

Introduction to Pathmaker

Abstract This manual introduces the Pathmaker product. It defines basic Pathmaker terminology, suggests several approaches for successfully using the product, and describes how an application created with the Pathmaker product looks and behaves.

Part Number 067867

Edition Fourth

Published September 1993

Product Version Pathmaker D20

Release ID D20.00

Supported Releases This manual supports C30 and all subsequent releases until otherwise indicated in a new edition.

Document History	Edition	Part Number	Product Version	Earliest Supported Release	Published
	Fourth	067867	Pathmaker D20	C30/D10.00	September 1993

New editions incorporate any updates issued since the previous edition.

A plus sign (+) after a release ID indicates that this manual describes function added to the base release, either by an interim product modification (IPM) or by a new product version on a .99 site update tape (SUT).

Copyright Copyright © 1993 by Tandem Computers Incorporated. Printed in the U.S.A. All rights reserved. No part of this document may be reproduced in any form, including photocopying or translation to another language, without the prior written consent of Tandem Computers Incorporated.

Export Statement Export of the information contained in this manual may require authorization from the U. S. Department of Commerce.

Examples Examples and sample programs are for illustration only and may not be suited for your particular purpose. Tandem does not warrant, guarantee, or make any representations regarding the use or the results of the use of any examples or sample programs in any documentation. You should verify the applicability of any example or sample program before placing the software into productive use.

Ordering Information For manual ordering information: Domestic U.S. customers, call 1-800-243-6886; international customers, contact your local sales representative.

New and Changed Information

The *Introduction to Pathmaker* manual has been completely restructured and rewritten for this release of the product. The previous version of this manual was written for the B41 version of the Pathmaker product; since that time, several new versions of the product have been released. This manual describes the product based upon the Release 3 D20 version of the Pathmaker product.

If you have already used previous versions of the Pathmaker product and want to quickly learn about the new features and capabilities of the product for Release 3, read the summary of new Pathmaker features in the New and Changed Information section of the *Pathmaker Programming Guide* for Release 3. In addition to providing a brief overview of the new features and capabilities, this summary includes a table that indicates where in the Pathmaker manual set detailed information about the Release 3 enhancements can be located. This summary is written specifically for readers who have used previous versions of the Pathmaker product and are already very familiar with the product's basic features.

The operating system for Tandem NonStop systems, formerly called the Guardian operating system, is now called the Tandem NonStop Kernel. This change reflects Tandem's current and future operating system enhancements that further enable open systems and application portability.

Contents

About This Manual xi

About the Pathmaker Manual Set xiii

Section 1 Introduction to Pathmaker and Pathway

Overview of Pathmaker 1-1

 Benefits of Using Pathmaker 1-1

 How Pathmaker is Used 1-2

Online Transaction Processing (OLTP) Applications 1-4

 Functions Performed by OLTP Applications 1-6

 Requirements of Online Transaction Processing 1-8

Overview of Pathway 1-9

 Benefits of Using Pathway 1-9

 How Pathway is Used 1-10

 Creating Applications Without Pathway 1-10

 Creating Applications With Pathway 1-11

 Requester Functions 1-13

 Server Functions 1-13

 Benefits of Using Requesters and Servers 1-14

 Service Functions 1-14

 Benefits of Using Services 1-14

 Communication With IPC Messages 1-16

 Request Messages 1-16

 Reply Messages 1-16

 Running Applications With Pathway 1-18

 Process 1-18

 PATHMON 1-18

 PATHCOM 1-18

 SCREEN COBOL Library 1-18

 TCP (Terminal Control Process) 1-18

 Database 1-19

 Service 1-19

 Server Process 1-19

 Server Class 1-21

 Context-Free Services 1-22

How Pathmaker Helps	1-26
Application Creation Assistance	1-26
Application Creation Tasks	1-27
Pathmaker Full Screen Interface Features	1-28
Pathmaker Application Definition Language (PMADL)	1-29
Management Assistance	1-32
Application Design Assistance	1-33
Tandem Databases and TMF	1-34
Tandem Database Products	1-34
NonStop SQL	1-34
Enscribe	1-34
Enscribe DDL (Data Definition Language)	1-35
Database Types and Pathmaker	1-37
Pathmaker and the DDL Dictionary	1-37
NonStop SQL Table Registration	1-37
Database Creation	1-40
Overview of TMF	1-40
TMF and Database Consistency	1-40
How to Protect an Application Using TMF	1-41

Section 2 Pathmaker Applications

The Architecture of a Pathmaker Application	2-1
Screen Layout	2-1
Multiple Pages—One Logical Screen	2-2
Function Key Actions	2-4
Help for the End User	2-4
Data Field Attributes	2-6
Operating a Pathmaker Application	2-7
Entering Data	2-7
Displaying Defaults	2-8
Receiving Data and Messages	2-8
Screen Navigation Within a Pathmaker Application	2-9
Customizing Pathmaker Applications	2-11
Screen Layout	2-11
Screen Painter	2-11
Help Text	2-12
Messages	2-12
Advanced Customization Techniques	2-12

Types of Pathmaker Applications	2-13
DB Requester Applications	2-13
Operating a DB Requester Application	2-14
Components of a DB Requester Application	2-16
Custom Applications	2-18
Operating a Custom Application	2-18
Components of a Custom Application	2-19
Other Capabilities	2-22
Accessing Pathway Applications From a Pathmaker Application	2-22
Pathway Open Environment Toolkit (POET)	2-22

Section 3 Using Pathmaker—An Overview

Major Pathmaker Components	3-1
Pathmaker Catalog	3-2
Full Screen Interface	3-3
Pathmaker Application Definition Language (PMADL)	3-3
Skeleton Files and Code Generators	3-3
Utilities	3-4
A Pathmaker Project	3-5
Project Catalog	3-7
Project Subvolume	3-8
Approaches for Using Pathmaker	3-9
Creating Production Applications	3-9
Designing Screens	3-9
Using Simulation	3-10
Creating Prototypes	3-11
A Suggested Life Cycle for a Pathmaker Application	3-12
Definition	3-12
Design	3-13
Planning	3-14
Setup	3-15
Development	3-15
Documentation	3-16
Testing	3-16
Move Into Production	3-16
Maintenance	3-16
A Final Note	3-18

Glossary Glossary-1

Index Index-1

Figures	Figure 1-1.	The Pathmaker Product in the Development and the Production Environments 1-3
	Figure 1-2.	Online Transaction Processing Applications 1-6
	Figure 1-3.	Functions of an OLTP Application 1-7
	Figure 1-4.	Monolithic Application 1-10
	Figure 1-5.	Requester-Server Application 1-12
	Figure 1-6.	Creating an OLTP Application With Pathway (Without the Pathmaker Product) 1-15, 1-30
	Figure 1-7.	IPC Messages 1-17
	Figure 1-8.	Components of a Running Pathway System 1-20
	Figure 1-9.	Server Class 1-21
	Figure 1-10.	Context-Free Services 1-23
	Figure 1-11.	Creating and Running an OLTP Application 1-25
	Figure 1-12.	Creating an OLTP Application With the Pathmaker Product 1-31
	Figure 2-1.	Requester Screen 2-3
	Figure 2-2.	Requester Help Screen 2-5
	Figure 2-3.	Preliminary Input Validation 2-8
	Figure 2-4.	Error Message From a Service 2-9
	Figure 2-5.	Hierarchical Pathmaker Application 2-10
	Figure 2-6.	DB Requester Application Screen 2-15
	Figure 2-7.	A Menu Screen 2-17
	Figure 2-8.	Custom Application Screen 2-19
	Figure 3-1.	Major Components of the Pathmaker Product 3-2
	Figure 3-2.	Pathmaker Projects 3-6
	Figure 3-3.	A Project Subvolume 3-8

Tables	Table 1-1.	Comparison of NonStop SQL and Enscribe Products (Page 1 of 2)	1-38
	Table 1-1.	Comparison of NonStop SQL and Enscribe Products (Page 2 of 2)	1-39
	Table 3-1.	Life Cycle of a Pathmaker Application	3-17

About This Manual

The Pathmaker product is a tool that assists in the creation of Pathway requester-server applications for computer systems that use the Tandem NonStop Kernel. The *Introduction to Pathmaker* manual provides a comprehensive introduction to the Pathmaker product. The manual explains how an application created with the Pathmaker product looks and behaves, defines basic Pathmaker terminology, and suggests several general approaches for successfully using the product. In addition, basic information about Pathway and online transaction processing (OLTP) is included. The information in this manual provides the foundation needed to effectively use the other books in the Pathmaker manual set.

Objectives of This Manual

Given this manual, readers will be able to:

- Identify many of the benefits and features of the Pathmaker product.
- Define Pathmaker terminology and concepts.
- Define Pathway terminology and concepts.
- Identify the main purpose of these Tandem products: Pathmaker, Pathway, NonStop SQL, Enscribe, Enscribe DDL, and TMF.
- Explain the architecture of an application produced with the Pathmaker product.
- Describe the types of applications that can be produced with the Pathmaker product.
- List the types of requesters and services used for creating Pathmaker applications.
- Explain where the Pathmaker product fits into the life cycle of an application.
- Effectively use the other manuals in the Pathmaker manual set.

Audience

The *Introduction to Pathmaker* manual is designed for all Pathmaker users. The manual is especially informative for those who are preparing to use the product for the first time and for those who must decide if the Pathmaker product is an appropriate tool for their particular situation.

The *Introduction to Pathmaker* manual explains all of the major concepts and terminology that should be understood before the Pathmaker product is used or an application development effort that will use the product is planned.

Organization of This Manual

The *Introduction to Pathmaker* manual is divided into three major sections. These are:

- Section 1—Introduction to Pathmaker and Pathway
- Section 2—Pathmaker Applications
- Section 3—Using Pathmaker —An Overview

Section 1, “Introduction to Pathmaker and Pathway,” introduces some of the benefits and features of the Pathmaker and Pathway products (including some of the features new for this release). This section also defines Pathway terminology and components and describes the main purpose of several related Tandem products.

Section 2, “Pathmaker Applications,” explains the architecture of applications produced with the Pathmaker product. In addition, this section defines Pathmaker terminology, describes the types of applications that can be created using the Pathmaker product, and lists the types of requesters and services used for creating each type of application.

Section 3, “Using Pathmaker—An Overview,” provides a high-level introduction to the ways that the Pathmaker product can be used. This section illustrates the major components of the Pathmaker product, introduces several general approaches for successfully using the product, and suggests a formal Pathmaker application life cycle. Details about the tasks within the suggested life cycle are provided in the *Pathmaker Programming Guide*.

How to Use This Manual

If you are new to the Pathmaker product and new to Pathway, you should read this manual from cover to cover.

If you are new to the Pathmaker product, but are already familiar with Pathway and other related Tandem products, you can choose to quickly scan those portions of Section 1 that describe the products you already know. However, make sure you understand the concept of a service and how your choice of database product affects the creation of a Pathmaker application. Both of these topics are described in Section 1. You should thoroughly read sections 2 and 3 for an introduction to the Pathmaker product.

If you have used previous releases of the Pathmaker product, you already know most of the information presented in this manual. If so, it is recommended that you consult the New and Changed Information section of the *Pathmaker Programming Guide* for an overview of the new product features and for the locations in the Pathmaker manual set of detailed information about these features and capabilities.

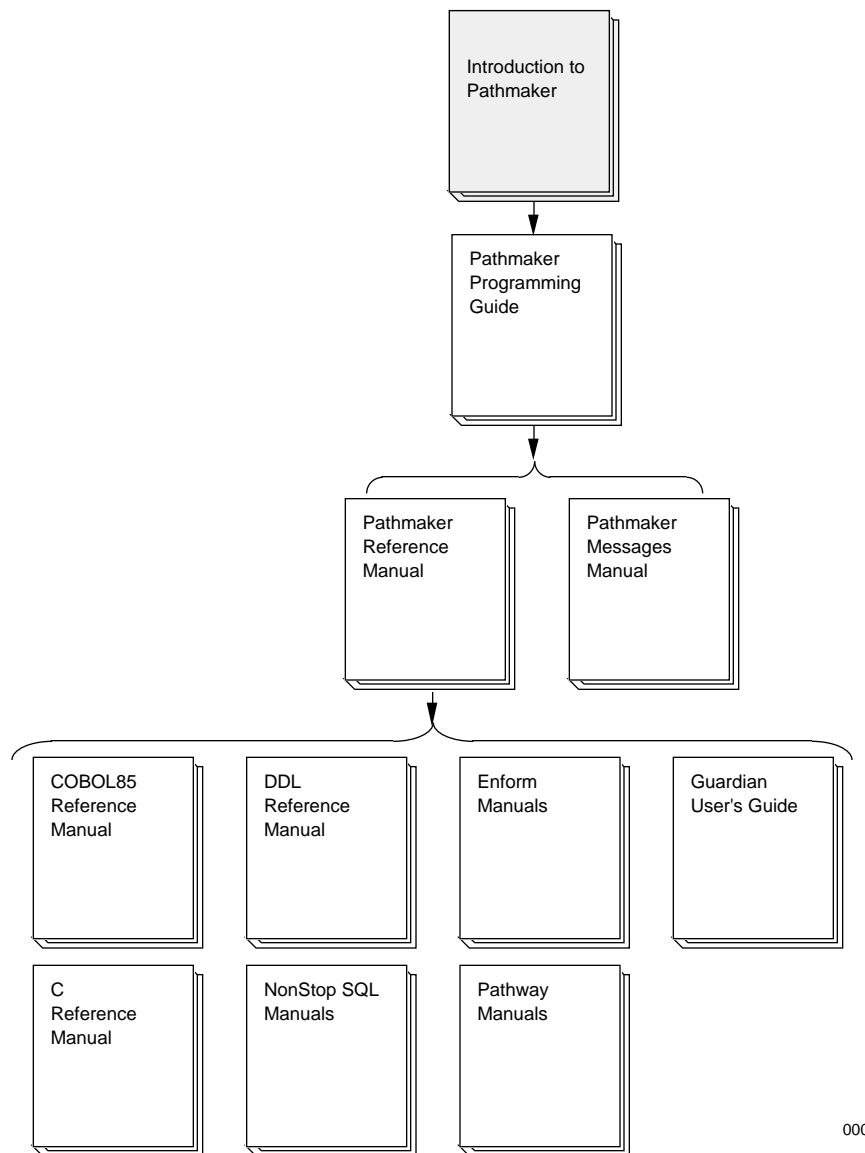
Throughout this manual, whenever an important term or phrase is used for the first time, it appears in bold-faced letters. Definitions of these important terms and phrases can be located in the glossary.

