALUE returns False, the statements in the context of If are not executed.

12.5.4 System variables

Unlike temporary variables that can be defined by the developer, system variables are predefined and available within PeopleCode programs to access system information. System variables are prefixed with "%" whenever they are referenced in a program. Information such as date, time, current language, panel name, and additional information can be retrieved through system variables. System variables can be used in place of a constant or as part of an expression when assigning variables. They can also be passed to functions as parameters such as SQL strings passed to the SQLExec function. A list of system variables and descriptions follow.

%BPName	Returns a string containing the name of the Business Process from a worklist entry when accessed from a panel within a worklist. If the current panel group is not accessed from a worklist, an empty string is returned.
%Date	Returns the current date in YYYY-MM-DD format.
%DateTime	Returns the date and time as a Date/Time value in YYYY-MM-DD-HH.MI.SS.SSSSSS format.
%DbName	Returns the name of the current database.
%DbType	Type of database expressed as a string. Some database types include SQLBase, DB2, Oracle, or Microsoft.
%EmployeeId	Returns the employee ID of the operator currently logged on. This can be used to limit access to an employee's own data, but is only effective when the employee ID on the operator security record is populated correctly.
%Import	During Import Manager sessions this variable is returned as True. All other times it is False. This variable can be referenced if we wish to execute PeopleCode during import manager sessions only.
%Language	A character string is returned indicating the operator language preference.
%Market	Returns a string representing the Market property of the current panel group.
%Menu	Returns the current menu name. This uppercase string can be used to process actions based on menu. Specific edit or function calls can be controlled by menu name.
%MessageAgent	Contains a string representing the current message definition name when the current panel is invoked by a message agent routine. An empty string is returned when the panel is not initiated by a message agent routine.

%Mode	When an operator initiates a panel group associated with a menu item, this field will contain the menu action selected. The values can be: • "A" Add • "U" Update • "L" Update All • "C" Correction • "E" Data Entry
%OperatorClass	Returns the primary operator class for the current operator.
%OperatorRowLevelSecurityClass	Returns the row-level security class of the current operator. The row-level security class is different from the operator's primary class.
%OperatorId	The ID of the current user logged on. This entry exists on the PSOPRDEFN table.
%Panel	Returns the current panel name.
%PanelGroup	Name of the current panel group. As with %Menu, this variable can be used to control program flow or take specific actions according to the panel group. A panel group contains the panels associated within the group and can be used to identify panels to a PeopleCode program. The difference between %Panel and %PanelGroup is that %Panel identifies only the current panel on which the cursor is focused. Let's assume we have three panels, PANEL_A, PANEL_B, and PANEL_C. They belong to a panel group named PANEL_ABC. If our code reads as if %Panel = Panel.PANEL_B and the cursor is on PANEL_A, the If statement condition returns False. When the intent is to take action during FieldChange events on any of these panels, the code should read if %PanelGroup = PANEL_ABC.
%SQLRows	When using the SQLExec in conjunction with an UPDATE, DELETE, or INSERT operation, the number of rows affected by such a statement is returned in this variable. During the SELECT operation, this variable returns zero when no rows are selected and non-zero if one or more rows are selected. Unlike the other operations, the non-zero value does not indicate the number of rows returned for a SELECT.
%Time	Returns the current time in HH.MI.SS.SSSSSS format.
%WLInstanceID	When accessing a panel from a worklist, this variable contains the name of the worklist instance ID. The variable will contain blanks when the panel is not accessed from a worklist.
%WLName	Contains the name of the worklist. A blank is returned when the current panel is not accessed within a worklist.

The next example is used in the PeopleSoft HRMS application. This example verifies the %PanelGroup variable for the JOB_DATA_HIRE that is used in the new hire process. The example also references the %Mode variable.

```
If %PanelGroup = PANELGROUP.JOB_DATA_HIRE Then
      If %Mode = "A" Then
         FUNCLIB_HR.DEFAULT_SETID = &SETID;
      End-If;
   End-If;
```

12.5.5 Global and local variables

Variables can be defined as global or local. By default, all variables are defined as local and do not necessarily need the Local prefix. In release 7 of PeopleTools, a local

variable, the variable name, and its contents cease to exist at the conclusion of the PeopleCode event. As with function definitions and declarations, variable declarations must be placed above the main body of a PeopleCode program. A variable declared as ANY (or one that doesn't have an explicit declaration) will have the data type chosen by PeopleCode based on field contents. An ANY data type can store various data types such as String and Number. Some risk exists when a field is used for various data types. A function that requires a number to be passed to it may actually receive a string or date field, which can result in unpredictable errors when a field defined as ANY is passed to a function.