After you have become familiar with the concepts presented in this manual, you will be prepared to participate in the phases of a Pathmaker application development effort. When it is time to begin these phases, use the *Pathmaker Programming Guide*, in conjunction with the *Pathmaker Reference Manual*, to guide you through the tasks for each phase.

About the Pathmaker Manual Set

The Pathmaker manual set for Release 3 consists of four Pathmaker manuals. Figure 1 is a documentation map that shows how the Pathmaker manuals are related to each other and to other Tandem manuals. The map, read from the top down, indicates the order in which the manuals should be read. Manuals grouped by brackets are corequisites.

Figure 1. Documentation Map



000

The following chart explains the purpose of each Pathmaker Release 3 manual and its intended audience:

Pathmaker Manual	Purpose	Audience
<i>Introduction to Pathmaker</i>	Provides a comprehensive introduction to the Pathmaker product. This manual explains how an application created with the Pathmaker product looks and behaves, defines basic Pathmaker terminology, and suggests several general approaches for successfully using the product. In addition, basic information about Pathway and online transaction processing (OLTP) is included. The information in this manual provides the foundation needed to effectively use the other manuals in the Pathmaker manual set.	Anyone who needs an overview of the product, including all individuals involved in a Pathmaker application development effort.
<i>Pathmaker Programming Guide</i>	Provides a comprehensive task-oriented guide for the effective use of the Pathmaker product. This manual explains, in detail, how to use the Pathmaker full screen interface, how to design, develop, and maintain a Pathmaker application, and how to prepare for and manage a Pathmaker application development effort.	Individuals who are directly involved in a Pathmaker application development effort.
<i>Pathmaker Reference Manual</i>	Provides detailed descriptions of all Pathmaker screens, Pathmaker utilities, the Pathmaker Application Definition Language (PMADL), and the macro language. Pathmaker screen descriptions are arranged alphabetically by screen title and include details about screen fields and function keys. Descriptions of Pathmaker utilities, PMADL, and the macro language include syntax diagrams and usage information.	Pathmaker users who need detailed information about Pathmaker screens, Pathmaker utilities, PMADL, or the macro language.
<i>Pathmaker Messages Manual</i>	Provides an alphabetical list of the Pathmaker messages, explaining the cause of the message, the effect of the error, and suggestions for recovery.	Pathmaker users who need assistance in understanding a Pathmaker message.

Related Manuals The Pathmaker product interacts with several other Tandem products. In addition to the Pathmaker manuals, you should have these other Tandem manuals available when using the Pathmaker product:

- COBOL85 Reference Manual*
- C Reference Manual*
- Data Definition Language (DDL) Reference Manual*
- Enform Manuals**
 - Enform User's Guide*
 - Enform Reference Manual*
- Guardian User's Guide*
- NonStop SQL Manuals**
 - NonStop SQL Conversational Interface Manual*
 - NonStop SQL Installation and Version Management Manual*
 - NonStop SQL Database Planning and Administration Guide*
 - NonStop SQL Programming Manual for COBOL85*
 - NonStop SQL Programming Manual for C*
 - NonStop SQL Reference Manual*
- Pathway Manuals**
 - Pathway Application Programming Guide*
 - Pathway SCREEN COBOL Reference Manual*
 - Pathway System Management Guide*

Your Comments Are Invited After you have had a chance to use this manual, please take a moment to fill out the Reader Comment Card at the back and send it in. Your comments will be used to improve future editions of the *Introduction to Pathmaker* manual.

1 Introduction to Pathmaker and Pathway

The first step to thoroughly understanding the Pathmaker product, is to understand the purpose of the product and how it works with other Tandem products. This section addresses these issues by:

- Introducing some of the benefits and features of the Pathmaker product.
- Describing online transaction processing (OLTP) applications.
- Discussing some of the benefits and features of the Pathway product and introducing Pathway terminology.
- Describing the main purpose of several related Tandem products including NonStop SQL, Enscribe, Enscribe Data Definition Language (DDL), and the Transaction Monitoring Facility (TMF) product.

Overview of Pathmaker

The Pathmaker product is a tool that assists in the creation of Pathway requester-server applications for computer systems that use the Tandem NonStop Kernel. These applications require the full support of all Tandem features such as fault tolerance, linear expandability, security, data integrity, and high performance.

The Pathmaker product can improve the productivity of professional application developers by removing many of the labor-intensive coding tasks associated with creating an application. Applications created with the Pathmaker product can access data from NonStop SQL and Enscribe databases.

Benefits of Using Pathmaker

Some of the benefits gained by using the Pathmaker product include:

- Shorter development time for online transaction processing (OLTP) applications.
- Shorter maintenance time for OLTP applications.
- More application developers without Pathway experience are able to code Pathway programs because the Pathmaker product generates the program statements specific to Pathway applications.
- A unified application development environment.
- Improved project control because the components of an application are stored in a central location.
- Significant end user input allowed prior to coding.
- Support for the development of a wide range of applications.
- Production of Pathway applications that utilize the full power of the Tandem multiprocessor system architecture.

Note

The Pathmaker product is *not* a fourth generation language or tool for developing decision-support applications that do not require the Tandem fundamentals.

How Pathmaker is Used Application developers enter information about the application through the Pathmaker full screen interface, which provides a series of screen-based entry forms. (Information about the application can also be entered using the Pathmaker Application Definition Language (PMADL), which is a syntax-based interface.)

The Pathmaker product stores the information and later uses it to generate code for the application. In some cases, the Pathmaker product generates all of the code, and in other cases (depending upon the type of application) it only generates part of the code. As part of this code generation process, the Pathmaker product automatically creates a screen for the end user. (This screen can be easily modified to fit individual needs.)

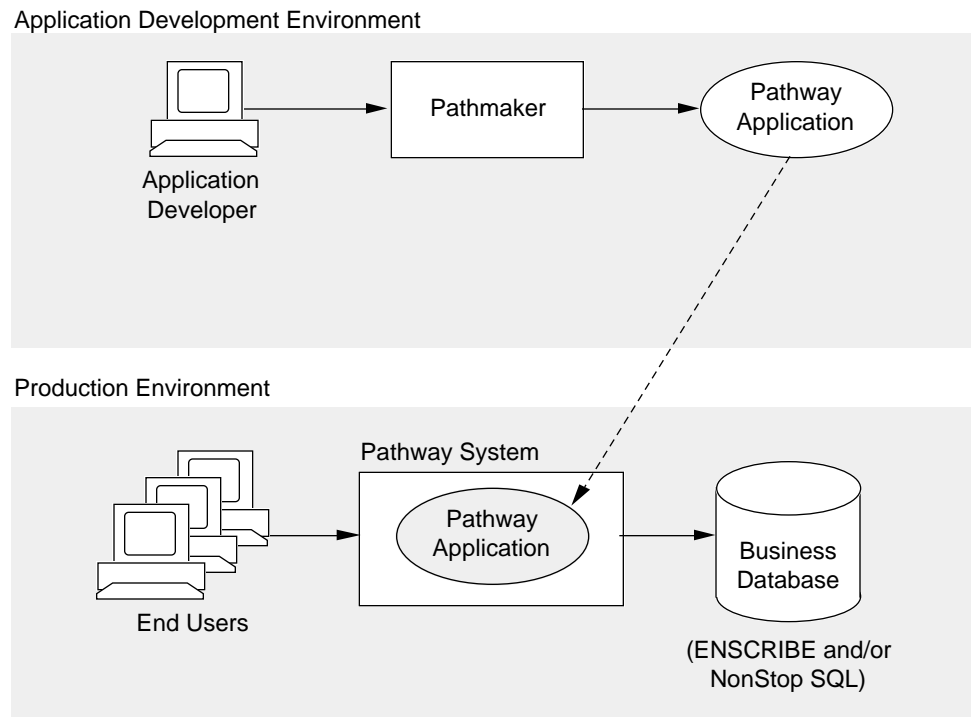
Using the simulation feature of the Pathmaker product, application screens and the navigation from one application screen to another can be previewed before a single line of code is written. This feature allows application designers and developers to simulate a proposed application for the end user and to obtain feedback and approval before investing a large amount of time developing the application.

In addition to simplifying the creation of Pathway applications, the Pathmaker product also provides support for the development of large applications by development teams that consist of many people. The product can be used to customize the Pathmaker development environment for a particular development effort. For example, it is possible to:

- Create shared code for the development team using the Pathmaker macro language.
- Print reports that provide vital information about components of the application being created.
- Modify the files (called skeletons) that are used by the Pathmaker product to generate the requester and server source code.

Figure 1-1 shows how the Pathmaker product fits in a development environment (top) and how the application it generates fits in a production environment (bottom).

Figure 1-1. The Pathmaker Product in the Development and the Production Environments



001

Online Transaction Processing (OLTP) Applications

It is essential to understand some basic information about the OLTP environment and about the Pathway product before learning more about using the Pathmaker product. This subsection explains the basic OLTP environment and some of its features and functions and describes Pathway and the various components it supports in the OLTP environment.

The term **online transaction processing (OLTP) application** has a specific definition within the Tandem environment.

An **application** is a set of programs designed to perform a group of related tasks, such as vehicle registration.

A **transaction** can be defined as a multistep operation that changes a database from one consistent state to another consistent state. A database is considered to be consistent when all parts of that database that represent a given entity are in agreement.

For example, consider the registration of a new motor vehicle. This transaction must perform several separate functions that affect different parts of the database, as follows:

- Add a row for the new vehicle to the Vehicle table.
- Add a row to the License Plate table for this vehicle.
- Add a row to the Owner table designating the owner of the vehicle.

In order for this transaction to complete successfully, each part of the transaction must be successfully processed. If any portion of the transaction fails, the database will no longer accurately reflect reality. For example, if during the registration of your new car, a row is added to the Vehicle table and another to the License Plate table, but a row is not added to the Owner table designating you as the owner, then the database's view and your view of reality will be radically different.

Online processing is the immediate handling of a transaction in a computer system. The tasks of data acquisition and information processing are not separate as they are in batch processing. Instead, these tasks are integrated together as one unified process to ensure that business information is current and correct at any point in time. The most significant characteristic of online processing is its immediacy.

Prior to the advent of computers, business was conducted in a transaction fashion and every transaction was accomplished from start to finish (that is, a bank deposit was initiated and then immediately completed). When computers were introduced, many transactions were gathered during the business day and later processed together. Usually, this processing was performed in batches processed at regular intervals, generally each evening after the close of business or perhaps twice a day. Unfortunately, the batch approach produced a lag between the time a business event occurred and the time it was recorded, making it impossible to get accurate, current information at any time during the business day.

By contrast, online transaction processing is an evolutionary change in automated information processing that allows data to be handled in a more timely way, allowing the database to accurately reflect the business at any point in time.

Providing flexibility and immediacy, online transaction processing allows a computer system to be integrated into the minute-by-minute operations of a business. In doing so, vital information is processed instantly as each transaction occurs.

This approach dramatically improves the timeliness of business information and enhances the decision-making capabilities of those who use and rely upon that information.

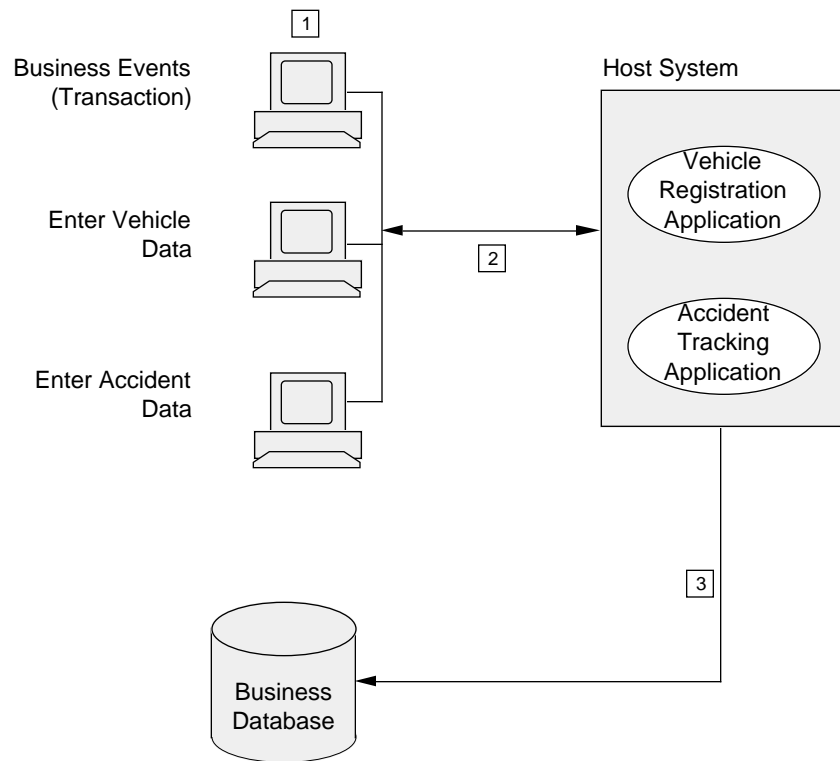
Functions Performed by OLTP Applications

All OLTP applications perform these basic functions:

- Display and accept transaction data.
- Perform input validation and allow the end user to correct input errors.
- Immediately update the database to reflect how the new transaction data affects the business.

Figures 1-2 and 1-3 illustrate these functions as they relate to two OLTP applications: one for vehicle registration and the other for accident tracking.