Global variables remain in effect during a PeopleSoft session and maintain their value from one panel group or PeopleCode event to another. The alternative is to pass values using derived/work record fields where possible. Passing and maintaining variables in a derived/work record allows us to share PeopleCode as well as work fields. In the Problem Tracking system, the table MY_DERIVED is an example of a derived/work record. This type record does not exist at the database level as a table. The fields on this type record can be shared across panels. Another useful application for derived/work records is the use of function libraries. (Function libraries will be discussed in chapter 16.) Examples of variable declarations are shown as follows:

```
/* Examples of variable declarations: */
Local string &NAME;
Local number &DEPARTMENT_COUNT;
Global number &TOTAL_COUNT;
Local any &ANY_TYPE;
```

TIP Global variables must be defined in every PeopleCode program that uses the variables. The PeopleSoft recommendation is to use global variables sparingly.

12.6 STATEMENTS AND EXPRESSIONS

This section examines how PeopleCode programs are constructed using various types of statements. PeopleCode statements include code that controls execution flow and can range from a simple If-Then-Else to complex loops. A statement can also be a simple expression. Let's examine statements and expressions and see how they work in unison.

12.6.1 Statements

Statements consist of data assignments, program language constructs, declarations, and subroutine calls.

A semicolon (;) is used to separate PeopleCode statements. The PeopleCode Editor disregards extra blank lines and spaces within the code. When a program is saved

to the database, excess spaces or blank lines are removed automatically by the editor. Let's consider some examples of PeopleCode statements:

```
/* This is a comment before a Declare Function Statement */
REM This is another comment before a Declare Function Statement;
/*This is an example of a function declaration statement */
Declare Function My_Schedule_Function PeopleCode MY_DERIVED.MY_USER_ID
FieldFormula;

&WORK_FIELD = "This is an example of a string assignment statement";

REM  This is a number assignment based on an "expression";
&RESULT_FIELD = &NUMBER_OF_ITEMS * &PRICE_PER_ITEM;

/* This is a function return statement */
&WORKSHEET_FIELD = CreateObject("Excel.Sheet");
```

Assignment statements, as represented by `&WORK_FIELD` above are the most basic types of statements within PeopleCode.

12.6.2 Control statements

Another type of PeopleCode statement is a control statement. A control statement is involved in the execution flow of a PeopleCode program and includes the following:

- `If-Then-Else`
- `Evaluate`
- `For`
- Loops with condition statements (`Repeat`, `While`)

If-Then-Else

When we write a PeopleCode program or any program for that matter, do we simply enter a bunch of statements and assume the program will figure it all out? If your answer to this is yes, then you've been watching too much science fiction (for the time being). Controlling the execution flow of a PeopleCode program is accomplished using branching statements, `For` loops and conditional loops. Basic examples of branching are illustrated below, using the `If-Then-Else` statement. The statement compares two strings and returns a message:

```
if &Text1 <> &Text2 then
   WinMessage ("Strings are not not the same");
 else
   WinMessage ("Strings, match!");
 end-if;
```

`If-Then-Else` construct statements are a key piece of program code that you will use often. The expression following the `If` keyword is evaluated as a logical `True` or `False`. If the evaluated expression is `True` (non-zero), PeopleCode executes all the

statements following `then` until an `Else` or `End-If` statement is encountered. These statements may also contain their own `If-Then-Else` statements. When the expression evaluated is `False`, the statements following the `Else` clause are executed. An `End-if` is required for every `If` statement.

The `Else` statement is not always required for an `If` statement. `Else` is specified when it is necessary to perform additional actions when an `If` condition is not satisfied.