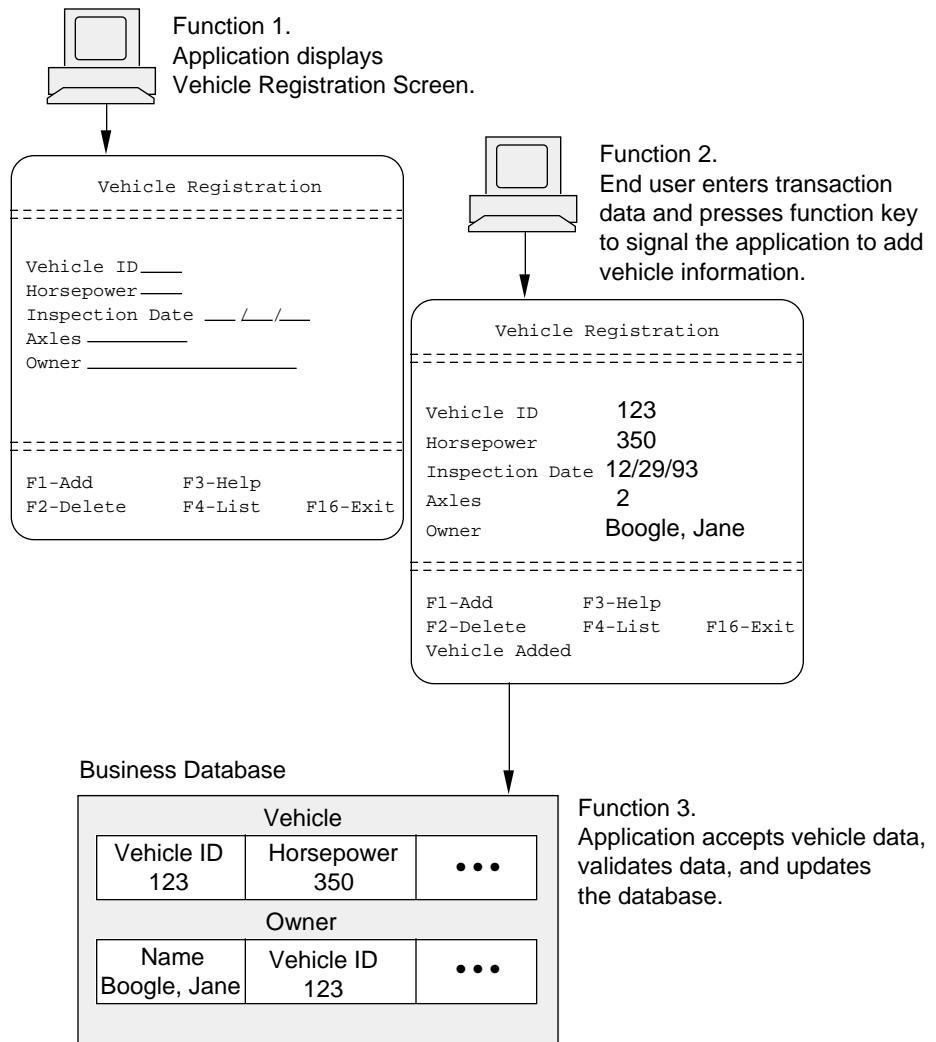
Figure 1-2. Online Transaction Processing Applications



Legend

- 1 End users enter transaction data.
- 2 Applications accept data from terminals and update database.
- 3 Changes to database occur immediately (online).

Figure 1-3. Functions of an OLTP Application



003

Requirements of Online Transaction Processing

Typically, online transaction processing applications must do the following:

- Handle multiple users performing related jobs such as simultaneous order entries.
- Handle concurrent requests to modify or retrieve the same information.
- Handle requests from different types of input devices, located at different geographic sites.
- Communicate with applications on other computers or share information stored on other computers.
- Perform a mix of operations consisting of some that must occur immediately and some that can occur periodically.
- Manage large amounts of geographically distributed information.

Pathway is designed to assist in the development and running of complex OLTP applications that meet all of these requirements. Because the function of the Pathmaker product is to help create Pathway applications, it is important to understand basic Pathway concepts and terminology. For this reason, Pathway is explained in greater detail in the next subsection.

Overview of Pathway Pathway is a Tandem product that supplies the tools and environment that assist in the creation and running of OLTP applications. Because the Pathmaker product's main purpose is to help create Pathway applications, a working knowledge of basic Pathway concepts and terminology is a prerequisite to understanding the Pathmaker product.

Benefits of Using Pathway There are two primary benefits gained by using the Pathway product. These benefits are as follows:

- The Pathway product allows application designers and developers to design and code their applications as if these programs were going to run in a simple, single-terminal environment.
- The Pathway product provides all of the complex support required to run these programs in a multiterminal, high-transaction volume environment.

If a business could automate its operations through batch processing or by using a single terminal to sequentially process a small number of online transactions, developing applications to support the business would be relatively simple.

Using this type of system, each application could:

- Process data from only one end user request at a time.
- Update the database without concern that other users may attempt to modify the same data at the same time.

Consider, however, the many demands and constraints of an environment where:

- Hundreds of terminals are connected to the computer.
- New terminals are added to the system periodically as the business grows.
- The transaction volume amounts to hundreds, or even thousands, of transactions each hour.
- Transactions can attempt to change the same data at the same time.

In these cases, OLTP applications are far more complex and sophisticated than applications developed for a single-terminal, low-transaction volume environment. The problem of developing and controlling these applications also becomes correspondingly more difficult—especially if a business requires its database to be correct and current at all times.

Pathway and the features provided by other Tandem products are designed to help solve these problems.

How Pathway is Used The Pathway product is used for two primary activities:

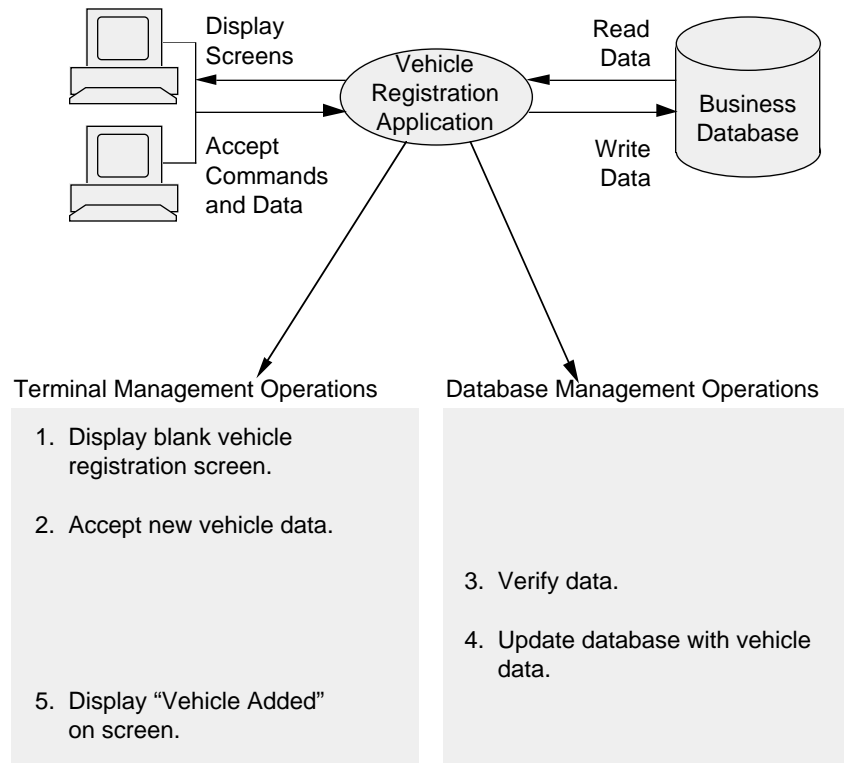
- Creating applications based on the requester-server model
- Running requester-server applications

These two activities, and the Pathway components that support them, are explained in detail in this subsection.

Creating Applications Without Pathway Normally, without Pathway, a **monolithic** application would be developed: one program that performs all terminal handling, communications, and database management operations that support transaction processing.

A monolithic application (Figure 1-4) can be very difficult to code and maintain. More importantly, this type of application can be especially difficult to change as a business grows and new functions are required by the application. In addition, the performance of a monolithic application often degrades when more functions and users are added.

Figure 1-4. Monolithic Application



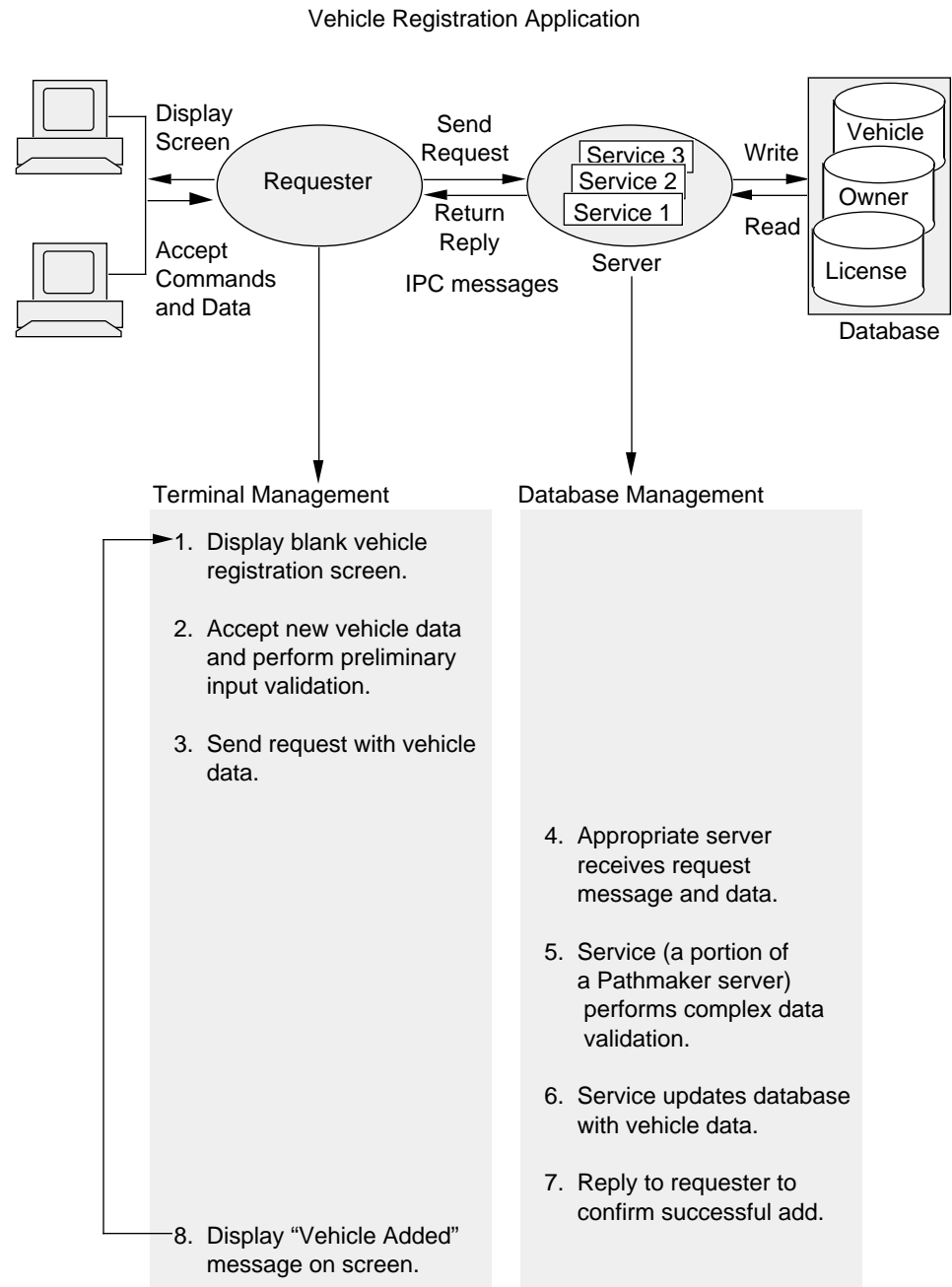
Creating Applications With Pathway

The OLTP cycle is basically a two-phase operation. First, input data is collected from the end user and then the collected data is used to read from or update the database.

When the transaction processing cycle is examined, it becomes clear that a more effective way to develop an application is possible. The approach used for Pathway applications involves dividing the application into two parts: one to perform data collection and the other to access and modify the data. The application programs that perform these two roles are known as the **requester** and the **server**.

Requesters and servers communicate with **interprocess communication (IPC) messages**. In general, Pathway requester-server applications are designed to place the majority of data processing work in the server because servers are generally faster than requesters. Figure 1-5 illustrates a Vehicle Registration application implemented with the Pathway requester-server approach.

Figure 1-5. Requester-Server Application



Requester Functions

A **requester** is an interactive program that displays and manages transaction data on the end user's terminal screen. In this capacity, a requester performs these functions:

- Provides the data entry screens for the application terminal.
- Accepts data entered at the terminal.
- Performs preliminary input validation.
- Builds a request message from the accepted data.
- Passes the request message to the appropriate server program to perform specific actions against the database, such as retrieving and updating data.
- Receives and evaluates the reply message from the server and creates screen displays for response to the end user.

Requesters are written in **SCREEN COBOL**, a high-level language similar to COBOL and designed specifically to create Pathway requester programs. Once coded, the requester is translated into pseudocode (using the SCOBOLX compiler) and is then stored in a SCREEN COBOL library.

Server Functions

A **server** is a database management program, written in a programming language such as COBOL85. In a Pathmaker application, a server is made up of one or more services. In fact, a server is a package of services that are grouped together according to certain aspects they have in common (such as processing time).

A server performs all message handling and provides a framework for one or more services that contain the code to manipulate data, perform calculations, validate input, and access the database.

Servers perform the following functions:

- Read messages from requesters and route them to the appropriate services.
- Relay the reply message created by a service back to the requester after the service has completed its work.

Note The Pathmaker product generates servers in either COBOL85 or C.

Benefits of Using Requesters and Servers

Terminal processing and database access are two operations performed by an OLTP application that have very little in common. Therefore, splitting the online application into a terminal handler and a database handler is useful because it is a logical way to modularize an application.

Also remember these points:

- Applications written in modular format are easier to write, control, and maintain.
- Requester-server applications make system tuning and load-balancing tasks easier for a multiprocessor Tandem system.

Service Functions

A **service** is a portion of a Pathmaker server that an application developer creates to perform a single, specific business transaction such as registering a new car. (Remember that a transaction is a multistep operation that changes a database from one consistent state to another consistent state.) Code for a service is written using a programming language such as COBOL85 or C. Services are contained in servers and cannot function outside of them.

After a request message is received by a service, the service can perform the following functions needed to complete all the steps of the transaction:

- Input validation
- Data manipulation
- Calculations
- Database access such as to read, write, or delete information from one or more related tables

When the service completes its work, it formats a reply message that will be sent back to the requester.

Benefits of Using Services

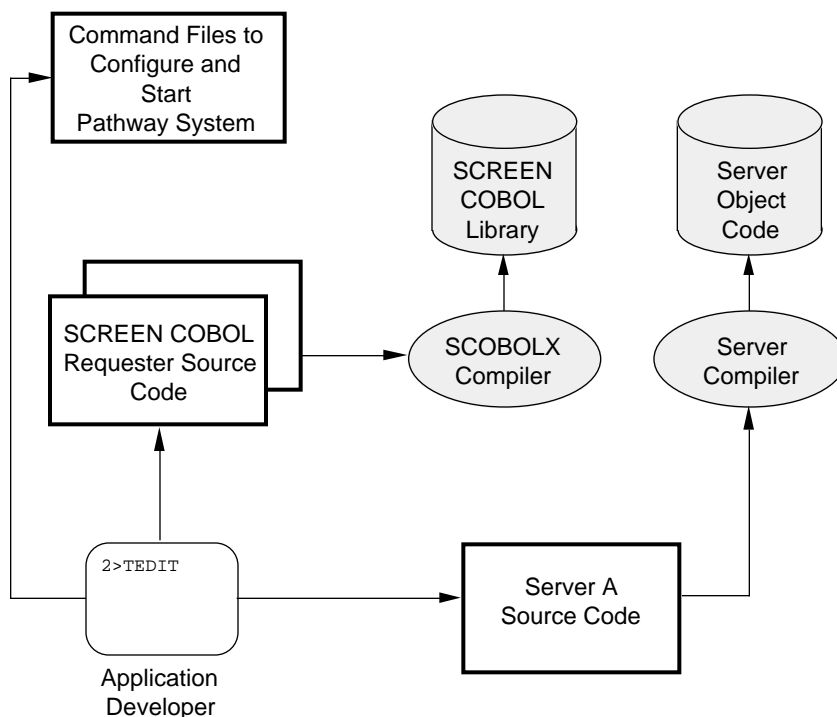
Splitting a database management program into even smaller units of code, with each performing a specific business transaction, adds an additional level of modularity to an application. In fact, this makes services easier to write and maintain because all of the code that deals with a transaction is contained in one place.

The decision as to which services are packaged together in a particular server affects the consistency of performance of an application. In general, services should be grouped together if they access a common table or tables and if they take approximately the same amount of time to run.

Because services are separate, they can be easily packaged and repackaged numerous times to achieve the best performance without application developers having to change source code.

Figure 1-6 illustrates how a requester-server application is created with Pathway (without using the Pathmaker product). The application developer uses TEDIT to create requester source code, server source code, and the files that will be used to configure and start the finished application. The source code is compiled and stored in the appropriate location.

Figure 1-6. Creating an OLTP Application With Pathway (Without the Pathmaker Product)



- Legend
- Application Developer Creates
 - Tandem Provides

006

Note The creation of an OLTP application using the Pathmaker product will be explained later in this section.

Communication With IPC Messages Requesters and servers (and the services the servers contain) communicate with each other through interprocess communication messages (commonly referred to as IPCs). There are two types of IPC messages: **request** and **reply**.

Request Messages

Request messages are sent *from* a requester and *received by* a particular service. When an end user presses a function key on the keyboard, the requester formats a message that identifies the service (the business task) associated with that function key, along with the data needed by the service to do its work. For example, if F4 is used to list a vehicle by vehicle ID number, the associated request message will identify the List By Vehicle ID Service and will contain the vehicle ID number the end user entered on the screen.

A request message is initiated in the SCREEN COBOL requester code with the SEND statement. Pathway then routes the request to a server containing the service. The server reads the message from a special file called \$RECEIVE and the appropriate service carries out the request.

Reply Messages

Reply messages are formatted *by* a service and *sent back to* a requester after the service has completed its work. A reply message is written to the \$RECEIVE file in the server code. Pathway then handles the routing of the reply message back to the appropriate requester.

The message will contain one of the following:

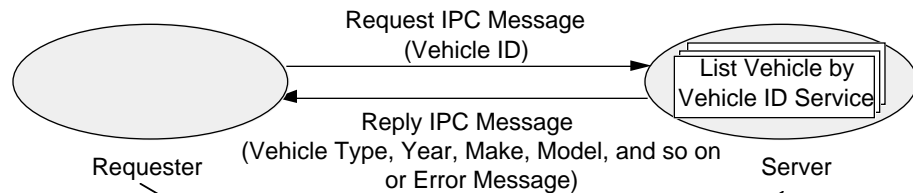
- Confirmation that the request was successful and the information that was requested, if any
- An error message if the request could not be fulfilled

The reply message can also contain logging and status information if application logging or performance measurement is being done.

Note The Pathmaker product generates code to handle the routing of all IPC messages for an application.

Figure 1-7 illustrates communication using IPC messages between a requester and a server containing a List Vehicle by Vehicle ID Service.

Figure 1-7. IPC Messages



1. End user fills in vehicle ID number.
2. End user presses function key for List Vehicle by Vehicle ID.
3. Request message is sent.

7. Requester displays screen.

4. Service reads database using vehicle ID number in request message.
5. If vehicle ID is found, reply message with all vehicle data is returned to requester, or if not found, reply message contains "Vehicle ID number not found" plus error codes.
6. Reply message is sent back to requester.

Running Applications With Pathway

The preceding discussion presented the definitions and functions of a requester, a service, and a server and explained how these components of a Pathway OLTP application are created (without the Pathmaker product). This subsection examines the various components of a running Pathway system.

Process

In a Tandem computer system, a **process** is essentially any running program. Each time a program is executed, the operating system creates a process. For instance, if a user runs two separate programs, two processes are created. Likewise, if two users run the same program concurrently, the operating system creates two processes. (An oval is usually used to symbolize a process.)

PATHMON

PATHMON is the supervising process of the Pathway environment; in this capacity, it manages all other processes in a running Pathway system. **PATHMON** is provided by Pathway and is started by the system operator.

PATHCOM

PATHCOM is the process that provides a command language interface, allowing the system operator to control and direct **PATHMON**. **PATHCOM** is provided by Pathway and is usually started by the system operator whenever it is needed.

SCREEN COBOL Library

The **SCREEN COBOL library** contains the requester programs after they are compiled by the **SCOBOLX** compiler provided with Pathway. This library consists of these files: **POBJDIR** and **POBJCOD** and, optionally, **POBJSYM**.

TCP (Terminal Control Process)

TCPs are the processes provided by Pathway that concurrently manage multiple terminals and transactions. **TCPs** interpret **SCREEN COBOL** programs (requesters) on individual end user terminals and handle message routing and preliminary editing of user input. The **TCP** locates and interprets the **SCREEN COBOL** programs in the **SCREEN COBOL** library.

Note In the figures found in this manual, the word requester enclosed in an oval represents **SCREEN COBOL** code being interpreted and controlled by a **TCP**.

The primary function of a **TCP** is to handle the complexities associated with multiterminal environments. Because the **TCP** handles these complexities, application developers can concentrate on the needs of the business and code the requesters to operate as though they were designed for a simple single-terminal environment. **TCPs** are started by the system operator.

Database

A **database** is an organized collection of information stored in a structured format on a disk. The database is created by an individual using NonStop SQL or Enscribe.

Service

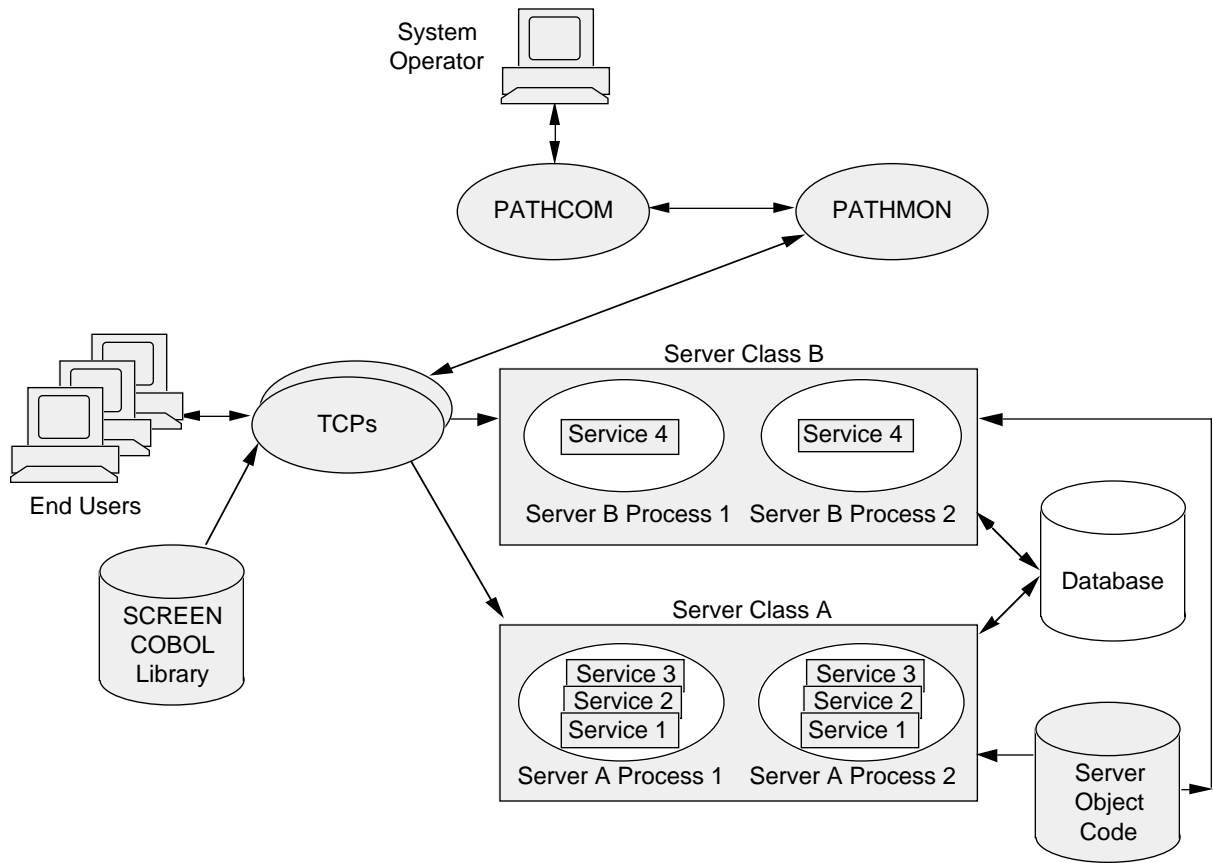
A **service** is a portion of database access code written in COBOL85 or C to perform a specific business task (transaction). Pathway applications created with the Pathmaker product use the concept of services.

Server Process


A **server process** is a running database management program that handles the IPC messages for one or more services. Pathway uses object code to create running server processes.

These eight components, which are illustrated in Figure 1-8, work together to run an OLTP application and to manage the complex tasks demanded in the world of multiple terminals and high-transaction volume.

Figure 1-8. Components of a Running Pathway System



Legend

 Tandem Provides

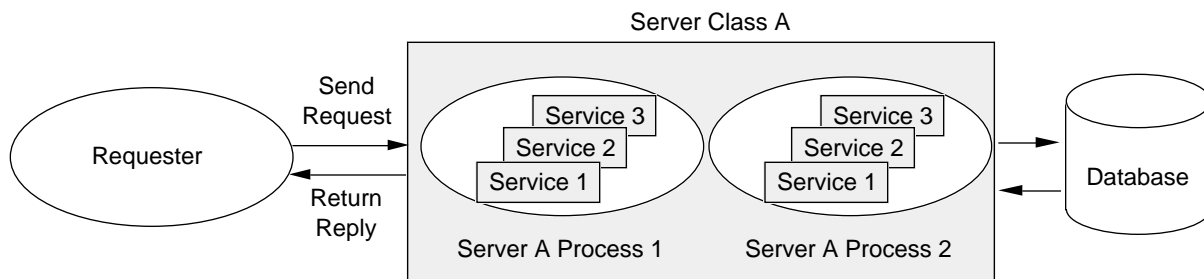
Server Class

A **server class** is a named family of identical server processes, each of which is created from the same object file on disk. Server classes are created by Pathway at run time.

The purpose of identical server processes is to maximize the performance of an application by fully utilizing the processing power of a Tandem system. Although several requesters can use a single server process, if too many requesters try to use a process concurrently, all the requests will be queued and delayed. To solve this problem, Pathway can start identical server processes and run them simultaneously to improve performance and throughput. Then, when the demand has subsided, Pathway can stop the unneeded processes.

In Figure 1-9, when the Pathway application that uses Server A is started, two Server A processes are created (and, therefore, two copies of the services it contains are created). This group of processes is referred to as Server Class A.

Figure 1-9. Server Class



009

If an end user presses a function key that initiates a business transaction (service) contained in Server A, Pathway selects which copy to use. For example, if Service 1 contains code to read one vehicle row by vehicle ID number, Pathway will select which Server A process, containing Service 1, will be used for this request. That copy of Service 1 will find the requested row.

Context-Free Services

As stated earlier, the ultimate destination of a Request IPC message is a service within a server. Several copies of a server process and, therefore, several copies of the services it contains can run concurrently. Each time a service is needed, any one of the available copies of the server that contains that service may be used (usually the least busy one).

If an end user is using a requester to send a series of requests that are related (such as listing, one by one, each person whose last name begins with the letter B), there is no guarantee that the same copy of the List Service will be used for each related request.

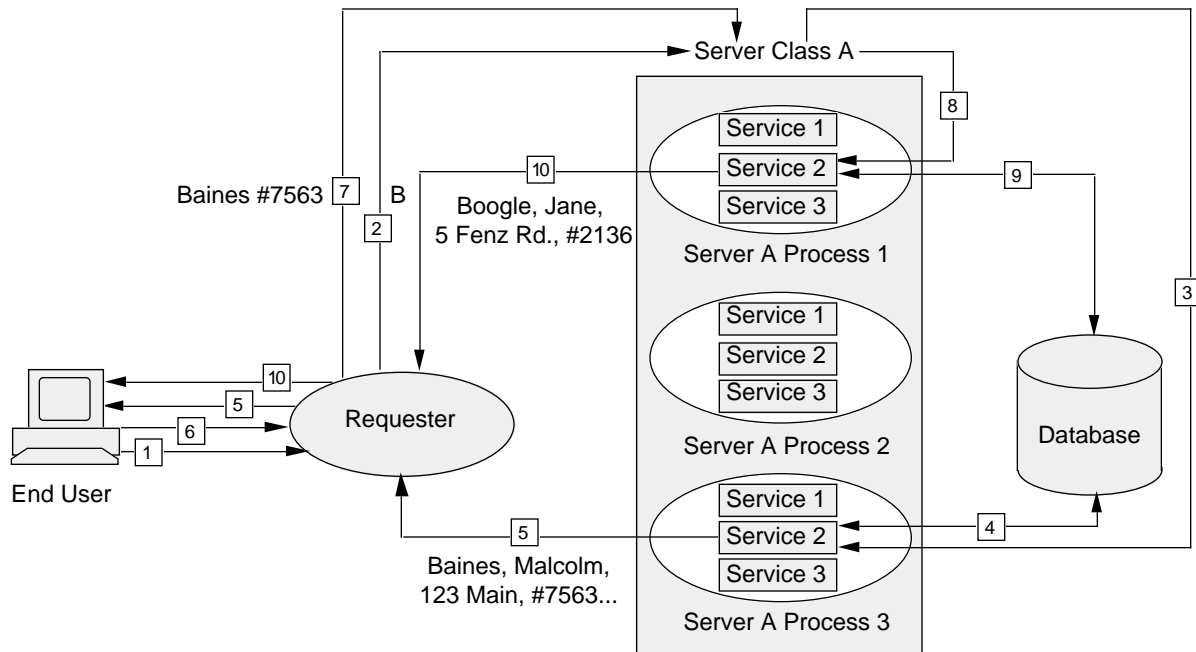
As a result, the requester must keep track of the most recent name beginning with the letter B displayed (BRADY, for example) and send that information with the next request. Having the name of the most recently displayed person will allow whichever copy of the service receives the next request to resume the list of persons correctly.