The following is another example of an `If-Then-Else` statement used to identify the value of the Problem Status field:

```
If MY_PROBLEM_STATUS = "1" Then
   WinMessage("This incident is in an initiated status");
Else
   WinMessage("This incident is not in an initiated status");
End-If;
```

A nested `If-Then-Else` statement contains more than one `If` statement and may also contain more than one `Else` statement. The following is an example of nested `If` statements. Note, that for every `If` statement there is a corresponding `End-If`:

```
If MY_PROBLEM_STATUS = "1" Then
   WinMessage("This incident is in an initiated status");
Else
   If MY_PROBLEM_STATUS = "2" Then
      WinMessage("This incident has been assigned");
   Else
      If MY_PROBLEM_STATUS = "3" Then
         WinMessage("This incident is in progress");
      End-If;
   End-If;
End-If;
```

In the example below each `If` statement has a corresponding `End-If`. The placing of an `End-If` is important, because it defines the scope of an `If` statement:

```
If MY_PROBLEM_STATUS = "1" Then
   If MY_USER_ID = " " Then
      WinMessage("This incident is in an initiated status, but has no
assigned user");
   Else
      If MY_PROBLEM_STATUS = "2" Then
         WinMessage("This incident has been assigned");
      Else
         If MY_PROBLEM_STATUS = "3" Then
            WinMessage("This incident is in progress");
         End-If;
      End-If;
   End-If;
End-If;
```

The example above appears intact, but upon closer examination we see that the statements checking for a value of "2" or "3" will never be executed because the code is interpreted as follows:

If MY_PROBLEM_STATUS is "1" and MY_USER_ID is blank, send a message, otherwise, if MY_PROBLEM_STATUS is "2", send a message. As you can see, the second message, based on a value of "2" will never be sent because it is also based on the field MY_PROBLEM_STATUS having a value of "1". Because these two conditions can never co-exist (except in science fiction), the PeopleCode statements require some adjustment. The corrected code is shown below:

```
If MY_PROBLEM_STATUS = "1" Then
   If MY_USER_ID = "" Then
      WinMessage("This incident is in an initiated status, but has no
assigned user");
   End-If;
Else
   If MY_PROBLEM_STATUS = "2" Then
      WinMessage("This incident has been assigned");
   Else
      If MY_PROBLEM_STATUS = "3" Then
         WinMessage("This incident is in progress");
      End-If;
   End-If;
End-If;
```

TIP String comparisons are always case-sensitive.

Evaluate

Another form of branching is done using the Evaluate statement. This statement can be used when multiple conditions exist. The syntax of the Evaluate statement can be written as follows:

```
Evaluate Expression
       When Comparison
          [statements]
          Break
       When Comparison
          [more statements]
          Break
       When-Other[Optional]
End-Evaluate;
```

The Evaluate statement, in conjunction with When, compares an expression using relational operators in a series of When clauses. In a fashion similar to If-Then-Else statements, when the result of the comparison is True, the statements in the When clause are executed. Once these statements are completed, the operation moves

on to evaluate the comparison in a subsequent When clause. In a nutshell, the statements in which the When comparison results in a True condition are executed. The optional When-Other clause is executed after any previous When comparisons in the Evaluate statement are False. An Evaluate statement can be exited by using the Break statement. A good practice is to include the Break statement when the intent is to prevent subsequent When statements from executing. The following illustrates the use of the Evaluate statement and sends a message based on the contents of the %Language system variable.

```
Evaluate %Language
When = "ENG"
   &MESSAGE = "We are using English";
When = "ESP"
   &MESSAGE = "We are using Spanish";
When = "FRA"
   &MESSAGE = "We are using French";
When = "GER"
   &MESSAGE = "We are using German";
When = "INE"
   &MESSAGE = "We are using International English";
When-Other
   &MESSAGE = "We are using another language";
End-Evaluate;
```

The statements used above can be rewritten using nested If statements (such as those below). Notice the additional statements required to accomplish the same task using the Evaluate statement.

```
If %Language = "ENG" Then
   &MESSAGE = "We are using English";
Else
   If %Language = "ESP" Then
      &MESSAGE = "We are using Spanish";
   Else
      If %Language = "FRA" Then
         &MESSAGE = "We are using French";
      Else
         If %Language = "GER" Then
            &MESSAGE = "We are using German";
         Else
            If %Language = "INE" Then
               &MESSAGE = "We are using International English";
            Else
               &MESSAGE = "We are using another language";
            End-If;
         End-If;
      End-If;
   End-If;
End-If;
```

Evaluate statements may behave differently in other programming languages. The following example contains a small piece of code. In the example, the variable &A is evaluated. Assuming the value of &A is 1 at the time of the Evaluate statement, the value is then changed to 2. At the time PeopleCode executes the second When statement, which compares the value &A to 2, the original value of &A is evaluated. We can therefore state that the results of each When statement are based on the condition in place at the time of the Evaluate.

```
Evaluate &A
When = 1
&A = 2;
When = 2
&A = 3;
End-Evaluate;
```