Services that *do not* save information about previous requests as a reference for subsequent requests are called **context-free**. All of the service code that application developers create must be written this way.

A service must not be required to maintain any request-dependent information between requests. (Request-dependent information can include file positioning, record counters, and so forth.)

Figure 1-10 illustrates a series of related transactions.

Figure 1-10. Context-Free Services



010

In this example, an end user wishes to list, one at a time, each person whose last name begins with the letter B. Service 2 contains the code to perform this task; it has been packaged with two other services (1 and 3) into Server A. Currently, three Server A processes are running. A step-by-step explanation of two of these related transactions follows.

First Transaction:

1. The end user types the letter B in the Last Name field on the screen and initiates the first request by pressing the function key associated with the List By Name Service.
2. The requester (working with the TCP) invokes Service 2 (the List By Name Service) by passing the letter B to the Server Class containing server processes that contain Service 2.
3. Pathway selects the server process best able to handle the request.
4. That server process (Process 3) directs its copy of Service 2 to read the database, using the letter B as the starting point.

5. The service returns BAINES, MALCOLM, 123 MAIN , ID # 7563 to the requester, which displays this information to the end user.

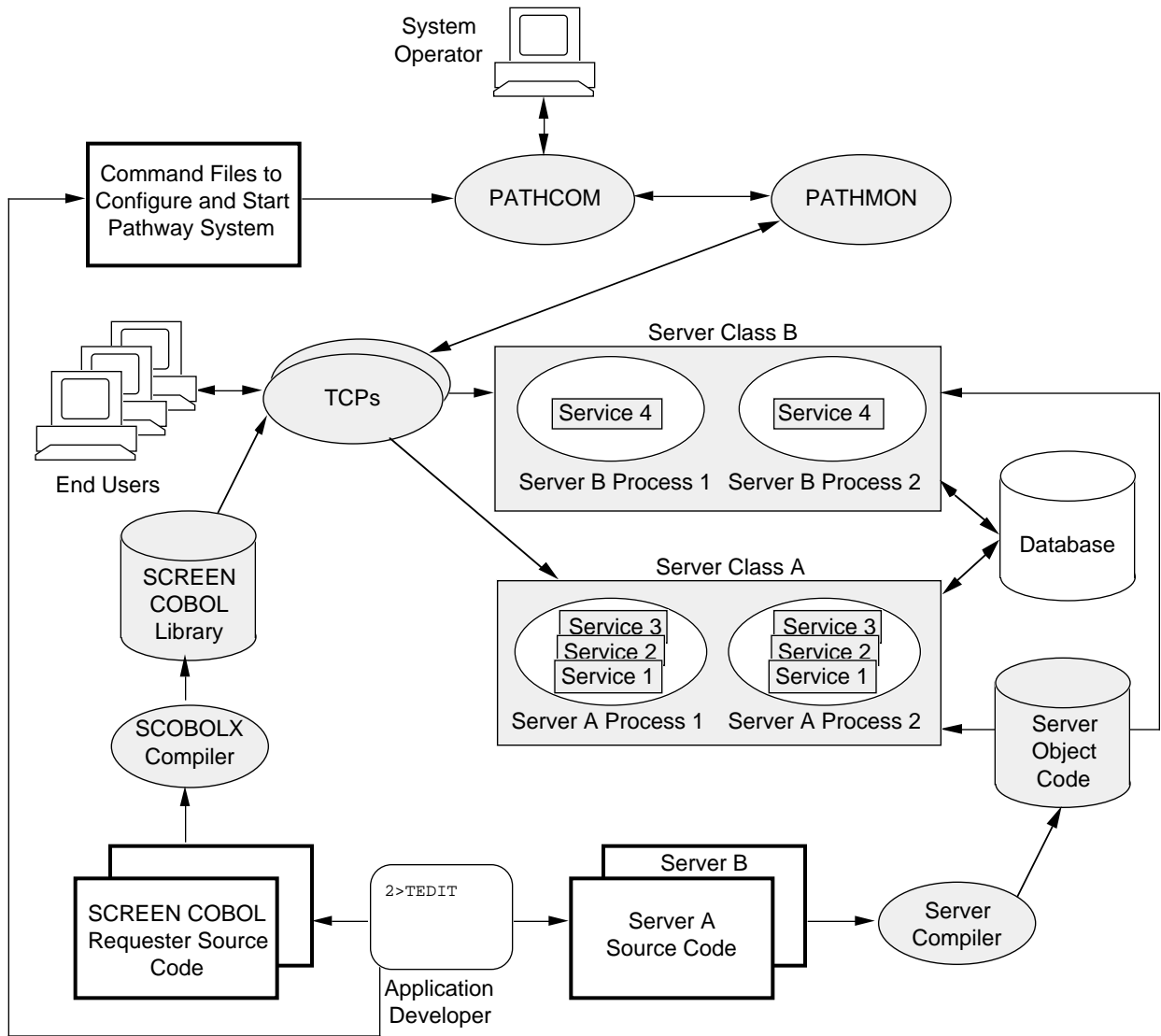
Second Transaction:

6. The end user continues the listing by pressing the appropriate function key again.
7. The requester (working with the TCP) again invokes Service 2, passing the name BAINES and the associated identification number back to the server class containing server processes that contain Service 2.
8. Pathway selects the server process best able to handle the request.
9. That server process (Process 1) directs its Service 2 to read the database using BAINES #7563 as its starting point.
10. The service returns BOOGLE, JANE, 5 FENZ RD., ID # 2136 to the requester, which displays this information to the end user.

This series of steps is continued for the remaining names.

Figure 1-11 illustrates the components provided by Tandem that are used to create and run a Pathway OLTP application.

Figure 1-11. Creating and Running an OLTP Application



Legend

- Application Developer Creates
- Tandem Provides

How Pathmaker Helps The Pathmaker product is a tool that helps create requester-server Pathway applications for computer systems that use the Tandem NonStop Kernel. Before the Pathmaker product was introduced, Pathway applications were often created manually. The requesters and servers were hand-coded (written from the beginning or modeled after an existing program). Even the files of Pathway commands that were needed to run the application had to be created manually. Not only was this process time-consuming and labor-intensive, it also made standards difficult to enforce and large projects (involving many application developers) difficult to track and control.

The Pathmaker product helps overcome these difficulties in several ways:

- It removes many of the labor-intensive coding tasks associated with creating the requesters, servers, and command files composing a Pathway application.
- It assists in customizing the development environment and in controlling and managing the development effort.
- It provides some application design assistance with the simulation feature.

This subsection discusses these features of the Pathmaker product in more detail.

Application Creation Assistance Using the Pathmaker full screen interface, application developers enter information about the requesters (and their associated end user screens), the services, and the servers that compose an application. This information is stored in a central location called a **catalog**.

Application developers also use TEDIT to create files containing the COBOL85 or C code that directs the services to perform database access, data manipulation, calculations, and input validation. For applications that access a NonStop SQL database, the Pathmaker product can be used to generate some SQL statements for accessing the database.

Using the information in the catalog and the TEDIT files created by the application developer, the Pathmaker product can generate the SCREEN COBOL requester source code (including the code describing the end user screen), the server source code, and the command files to configure and start the finished Pathway system for testing. (For some applications, the Pathmaker product provides 100 percent of the server source code; no creation of services is required.)

Application Creation Tasks

Some of the tasks an application developer performs to create a Pathway application with the Pathmaker product include:

- Describing the data that the application will use.
- Directing the Pathmaker product to generate SQL database access statements for inclusion in the server code.
- Writing the service code.
- Grouping services into servers.
- Specifying information about requesters being created and their screens, including the content of the help screens.
- Specifying the components of the IPC messages.
- Instructing the Pathmaker product to generate source code and then to compile it.
- Initiating the generation of Pathway command files used to test the completed application.

The Pathmaker product helps in the creation of a Pathway application by:

- Generating application source code in a uniform structure for all requesters and servers to help simplify maintenance and modification.
- Producing the program statements specific to Pathway applications (such as reading the \$RECEIVE file), making programming easier for anyone unfamiliar with Pathway.
- Optionally, producing SQL data manipulation statements for individual NonStop SQL tables.
- Providing a central location for most application information.
- Creating error-handling code for the most commonly encountered errors.
- Producing application code that contains no logic, syntax, or typographical errors.

Without the Pathmaker product, the SCREEN COBOL code, all of the server code, and the Pathway command files for an application must be created by an application developer using TEDIT.

Pathmaker Full Screen Interface Features

Other Pathmaker full screen interface features used during the development of an application include:

- Help screens that provide information about the purpose of the Pathmaker screens and the legal values that can be entered in the screen fields.
- Ability to incorporate additional code into the requesters and servers that the Pathmaker product generates.
- The screen painter, which shows changes to the layout of the end user screen as they are made.
- Simulation of the end user interface during development.
- Ability to easily provide the end users with help screens within the application being created.
- Ability to modify the end user help screens after the application has been placed into production.
- Generation of reports about the contents of the catalog.
- Interactive access to other Tandem products such as TEDIT, the File Utility Program (FUP), and so on.
- Creation of Pathway command files to configure and start the application for testing. (These command files can serve as a model for the production command files.)

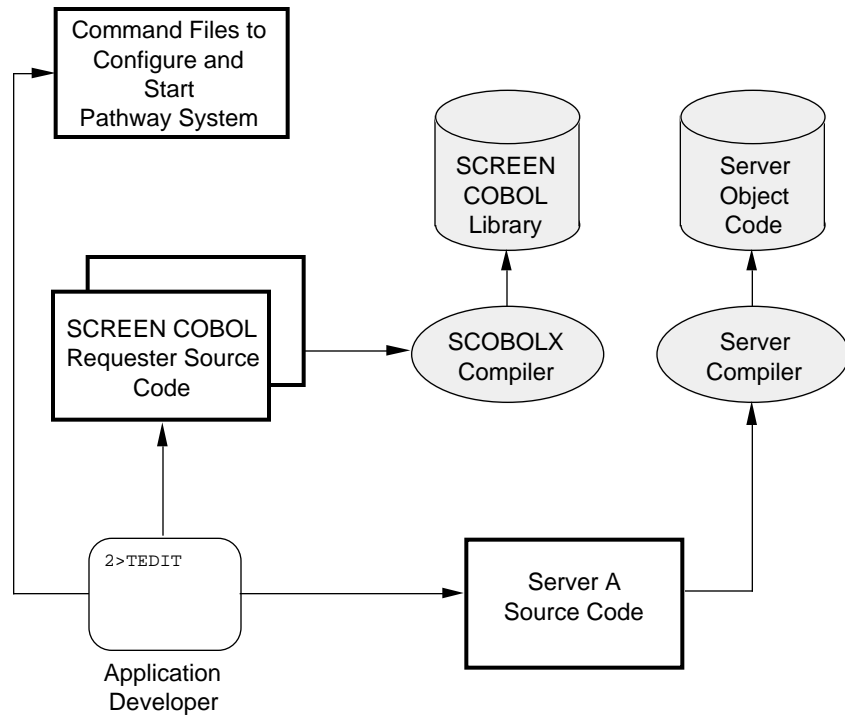
Pathmaker Application Definition Language (PMADL)

In addition to the full screen interface, the Pathmaker product also provides PMADL, which is a syntax-based interface. PMADL can help an application developer quickly make mass changes to the components of a Pathmaker application; it is also useful for quickly creating a Pathmaker application that is similar to an existing Pathmaker application.

Normally, PMADL is used to read the information about an application from an existing Pathmaker catalog and to convert this information into a text file that can be modified. The modified file is then used as input to PMADL, which creates entries in the target Pathmaker catalog using the modified information.

Figure 1-12 illustrates the process of creating a Pathway application with the Pathmaker product. Compare Figure 1-12 to Figure 1-6 (repeated from a previous page for your convenience), which shows creating a Pathway application without the Pathmaker product.

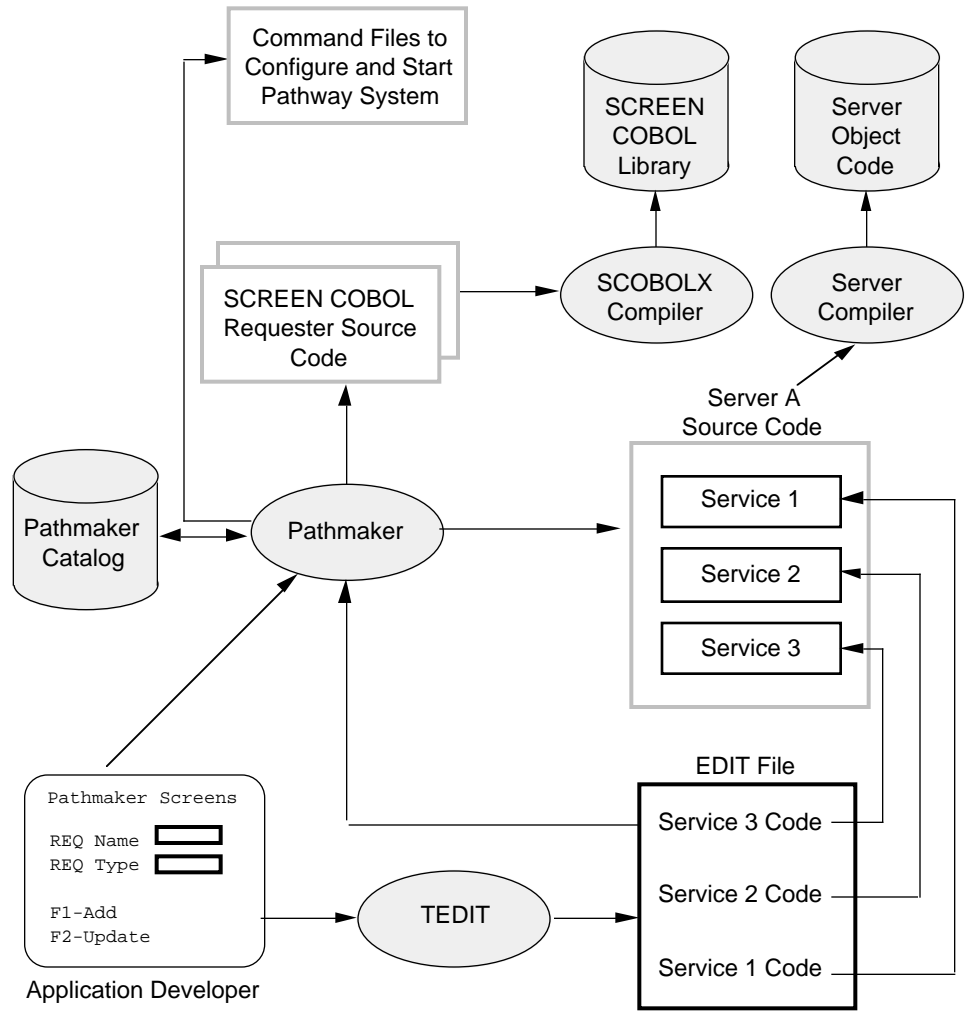
Figure 1-6. Creating an OLTP Application With Pathway (Without the Pathmaker Product)



Legend

- Application Developer Creates
- Tandem Provides

Figure 1-12. Creating an OLTP Application With the Pathmaker Product



Legend

- Application developer creates
- Tandem provides
- The Pathmaker product creates using information entered by application developers.

012

Management Assistance The Pathmaker product not only makes developing a Pathway application simpler, it also makes preparing, controlling, tracking, and managing an application development effort easier. This is especially important for large application development efforts involving many people.

The Pathmaker product features that provide assistance in these areas include:

- The Pathmaker catalog, which provides a central location for most application information.
- PMADL, which is an alternative to the full screen interface and is especially efficient for making mass changes to a Pathmaker application under development and for quickly creating an application modeled after an existing Pathmaker application.
- The Pathmaker macro language, which is supplied with and is unique to the Pathmaker product. The macro language simplifies the modification of the Pathmaker skeletons used to generate source code and can be used to create and register shared code macros for use by the development team.
- Hard copy reports, which provide information about Pathmaker applications under development.
- The ability to register requesters written outside of the Pathmaker product for access by Pathmaker requesters.
- The ability to register services and servers written outside of the Pathmaker product for access by Pathmaker requesters.

These Pathmaker product features are described in detail in the *Pathmaker Programming Guide*.

Application Design Assistance One way that the Pathmaker product helps with application design is by providing a simulation feature. Using the simulation feature, application screens and the navigation among them can be previewed before any code exists.

This feature allows application designers and developers to simulate a proposed application for the end user and to obtain feedback and approval before investing a large amount of time developing the application.

The Pathmaker product also helps with application design by allowing application developers to quickly create a prototype of an application and then later use that prototype as the basis for a production application.

Tandem Databases and TMF

This subsection discusses the Tandem database management products and the Transaction Monitoring Facility (TMF) product, which helps ensure the integrity of data. This discussion will focus on how these products interact with and affect the Pathmaker product.

Tandem Database Products

Tandem offers two database management products: the **NonStop SQL** product and **Enscribe**.

Both of these products support the creation and use of large databases capable of operating either as local or distributed systems.

A Pathmaker application is capable of accessing data in a Non Stop SQL or Enscribe database. A single server can even access data from both types of database.

NonStop SQL

The NonStop SQL product is the Tandem implementation of a distributed structured query language (SQL) Relational Database Management System. NonStop SQL is based on the American National Standards Institute (ANSI) specification for the SQL language.

SQL is a language used for:

- Database definition and creation
- Database access and data manipulation

Data in an SQL database is represented as two-dimensional structures called tables.

NonStop SQL data manipulation statements can operate on a single row or a set of rows. NonStop SQL statements can either be entered directly from a terminal or embedded in another language such as COBOL85. The Pathmaker product includes a facility that generates some NonStop SQL database access statements from information that application developers enter on certain Pathmaker screens.

Enscribe

Enscribe is part of the Encompass family of products offered by Tandem. These products made up the first commercial distributed database system sold by Tandem.

Enscribe is a database record management facility that executes statements in a server (such as READ or WRITE) to access a database and manipulate data. This powerful system provides one-at-a-time access to records in different types of commonly used files, including key-sequenced, entry-sequenced, and relative record files.

Enscribe DDL (Data Definition Language)

Enscribe DDL is one of the Tandem Encompass products. As its name implies, DDL is a language used to define data. The language consists of:

- Statements to define data
- Commands to control how those statements are compiled

DDL is also a program that compiles the data definitions and generates output from the compiled definitions. Depending on which compiler commands are entered, DDL can do the following:

- Build a dictionary from the definitions
- Translate record definitions into File Utility Program (FUP) commands
- Generate source code that describes data in a variety of programming languages

A DDL dictionary functions as a central repository for the data definitions of an application. The dictionary helps to maintain consistency so that the same data, regardless of where it is used, is described in the same way.

Although commonly used to describe data in a database, a DDL data dictionary can also be used to store information about other kinds of data. For example, application variables, constants, and screen-only fields can be defined with DDL and have their definitions maintained in a DDL dictionary.

Creating an Enscribe Database. Enscribe DDL can be used as a language and compiler to generate FUP commands that can be used to create an Enscribe database. When DDL is used for this purpose, the person developing the database must specify certain criteria about the database structure and characteristics. These criteria include:

- File structure
- Record structure
- Data-item characteristics

This information is coded using the data definition language and is usually stored in a TEDIT file called a **schema**. A schema is a formal representation of data composed of DDL statements that describe the data as fields, groups of fields, or records.

The schema is compiled and, if the DDL compiler is directed to generate FUP commands as output, the results can be used by FUP to generate the actual database.

Note The Pathmaker product provides a utility that can be used to create these FUP commands.

Updating the Enscribe DDL Dictionary. Another use of Enscribe DDL is to update the **DDL dictionary**. The DDL dictionary is an actual database that contains the formal description of the data in accordance with the data definitions in a schema.

The DDL dictionary contains detailed information about:

- Data in a database
- Other screen fields that an application might need

When the Pathmaker product is used to prepare an application development environment, one of the things the product does is to create a DDL dictionary. This DDL dictionary is an integral part of the Pathmaker catalog. The Pathmaker catalog contains the application information that application developers enter on Pathmaker screens in addition to the descriptions of the data in the DDL dictionary files.

Several Tandem products, including the Pathmaker product, recognize the format of these dictionary files. These products use the information contained in the DDL dictionary to perform various tasks, such as reporting, program generation, and the creation of SQL commands for table creation.

Keep in mind that a DDL dictionary is not actively integrated with the database files or source code it describes. Therefore, if a definition in the dictionary is changed, DDL does not automatically change the associated database or any source code. Conversely, if the database or the source code is directly changed, the associated dictionary is not affected unless someone directly implements the change.

Note For applications being developed with the Pathmaker product, DDL must always be accessed through the Pathmaker full screen interface. This version of DDL might differ from the version that can be accessed outside of the Pathmaker full screen interface.

Database Types and Pathmaker

Basically, all Pathmaker applications are built in the same way, regardless of whether they will be accessing data in a NonStop SQL database or in Enscribe files.

The area most affected by the choice of database is the creation and compilation of service and server code. The programming language statements used to read, write, and modify a NonStop SQL database are different from those used to perform the same types of functions on an Enscribe database. In addition, for applications that access a NonStop SQL database, the Pathmaker product can be used to generate SQL data manipulation statements from information application developers enter on Pathmaker screens.

Pathmaker and the DDL Dictionary

The Pathmaker product uses information in the DDL dictionary to generate requesters and servers and perform other tasks. A source schema describing all IPC messages and certain screen fields should be created, regardless of the database product (NonStop SQL or Enscribe) an application will be using. This schema should be compiled to update the DDL dictionary that is contained in the Pathmaker catalog.

For more information regarding Enscribe DDL, consult the *Data Definition Language (DDL) Reference Manual*.

NonStop SQL Table Registration

If a Pathmaker application is being created to access NonStop SQL tables (and views of tables), the Pathmaker product must be able to locate those tables. A data entry screen, called the SQL Table Registration screen, is used to provide the Pathmaker product with this location information. All tables used by a Pathmaker application must exist and be registered in the Pathmaker catalog before the application's requesters, services, or servers are created.

Variations in the procedures for using the Pathmaker product caused by the database product (NonStop SQL or Enscribe) selected will be noted in the applicable sections of the *Pathmaker Programming Guide*.

Table 1-1 compares some of the features and uses of the NonStop SQL and Enscribe products.

Table 1-1. Comparison of NonStop SQL and Enscribe Products (Page 1 of 2)

	NonStop SQL	Enscribe
How data is stored	Tables on disk.* (A special set of tables, called an SQL catalog, holds information describing these tables.) Database must exist before the Pathmaker product is used, although tables can be empty.	Files on disk (entry-sequenced, relative, key-sequenced). Database does not have to exist until Pathmaker application is tested.
How database is created	Database administrator: <ol style="list-style-type: none"> 1. Creates TEDIT file (schema) of DDL statements describing the database and other structures. 2. Compiles schema with DDL program (through Pathmaker interface) requesting DDL dictionary updates. 3. Uses NonStop SQL Conversational Interface (SQLCI) CONVERT utility to create TEDIT file that contains SQL commands created from information in DDL dictionary. 4. Obeys the file created in Step 3 through SQLCI interface to create tables, indexes, and so forth. 5. Loads data. OR <ol style="list-style-type: none"> 1. Creates TEDIT file of SQL statements for creating tables, views, indexes, and so forth. 2. Uses TEDIT file as input to NonStop SQL Conversational Interface (SQLCI) to create tables, indexes, and so forth. 3. Loads data. 	Database administrator: <ol style="list-style-type: none"> 1. Creates TEDIT file (schema) of DDL statements describing the database and other structures. 2. Compiles schema with DDL program (through Pathmaker interface) asking for FUP output. 3. DDL program generates FUP commands. 4. Executes FUP commands to create files. 5. Loads data.

* SQL tables are perceived by the end user to be two-dimensional tables, although they are actually stored in key-sequenced, entry-sequenced, or relative files.

Table 1-1. Comparison of NonStop SQL and Enscribe Products (Page 2 of 2)

	NonStop SQL	Enscribe
How database and other application structures (IPC messages and so forth) are described to Pathmaker	<p>Application developer:</p> <ol style="list-style-type: none"> 1. Registers SQL tables with Pathmaker screen. 2. Creates DDL schema describing data that is not part of the database (such as screen-only fields) then compiles with DDL program (through Pathmaker interface) requesting DDL dictionary updates. 	<p>Application developer:</p> <ol style="list-style-type: none"> 1. Compiles schema used to create the database with DDL program (through Pathmaker interface) requesting DDL dictionary updates.
How database is accessed by a server	<p>Application developers write SQL data manipulation statements, such as SELECT and INSERT, in the services. (Pathmaker can generate some of these statements.) Data manipulation statements cause NonStop SQL to perform processing on one or more rows from a table.</p>	<p>Application developers write program statements in the services (such as Read and Write) that cause Enscribe to perform record-at-a-time access to the database.</p>

Database Creation

As stated earlier, a production database is normally created and controlled by a specially authorized individual in a business, such as a database administrator.

For testing purposes, the Pathmaker product provides a utility program named **PMPROJECT** that can assist in creating Enscribe files. The NonStop SQL product must be used to create a test SQL database.

Before creating a database, consult the applicable Tandem reference manual for further details.

Overview of TMF

The Tandem Transaction Monitoring Facility (TMF) product is a part of the Encompass data management system. The TMF product simplifies the following tasks:

- Maintaining database consistency
- Protecting data from damage caused by system failures

TMF and Database Consistency

One of the steps that must be performed during database design is the establishment of certain criteria for the relationships between data.

For example, an account balance must always equal its credits minus its debits. When the data satisfies this criterion, it is considered consistent.

The TMF product helps maintain consistency by ensuring that concurrent transactions do not interfere with each other. The TMF product also ensures that any changes to the database resulting from a transaction become permanent only after the *entire* transaction is complete.

Remember, a transaction is a multistep operation that changes the database. While these changes transform the database from one consistent state to a new consistent state, they can make it inconsistent at points during the transformation.

For example, consider a banking transaction that transfers money from a savings account to a checking account. If a system failure occurs after the withdrawal from the savings account but before the deposit to the checking account, the database will be inconsistent.

To maintain consistency, the TMF product works with an application to ensure that one of the following is true:

- All of the changes a transaction attempts to execute are permanent.
- None of the changes is permanent.

In either case, the effect on the database is that consistency is preserved; that is, the database does not reflect any partial transactions.

How to Protect an Application Using TMF

In the requester code, the SCREEN COBOL verb **BEGIN-TRANSACTION** identifies the start of a TMF transaction. The **END-TRANSACTION** verb indicates that the transaction is complete and changes to the database should be committed; that is, the changes resulting from the transaction are permanent and will not be undone by the TMF product.

If a transaction is aborted for any reason *before* the END-TRANSACTION occurs or *before* its changes are committed, TMF will reverse the changes, thereby restoring the database to its original state before the transaction was started. The TMF product does this by using an audit trail that contains images of each record before and after any changes. The transaction can then be restarted from its beginning.

The Pathmaker product will automatically generate these statements in requesters when the application developer designates TMF protection for a particular service. Then, in the code written for that particular service, the application developer must set a flag to indicate whether the transaction should be aborted or committed. This flag will be sent back to the requester in the IPC reply message, and the requester will then direct the TMF product to act accordingly. (Keep in mind that in Pathmaker applications, a service is the equivalent of a transaction.)