SQR users may be aware that Evaluate statements work differently. If the field being evaluated has its value changed within the When statement, it is possible that the newly changed value will impact subsequent When statements because the new value is evaluated in each When statement. In SQR, the second When statement would have been executed assuming the initial value is "1" and then is changed to "2". A good practice to avoid these pitfalls is to use the Break statement. While this does not occur within PeopleCode, we can rewrite the Evaluate statement below to develop good programming habits:

```
Evaluate &A
When = 1
   &A = 2;
   Break;
When = 2
   &A = 3;
   Break;
End-Evaluate;
```

Because the Break statement transfers control to the End-Evaluate, any remaining When statements are bypassed, which also improves performance.

For

For can be a useful statement when the need to execute statements repetitively becomes necessary. The statement works in conjunction with an initial setting of a variable that is subsequently incremented by a value—after the statements in the scope loop are executed. The format of the For statement is as follows:

```
For count = expression1 to expression2
        [Step i];
        PeopleCode Statements
     End-For;
```

The statements in the loop are continuously executed until expression2 is true. Step represents the increment value that is added to the count field each time the loop is executed. When Step is omitted, the default increment is 1. When we wish to decrement by 1 or count backwards, use Step –1 as expression2.

Loops can be nested—that is, they can contain other For or While loops as well as other types of PeopleCode statements. The Break statement terminates the current active loop. When the current loop is part of a nested loop, any higher level loops are returned following the Break statement. The higher level loops can be subsequently terminated with a Break statement or when the value of count is equal to expression2. If no higher level loop exists, processing continues with the statement following the end of the loop.

The following is an example of a For loop which contains an update to a field on a scroll area. The loop begins on row number one and continues until all the number of active rows in the scroll are processed. The scroll function ActiveRowCount is used to obtain the number of active rows. The variable &I is incremented by 1 based on the rules of the For statement.

```
For &I = 1 To ActiveRowCount(RECORD.MY_LOCATIONS);
   UpdateValue(MY_LOCATIONS.EFFDT, &I, &EFFDT);
End-For;
```

The next example combines the decrement feature with an If statement. The loop begins with the highest active row and is processed from highest to lowest using the Step –1 statement.

```
If %Panel = "MY_LOCATIONS" Then
   For &I = ActiveRowCount(RECORD.MY_LOCATIONS) To 1 Step - 1;
      DeleteRow(RECORD.MY_LOCATIONS, &I);
   End-For;
End-If;
```

Repeat

The Repeat statement initiates a loop and executes the statements within the scope of the loop until the PeopleCode expression is True. These statements can be other loops or other PeopleCode statements and function calls. In a manner similar to loop statements such as For and While, a Break statement within a Repeat loop returns control to the next higher level loop. When the current loop is not part of a higher level loop, processing continues with the statement following the end of the loop.

The format of the Repeat statement is as follows:

```
Repeat
     [Statement List]
Until
     [expression]
```

The example below illustrates the use of Repeat to obtain the effective date and remove the row from a scroll area. The function ActiveRowCount is used to obtain the number of rows the loop will process. The FetchValue function obtains the effective date, and the subsequent RowFlush removes the row from the scroll area.

```
&I = 0;
/* Obtain number of active rows */
&ROW_COUNT = ActiveRowCount(RECORD.MY_LOCATIONS);
Repeat
   &I = &I + 1;
   &EFFDT = FetchValue(EFFDT, &I);
   If &EFFDT < %Date Then
      RowFlush(RECORD.MY_LOCATIONS, &I);
   End-If;
Until &I = &ROW_COUNT;
```

TIP The Break statement—not an If statement—terminates the loop.

While

The While control statement initiates a loop and is repeated until the PeopleCode expression is False. This can be somewhat misleading because many loop statements terminate when the condition is True. The While statement, however, terminates when the condition is False.

And you thought this was going to be another clone of the Repeat and For statements! In many aspects the statements are the same, including the execution of the statements within the scope of the loop. These statements can be other loops or other PeopleCode statements and function calls. As with other loop statements, a Break within a loop returns control to the next higher level loop or processing continues with the statement following the end of the loop. The difference and inherent danger particularly with the While statement is the potential for a runaway loop. Because the statement is terminated as a result of a False condition, improper wording often leads to more runaway loops with the While statement than with other loop statements.