In addition to protecting a business database, the TMF product can also be used to help protect other components of the development and production environments. Updates to the Pathmaker catalog and its data dictionary *must* be audited by the TMF product.

A NonStop SQL database must use the TMF product to protect the SQL catalog that describes that NonStop SQL database.

For more information about TMF, refer to the following TMF manuals: *Introduction to the Transaction Monitoring Facility (TMF)* and *Transaction Monitoring Facility (TMF) Reference Manual*.

2 Pathmaker Applications

An important part of understanding the Pathmaker product is knowing how an application produced with the product looks and behaves. This section helps you gain this knowledge by:

- Explaining the architecture of an application produced with the Pathmaker product.
- Describing how an end user operates an application produced with the Pathmaker product.
- Discussing ways that an application developer can customize a Pathmaker application.
- Describing the types of applications that can be created using the Pathmaker product.
- Introducing additional capabilities of the Pathmaker product.

The Architecture of a Pathmaker Application

A Pathmaker application has a specific architecture; that is, an application produced with the Pathmaker product has a distinctive appearance, way of behaving, and is constructed in a particular way. Every application created with the Pathmaker product conforms to the Pathmaker style.

The Pathmaker product can produce Pathway requester-server applications that can run in block mode on either the 6520/6530 family of Tandem terminals (including PCs with 6530 emulators) or on 3270-type terminals (including PCs with 3270 emulators). Pathmaker applications can also be produced for JET6530 terminals and 3270 terminals that support Kanji.

Screen Layout

The screen the Pathmaker product creates for most requesters is divided into four major areas:

- Title area
- Data field area
- Function key list
- Message area

The **title area** contains the requester title, which is a brief description that helps the end user identify the major purpose of a particular requester screen, and a page designator (for example, Page 1/3), which shows the number of the current page and the total number of pages for this screen. (Pages are explained in a subsequent discussion.) The title area is usually the first line of the screen.

The **data field area** is usually found between the two rows of equal signs on the screen. When the Pathmaker product creates a requester screen, it places all of the data fields and the headings that identify them within this area. Data fields are the places on the screen where data is entered by, or displayed to, the end user.

The **function key list** displays the function keys that are available and their associated action. (The list shown on the screen may include only a portion of the available keys.) The function key list is usually located near the bottom of the screen.

The **message area** is a reserved portion of the screen where messages and errors are displayed. The message area is usually below line 24 of the screen.

Multiple Pages–One Logical Screen

Each requester produced with the Pathmaker product has only one screen, which can consist of one or more pages. The Pathmaker product makes it easy to create a multiple-page screen within a requester by automatically creating a **paging area** and multiple **pages** when there are too many data fields to fit within the data field area. The paging area determines the size and placement of the requester's pages, which are implemented in the SCREEN COBOL code as overlays. The Pathmaker product creates the paging area between the two rows of equal signs on the screen. (Changes to the paging area and pages can be made with the screen painter.)

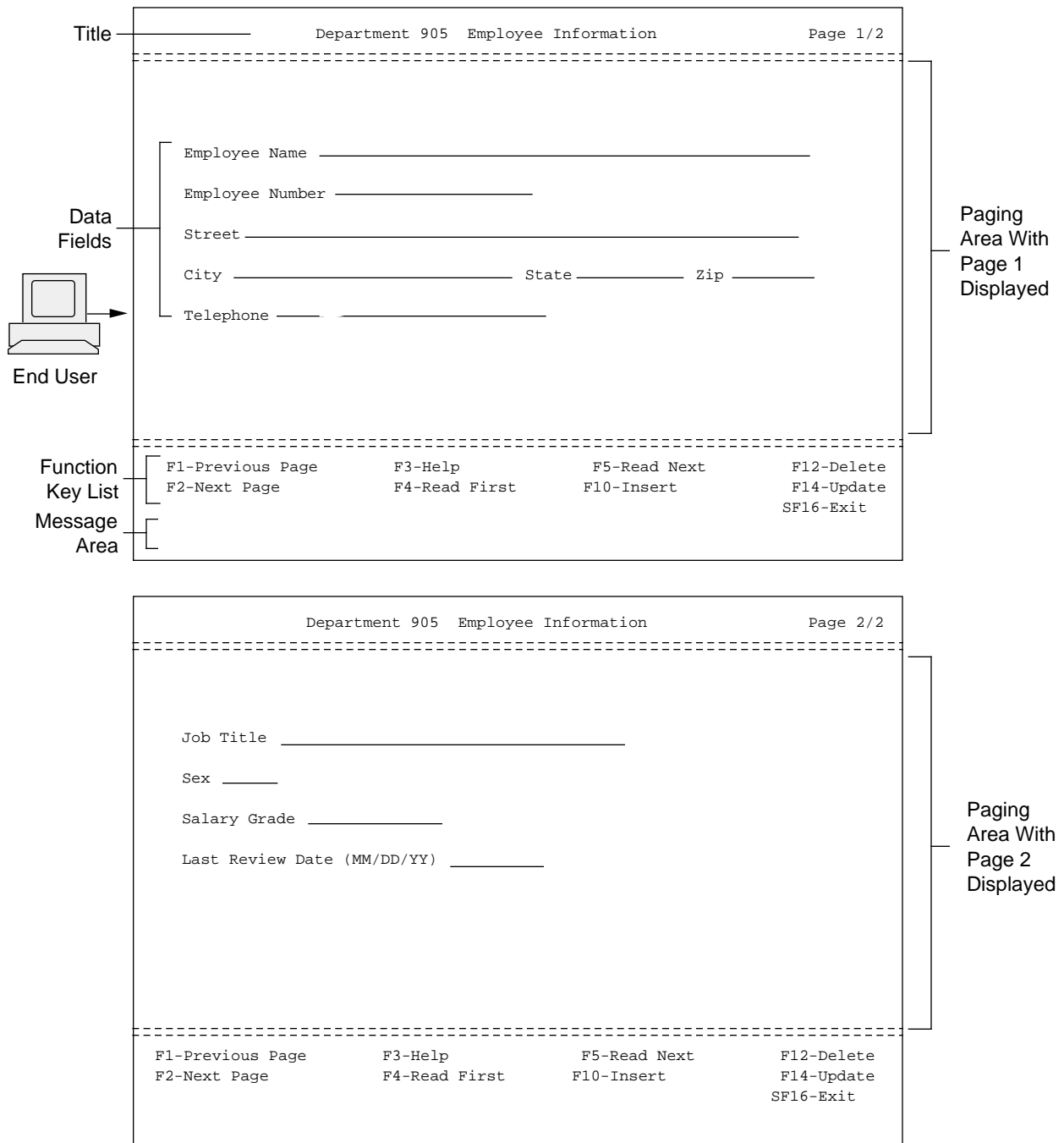
When a requester that has a multiple-page screen is used, usually the following is true:

- The end user has to ensure that all pages are reviewed for data content. (Pages not viewed contain default values.)
- Function keys have the same action regardless of the page the end user is viewing.
- Function keys are active on all pages.
- All pages are processed together, not separately.

(The application developer can write additional SCREEN COBOL code that can alter the way a requester with a multiple-page screen is used.)

Figure 2-1 shows the screen for a typical Pathmaker requester. The underlined areas represent unprotected fields where data is entered or displayed. This screen contains two pages. When any function key associated with a paging function is pressed (such as F2–Next Page), the title, function key list, and message area remain static. Only the paging area containing the data fields and the current page number in the top right-hand corner change.

Figure 2-1. Requester Screen



- Function Key Actions** Each Pathmaker requester screen can utilize many function keys. In a Pathmaker application, pressing a function key initiates one, and only one, action. The actions available are as follows:
- Processing:
 - Invoke a service to perform a specific task (usually after data has been entered on the screen).
 - Execute additional SCREEN COBOL requester code added by the application developer. (This additional code is placed in a special file called a requester copy library.)
 - Screen Navigation:
 - Go to another requester.
 - Display a different page of a screen.
 - Return to a previous requester.
 - Screen Support:
 - Print the information appearing on a screen.
 - Display the defaults for the data fields on this screen.
 - Display help text for this screen.
 - Recover the screen after a terminal failure.

Help for the End User Each Pathmaker requester provides help text to assist the end user in using the application.

The help text normally consists of information on the general purpose of the requester and on the purpose of and allowable values for data fields appearing on the requester screen. A complete list of the available function keys for the requester is also provided. The end user can quickly scroll through the help text on a help screen that is included with every application created with the Pathmaker product.

The end user displays the help screen by pressing the appropriate function key (F3 is the default.) The position of the cursor when the function key is pressed determines which portion of the help text is displayed first. When the end user returns from the help screen, the requester screen and any data previously entered is redisplayed.

Figure 2-2 shows an example of two pages of help text displayed on the help screen. The help screen is accessed from the application screen shown in Figure 2-3 when the end user positions the cursor next to the title of the screen (Department 905 Employee Information) and presses F3. In this example, the application developer provided help text for only two of the data fields and two of the function keys found on the requester screen.

Figure 2-2. Requester Help Screen

```

Help for: Department 905 Employee Information                               Page 1/2
-----
This screen is used to maintain a test database concerning employees in Dept. 905

EMPLOYEE NUMBER
This field is a unique four-digit value that identifies an employee.

SALARY GRADE
A two-digit number from 01 to 79. This field may be left blank.

F1-Previous Page           SF1-First Page
F2-Next Page              SF2-Last Page
F3-Help                   SF3-
F4-Read First             SF4-
F5-Read Next              SF5-Print Screen

-----
F1-Prev   F2-Next   F3-Help   F4-Down   F5-Up   F14-Print   F16-Return
SF1-First SF2-Last   SF14-Recover SF16-Exit
Help text listed

```

```

Help for: Department 905 Employee Information                               Page 2/2
-----

F6 -Read Approx.         SF6 -
F7 -Read Exact           SF7 -
F8 -Read Generic         SF8 -
F9 -                     SF9 -
F10-Insert               SF10-
F11-                     SF11-
F12-Delete               SF12-
F13-                     SF13-
F14-Update               SF14-
F15-Insert Defaults      SF15-Recover Screen
F16-Return               SF16-Exit

F14
Use this f-key after you've entered information on the screen.
F15
Use this f-key to display all available default values before you enter any
information.

-----
F1-Prev   F2-Next   F3-Help   F4-Down   F5-Up   F14-Print   F16-Return
SF1-First SF2-Last   SF14-Recover SF16-Exit
Help text listed

```

Data Field Attributes Each data field can have certain attributes that determine how the field will look and act.

These attributes can include:

- Length
- Data type
- Default value
- Allowable values (for example, must be “A,” “B,” or “C”)
- Video attributes (for example, bright, blinking, reversed)
- Protection from data entry
- Display format (picture)—for example, \$99.99
- Content of the heading describing the field
- Upshifting of entered data from lowercase to uppercase (for example, Abc to ABC)
- Help text

Operating a Pathmaker Application

When an end user operates an application created with the Pathmaker product, the application functions in certain predictable ways. These include the way that data entered by the end user is handled, the way default values are displayed, the way data and messages received from the servers are processed, and the way that the end user is allowed to navigate among screens in the application.

Entering Data

The data that the end user enters will be checked for proper length, data type, and allowable values when the end user presses a function key that activates a service or another requester. (Lowercase alphabetic data will be upshifted, if appropriate.)

If any fields contain data that does not conform to the appropriate attributes, those fields will be highlighted, one at a time, with an explanatory message appearing in the message area at the bottom of the screen.

The end user can then correct the data entered and resubmit the corrected data by pressing the appropriate function key again. The help screen can be used to obtain more information about the legal values for each field (if the application developer has provided this information).

If the data passes this preliminary input validation, the requester will format a message that contains the edited data and will invoke either a service or another requester.

Figure 2-3 shows an example of a requester screen containing a data entry error. The underlined areas represent unprotected fields where data is entered or displayed. Text in italics represents data entered by the end user. In this example, the employee number field must contain a valid number before an insert (F10) can be initiated.

Figure 2-3. Preliminary Input Validation

```

Department 905 Employee Information Page 1/2
-----
Employee Name Joe Blake
Employee Number                     
Street 123 Main
City Salt Lake City State Utah zip 84111
Telephone 555-2631
-----
F1-Previous Page      F3-Help      F5-Read Next      F12-Delete
F2-Next Page         F4-Read First F10-Insert        F14-Update
                                     SF16-Exit
001 Data Entry Error Found
    
```

023

Displaying Defaults If default values have been defined for any data field on the end user's screen, those values are displayed when the screen initially appears on the terminal or when the end user presses the appropriate function key (if available) prior to entering data. The purpose of default values is to make data entry quicker and easier for the end user by automatically filling in data entry fields with the most commonly used values or with blanks and zeros, as appropriate.

Receiving Data and Messages Whenever a requester receives data from a service it has invoked, each field is displayed according to the field's video attributes, picture, length, and data type. After a service has been invoked, a message (usually coded by the application developer) will be displayed in the message area at the bottom of the screen indicating either that the service was successful or that an error occurred (with a description of the error). The Pathmaker product highlights such errors by displaying the message in reverse video. In some cases, a message will be displayed to the end user after a requester returns from a lower level requester.

Figure 2-4 shows how a requester screen looks after an attempt is made to insert a row with an employee number that is already in the database. The underlined areas represent unprotected fields where data is entered or displayed. Text in italics represents data entered by the end user.

Figure 2-4. Error Message From a Service

Department 905 Employee Information		Page 1/2	
Employee Name	<u>Joe Blake</u>		
Employee Number	<u>4375</u>		
Street	<u>123 Main</u>		
City	<u>Salt Lake City</u>	State	<u>Utah</u> Zip <u>84111</u>
Telephone	<u>555-2631</u>		
F1-Previous Page	F3-Help	F5-Read Next	F12-Delete
F2-Next Page	F4-Read First	F10-Insert	F14-Update
			SF16-Exit
Insert Unsuccessful - Employee Number already in use.			