The format of the While control statement is:

```
While
     [PeopleSoft expression]
     [Statement List]
End-While;
```

Following is an example of a While statement that displays a message and numeric data type variable several times.

```
&MAX_LOOP = 0;
While &MAX_LOOP < 5
```

```
   WinMessage("This is pass number " | &MAX_LOOP);
   &MAX_LOOP = &MAX_LOOP + 1;
End-While;
```

The following is an example of what NOT to do when using the `While` statement. The example results in a runaway loop, because in this sample the value of `&MAX_LOOP` is always going to be greater than zero.

```
&MAX_LOOP = 1;
While &MAX_LOOP > 0
   WinMessage("This is pass number " | &MAX_LOOP);
   &MAX_LOOP = &MAX_LOOP + 1;
End-While;
```

12.6.3 Expressions

PeopleCode expressions

Basic PeopleCode expressions can be written as:

```
&WORK_FIELD = Expression;
```

In the example, the result of the expression to the right is placed into a variable named `&WORK_FIELD`. The target of an expression is either a variable or a record field. In their most basic form, expressions are characterized by one or more data elements on either side of the assignment operator. The assignment operation in the example below is used to initialize the record field MY_USER_ID with the current operator ID.

```
MY_USER_ID = %OperatorId;
```

Expressions can combine any of the following:

- constants
- temporary variables
- system variables
- record fields
- other expressions
- function parameters

Simple expressions can be combined into compound expressions using math operators:

```
&X = (&A + &B) * ((&D - &C) / &E);
```

In PeopleCode the equal sign (=) has a dual purpose. It behaves as an assignment operator, in the preceding example or as a comparison operator. (Remember our example that combined the decrement feature with an `If` statement.) On one line, the equal sign is used as a comparison operator, and on the following line, as an

assignment operator. The context of the equal sign is dependent on how a particular expression is used and the PeopleCode statements that contain the expression. A simple assignment operation contains a variable to the left and another variable, constant or literal, to the right. A comparison operation may use a statement such as `If`, which implies that the equal sign is being used as a comparison operator.

Putting PeopleCode expressions to work

Now that we have a basic understanding of PeopleCode, we can begin to put together simple statements and expressions using strings, dates, and operators for statements using comparison, math, and Boolean expressions.

The combining of two or more strings can be accomplished using the vertical bar symbol (|) which acts as a concatenation operator. Operands supplied to the concatenation operator are automatically converted to strings. An example of string concatenation is illustrated below:

```
&TEXT_MESSAGE = "Hello " | %OperatorId | "The current time is " | (%Time);
WinMessage(&TEXT_MESSAGE);
```

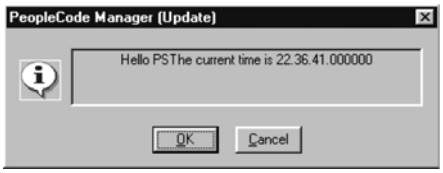


Figure 12.4 Contents of concatenated string

In the example, we see that the statement is a string assignment. Two string literal types and two system variable types exist here. The expression on the right of the equal sign is a concatenation expression. The statement is terminated with a semi-colon.

The string variable &TEXT_MESSAGE contains the concatenation results displayed in the message in figure 12.4.

NOTE All data types used in concatenation expressions are converted to string.

Expressions using Date/Time values can be constructed to perform date arithmetic. Dates/Time values can be added or subtracted resulting in a number. The result is a value expressed as number of days. When working with time variables, the number result is a value expressed in seconds. Constants, variables, and record fields containing numbers can be added to Date/Time values. The resulting number will be days or seconds.

As an example, to determine the number of elapsed days since the initial reporting of a problem in the Problem Tracking system, we could construct the statement shown below.

```
&DAYS_ELAPSED = (%Date - MY_PROBLEM_TRKG.INCIDENT_DT);
```

Here, the variable `&DAYS_ELAPSED` contains a number that represents the difference between the system variable `%Date` and the date value contained in MY_PROBLEM_TRKG.INCIDENT_DT.

Comparison operators are used when we wish to compare two expressions containing the same data type. A Boolean value is returned as a result of the comparison. Comparison operators are represented by the following symbols:

=	equal
<>	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Let's now take a look at some comparison operators. When comparing strings, comparisons are always case-sensitive. The following compares two strings that contain the same words. Do you know what message is displayed?

```
&TEXT1 = "TODAY IS A FINE DAY";
&TEXT2 = "Today is a fine day";
If &TEXT1 <> &TEXT2 Then
   WinMessage("Strings are not not the same");
Else
   WinMessage("Strings, match!");
End-If;
```

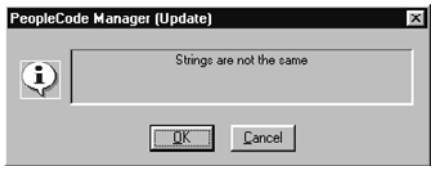


Figure 12.5 Result of string comparison

Figure 12.5 indicates that although the strings contain the same words, one string is uppercase and the other string is mixed case. The comparison results in a `False` condition when both strings are compared. Math Operators are your normal everyday arithmetic operators. The following represents the arithmetic operators used in PeopleCode:

+	addition
-	subtraction
*	multiplication
/	division
**	exponential

In PeopleCode, math operations are performed within a hierarchy as illustrated. The hierarchy of operations is specified explicitly. When used in an operation, parentheses override the hierarchy.

Highest

Unary operator	+ B or - C
Exponeniate	(A ** B)
Multiply, Divide	(A * B / C)
Add, Subtract	(A + B - C)
Relational, sign, conditions	(A > B)
Logical NOT	NOT (A > B)
Logical AND	(A AND B)

Lowest

Logical OR	(A OR B)	
(A * B * C)	[Same as]	(A * B) * C
(A + B - C ** D)	[Same as]	(A + B) - (C ** D)

TIP Operations having equal hierarchy are evaluated from left to right

---

Boolean operators are formed by the logical operators `And`, `Or`, and `Not`. Examples of Boolean operators are shown below. As with mathematical operators, parentheses can be used to override precedence.

```
/* Example of Boolean operators and expected results */
If &A > &C And
      &B > &A Or
      &C < &B Then
   &MESSAGE = "This is an example of Boolean operators and is TRUE when the
value of &A is greater than &C  and the value of &B is greater than &A Or
the value of &C is less than &B.  Our example can be rewritten with paren-
theses";
End-If;

If (&A > &C And
      &B > &A) Or
      &C < &B Then
   &MESSAGE = "This example is TRUE when the value of &A is greater than &C
AND the value of &B is greater than &A. The example is also TRUE if the
value of &C is less than &B ";
End-If;
```

12.7 PEOPLECODE TOOLS TABLES

After inserting or modifying PeopleCode statements, it is necessary to save the code. When the save button is pressed, the related PeopleCode system tables that are updated include:

PSPCMNAME is the PeopleCode reference table. It contains the internal name assigned by PeopleTools and all other record fields referred from a PeopleCode event. As

an example, the panel MY_PROBLEM_TRKG contains data elements from the record MY_PROJECT_TBL. When a reference is made to fields on MY_PROJECT_TBL from a PeopleCode program which resides on the MY_PROBLEM_TRKG record, a row exists for each field referred to on MY_PROJECT_TBL, from PeopleCode in MY_PROBLEM_TRKG. Figure 12.6 contains a query of the PSPCMNAME table contents for MY_PROBLEM_TRKG.MY_USER_ID. We see that `PCM105665` is the internal name, ❶ assigned by the system to an event for the field MY_USER_ID.

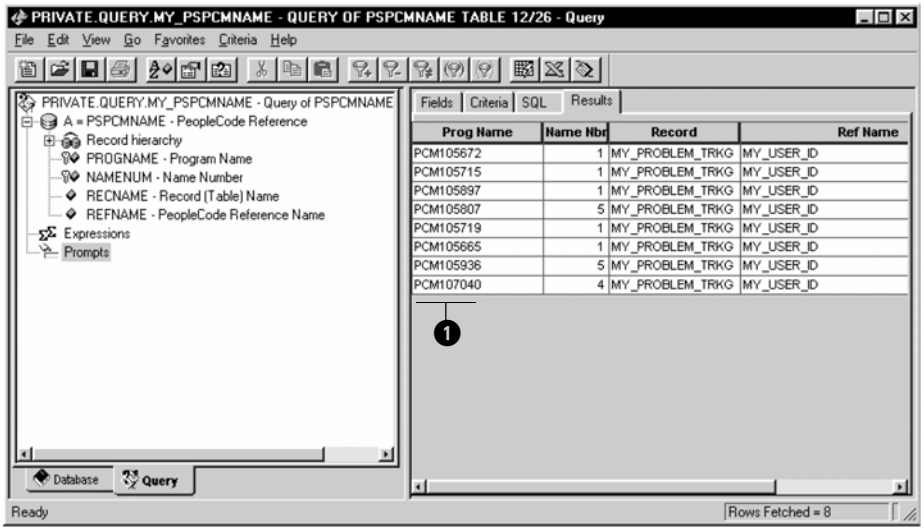


Figure 12.6 Contents of PSPCMNAME

PSPCMPROG contains the internal PeopleCode name and Date/Time stamp of the last update including the user ID. The table also contains the actual PeopleCode program text.

PSPROGNAME contains the internal PeopleCode name, fieldname and record name. Figure 12.7 contains the table entries for internal program name `PCM105665`.

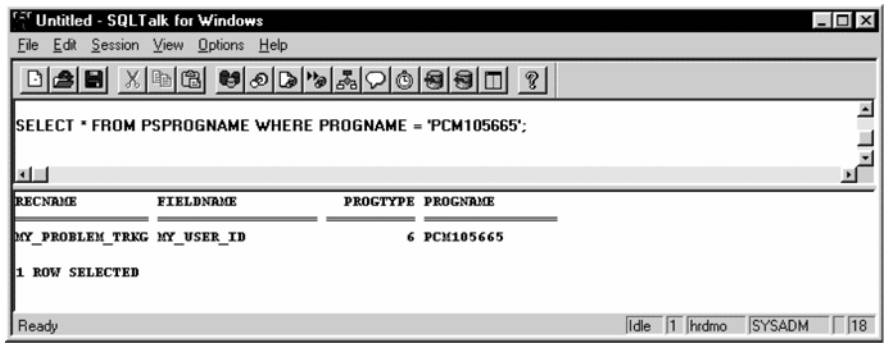


Figure 12.7 Contents of PSPROGNAME

PSRECFIELD contains records and their associated field definitions. One important field on this table is PROGCOUNT which contains the number of events for a record field that contains PeopleCode. A SQL SELECT statement (figure 12.8) for the record field MY_PROBLEM_TRKG.MY_USER_ID indicates a PROGCOUNT value of 4, which identifies PeopleCode existing in several events for the record field.

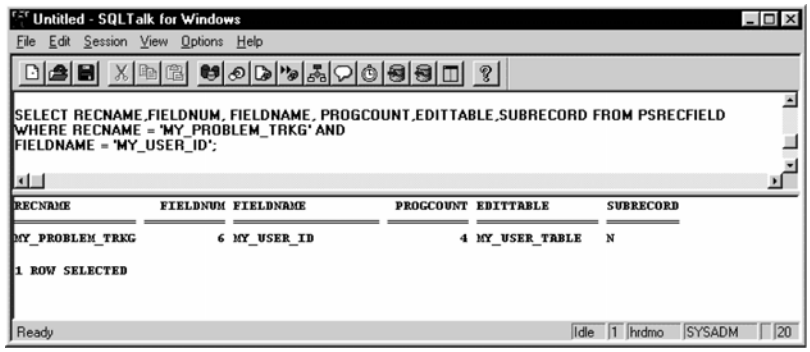


Figure 12.8 Contents of PSRECFIELD

KEY POINTS

1. The PeopleCode editor provides us with a facility to insert and maintain PeopleCode statements.
2. PeopleCode comments allow for internal code documentation.
3. PeopleCode handles various data types such as String, Numeric, Date, Time, and Object. A Boolean data type can only have a `True` or `False` value.
4. Record Field naming standards enable the PeopleCode program to access the contents of fields. The [RecordName].FieldName format is used to access fields in records other than the record in which the PeopleCode resides.
5. System variables are available to PeopleCode programs to access system information such as current date, database type, operator ID, and operator security class.
6. PeopleCode recognizes global and local variables. Global variables remain in effect during a PeopleSoft panel session but use more overhead. Local variables only exist during the PeopleCode event.
7. Statements include data assignments, declarations, and subroutine calls. Expressions can be constants, variables, record fields, or values passed to functions.
8. PeopleCode execution flow can be handled using `If-Then-Else`, `Evaluate`, `For`, `Repeat`, and `While` statements.

CHAPTER 13

PeopleCode & the Application Processor

The Application Processor is the system tool responsible for carrying out tasks such as displaying selected panels and panel groups, invoking PeopleCode at various events and controlling updates to database records. Having a good understanding of how the Application Processor interacts with PeopleCode is important when designing an application or determining where to insert custom code.

13.1 THE APPLICATION PROCESSOR

Imagine, if you will, a busy intersection with no traffic lights at the height of rush hour on the Friday afternoon of a long holiday weekend. No traffic lights or traffic officer! This is a scene of chaos if ever there were one.

When an end user sits in front of his/her terminal during a PeopleSoft session, there can be much confusion from the perspective of the operating system and PeopleTools environment. All those mouse clicks and menu selections have to be controlled and guided somehow. The PeopleTools Application Processor performs just such a function. When the end user requests information or wishes to update information, the panels, menus and panel groups are controlled by… you guessed it, the Application Processor! The Application Processor also interacts with PeopleCode programs, and this interaction is the cornerstone of PeopleCode within the PeopleTools environment. During the course of an online request, events are triggered which may result in the execution of PeopleCode programs tied to such events.

PeopleCode programs are tied to an event and either a record (record PeopleCode) or a menu item (menu PeopleCode). If, during the course of a work session, an event is triggered that is tied to PeopleCode, the statements within that event are executed. The Application Processor ensures that each event is allowed its fair share of memory and an "at bat" in the event line-up. Menu PeopleCode operates when menu items are selected. Record PeopleCode operates on data rows associated with record objects and is normally triggered during events in an online session.

PeopleCode events are utilized during the phases of an online session. Chapter 9 discussed The Application Processor and how it interacts within the following stages:

- search processing
- data retrieval
- panel group display
- data entry or inquiry

During these stages, PeopleCode programs can be inserted to enhance an application or perform required tasks that cannot be easily accomplished using basic Application Designer panel functionality. Knowing where to insert code is just as important as knowing how to write code. Events exist which may not be triggered based on specific actions. Some events are based on actions that occur before a panel group is displayed. These events are related to search processing, interpretation of search key values, and default processing. When new data are added, certain events and actions occur that do not occur when existing data are displayed for update. Whether data are being added or displayed for update, PeopleCode can be inserted into these events to establish default values or control the look and feel of menus, panels, and scroll areas as well as to submit batch processes when necessary.

13.2 SEARCH PROCESSING

When a menu item is selected, the Application Processor interacts with PeopleCode events based on the menu action requested. Where PeopleCode is placed is important because some events do not permit specific actions. For example, a SQLExec SELECT statement that enables the execution of SQL statements against a database table can be placed in any event. SQLExec statements containing database updates, however, are only allowed during events such as `SavePreChg`, `WorkFlow`, and `SavePostChg`. Similarly, message functions containing more than one button are not allowed during "Think Time" PeopleCode events. Think Time PeopleCode events are actions that interrupt processing and wait for a user reply to a message box containing more than one push button. A message box with multiple buttons can impact the course of a program's flow.

13.2.1 Menu item is chosen

A panel session is commonly initiated with the selection of an item from a menu. As discussed in part 2, menu actions can be `Add`, `Update/Display`, `Update/Display All`, or `Correction`.

When a menu item is selected, the Application Processor loads into its memory buffers the panel group definition and search records associated with the selected menu item. Included in these objects are the records and events containing PeopleCode associated with the Application Processor's flow of execution. Within these events, the Application Processor retrieves the necessary database keys for the records contained in the panel group. Records that make up the panels in a panel group are retrieved and presented to the end user on one or multiple panels.

13.2.2 Search processing—Add mode

Panel group startup process and associated PeopleCode events are triggered, depending upon the menu action selected. We have developed a small application which links security operator classes and office locations. This application is used to link employees to the operator class/location combination, primarily for reporting purposes. Let's assume the menu contains two actions, `Add` and `Update/Display`. After selecting `Add` from a menu, an end-user is presented with an `Add` dialog box. Fields defined as search keys in the search record are assembled on the input dialog box. In `Add` mode, however, fields defined as alternate search keys in the search record do not appear on the input dialog box. To a user, an `Add` dialog box is automatically displayed; but to the Application Processor and any associated PeopleCode, it's another Friday afternoon on the freeway!

Before an `Add` dialog box is displayed several PeopleCode events occur. The events are

- `FieldDefault`
- `FieldFormula`