024

Screen Navigation Within a Pathmaker Application

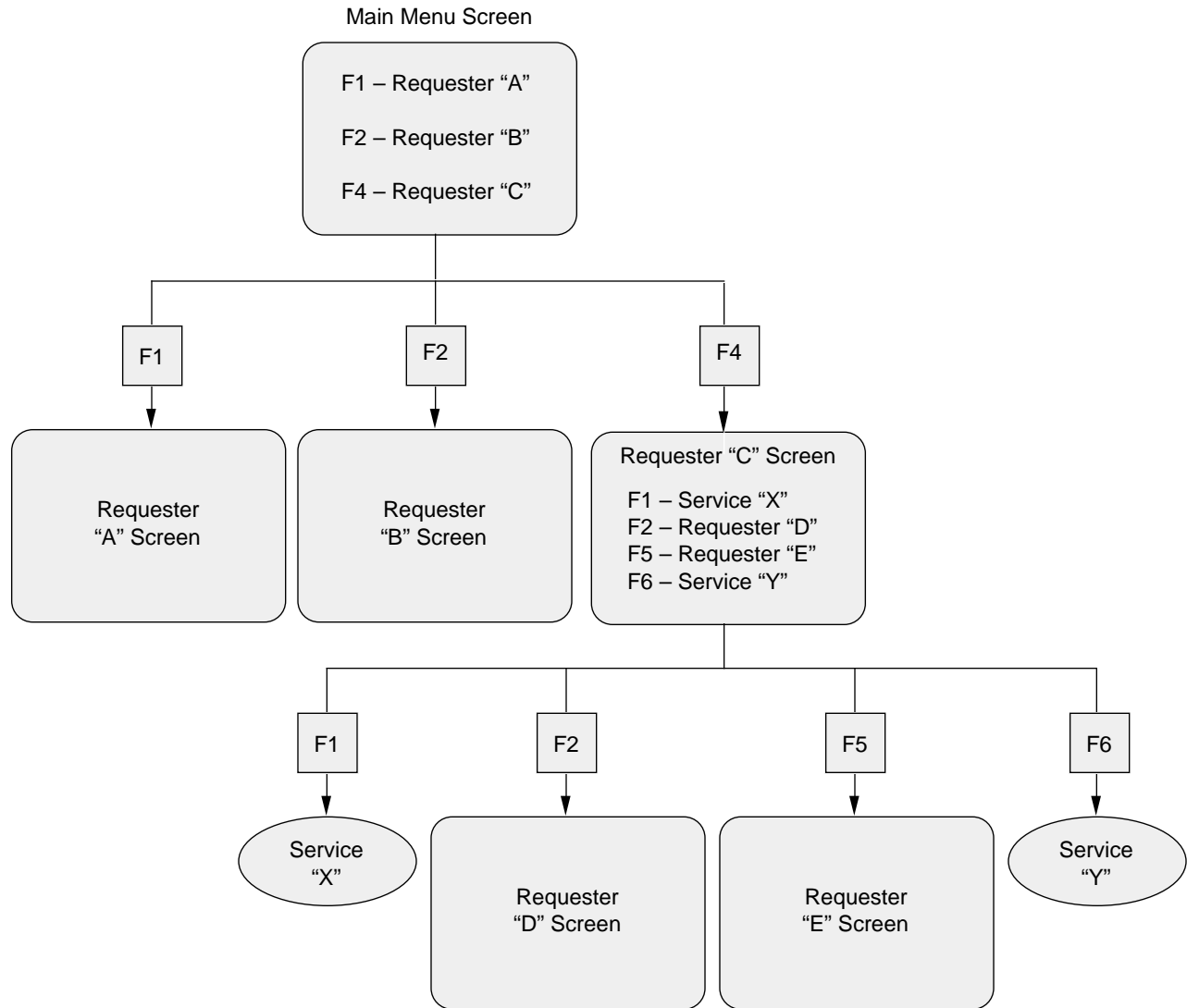
One important aspect of any application is how it allows the end user to navigate among screens. The Pathmaker product produces applications that have a **hierarchical** structure.

In an application with a hierarchical structure, the application's requesters are arranged in a fixed hierarchy, sometimes referred to as a tree. Typically, this hierarchy has a main menu requester at the top level with submenu requesters below leading to detail requesters (used for data entry or data retrieval) at the lowest levels.

When using an application that has a hierarchical structure, the end user navigates between requester screens and related business tasks by pressing a function key once for each step up or down in the hierarchy.

Figure 2-5 illustrates an application with a hierarchical structure. In this example, to get to Requester "E" from the Main Menu, the end user first presses F4 to go to Requester "C" and then, from that screen, presses F5 to go to Requester "E."

Figure 2-5. Hierarchical Pathmaker Application



Customizing Pathmaker Applications

Although the Pathmaker product creates an application with a certain predetermined architecture, it provides many ways to customize that application to meet end user specifications.

Screen Layout

For each requester, the Pathmaker product can create the initial layout for the end user's screen based on the type of requester chosen and on the attributes specified on Pathmaker screens. This initial layout is referred to as the **default screen**.

The attributes that can be specified on Pathmaker screens that affect the default screen layout include:

- Screen title text
- Function keys and their associated actions
- Data fields and their order
- Data field attributes
- End user's terminal type

Note

Certain attributes associated with a data field in the DDL Dictionary can also influence the appearance of the default screen.

Screen Painter

The default screen that the Pathmaker product creates from the preceding list of attributes can often be used without further modification.

To further customize the screen, application developers can use the Pathmaker **screen painter**. Reasons for modifying the default screen might range from making the application easier to use to conforming to the end user's standards.

Screen painter is a versatile tool that can help perform many screen customizing functions, including:

- Adding, rearranging, or deleting data fields on the default screen
- Adding decorations or custom text to the default screen
- Changing the video attributes of a data field
- Modifying the screen's paging (overlay) area

Help Text When application developers create requesters with the Pathmaker product, one of their tasks is to write and enter the text that will appear on the end user's help screen. The Pathmaker product handles all of the complexities of storing and displaying the information to the end user in the appropriate format. The Pathmaker product even provides a utility (HELPUTIL) that allows help text to be easily changed after the application has been moved to production.

Messages Application developers can provide the end user with meaningful messages from a service by including the messages in the service code they write. These messages could explain an error, suggest a next step, or simply indicate a successful action.

Advanced Customization Techniques The Pathmaker product offers additional ways to customize applications. These customization techniques allow the greatest amount of flexibility to application developers creating sophisticated applications. To use these methods effectively, experience in coding Pathway applications is very helpful.

These techniques are discussed in detail in the *Pathmaker Programming Guide* and include:

- Incorporating SCREEN COBOL code written by application developers into specific locations in the requester code the Pathmaker product generates.
- Changing global application attributes such as the advisory message attributes and terminal time-out limits.
- Defining data for use with the Pathmaker product.
- Registering applications developed outside the Pathmaker product.
- Creating code macros that can be shared.
- Modifying the skeleton files that the Pathmaker product uses to generate source code for requesters and servers.

Types of Pathmaker Applications

There are two types of applications that can be produced with the Pathmaker product:

- Database (DB) requester applications** (sometimes referred to as simple applications because they are simple to create)
- Custom applications**

Although Pathmaker applications used in production are often combinations of these two types, this manual and the other manuals in the Pathmaker Release 3 library address DB requester applications and custom applications as separate entities. This approach is meant to make learning how to use the Pathmaker product easier.

DB Requester Applications

DB requester applications are data management applications that must be developed quickly but are not usually used for high volume OLTP applications that are critical to the operation of the business. These applications can be used to add, maintain, or retrieve information from one or more related tables in a database. DB requester applications cannot perform calculations or data manipulations or, more importantly, perform complex integrity checks on the database. A DB requester application can be created so quickly that it is even practical to use for an application that will be used only once.

Typical uses for DB requester applications include:

- Entering test data into a database
- Listing information from the database
- Maintaining infrequently updated production files
- Creating a working **prototype** of a custom application

Note A prototype is a simplified working model on which a custom application will be based. A prototype is not intended for use in production.

Operating a DB Requester Application

A DB requester application uses **keys** to locate specific rows (records) in a database.

Three categories of keys exist for NonStop SQL tables:

- Primary keys
- Indexes
- SYSKEYs

Three categories of keys exist for Enscribe files:

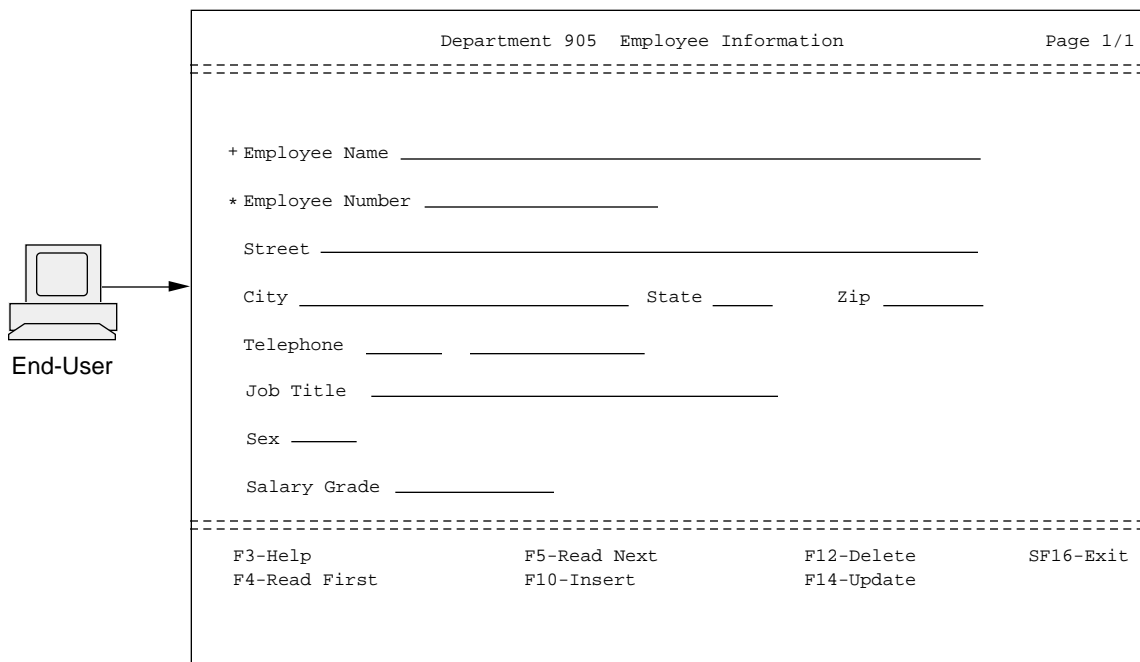
- Primary keys
- Alternate keys
- Courtesy keys

DB requester applications generally use primary keys and indexes (alternate keys) to locate information in the database. Courtesy keys cannot be used, and SYSKEYs are not generally used.

As with most other end user screens created with the Pathmaker product, a DB requester application screen contains a title area, a data field area, a function key list, and a message area. If an asterisk (*) appears to the left of a data field heading on the screen, it identifies a primary key field. If a plus sign (+) appears to the left of the data field heading on the screen, it identifies an index or alternate key field. The end user indicates to which row(s) an operation applies by entering data into these key fields on the screen.

Figure 2-6 provides an example of a screen from a DB requester application used to maintain information about fictitious employees in a test database.

Figure 2-6. DB Requester Application Screen



The diagram shows an end-user, represented by a computer icon and the label "End-User", interacting with a terminal screen. The screen displays the following information:

```
Department 905 Employee Information Page 1/1
-----
+ Employee Name _____
* Employee Number _____
Street _____
City _____ State ____ Zip _____
Telephone _____
Job Title _____
Sex _____
Salary Grade _____
-----
F3-Help           F5-Read Next     F12-Delete       SF16-Exit
F4-Read First    F10-Insert       F14-Update
```

026

Components of a DB Requester Application

A DB requester application is composed of:

- One or more **DB requesters** that application developers create using the Pathmaker product
- The **standard services and servers** provided by the Pathmaker product
- One or more **menu requesters** that application developers create using the Pathmaker product (optional)

It is possible to create a DB requester application that accesses:

- Only one table or file (a **single-file DB requester**)
- Multiple related tables or files (a **multifile DB requester**)

NonStop SQL tables and Enscribe files, however, cannot both be accessed by a single DB requester.

DB Requester. A database (DB) requester displays a screen that end users can use to enter, retrieve, or modify data in a database. Only a DB requester can be used to invoke the standard services provided with the Pathmaker product.

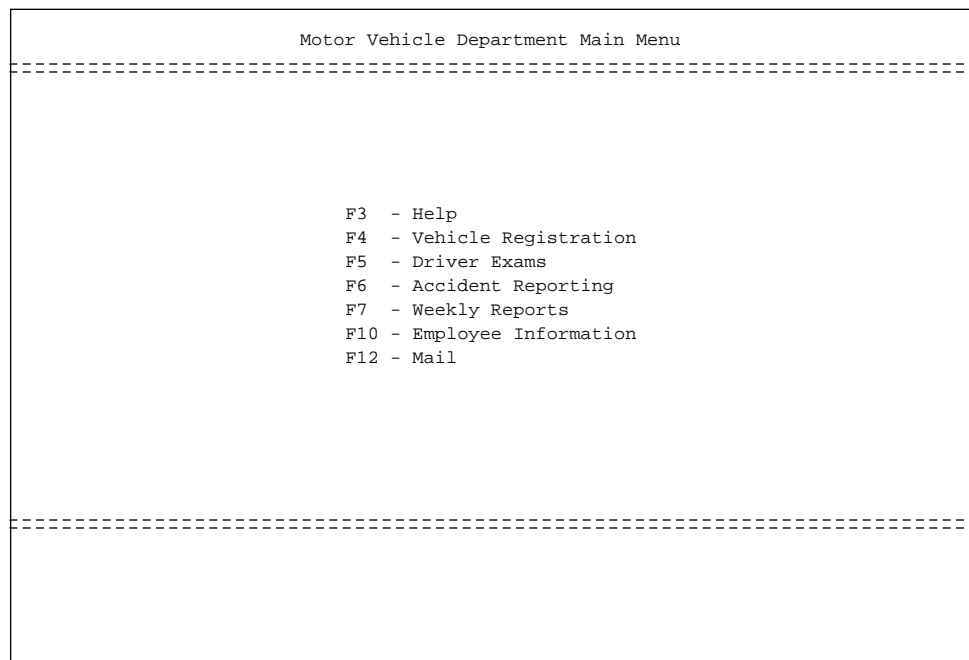
Note DB requesters *cannot* use custom or registered services.

Standard Service. Standard services perform common database functions such as inserts, deletes, and updates. These services are packaged into two standard servers provided by the Pathmaker product—one for use with NonStop SQL databases and one for Enscribe files.

Menu Requester. A menu requester displays a screen to the end user composed of a list of function keys that are used solely to navigate to other requesters. Menu requesters can be used with both DB requester applications and custom applications. A Menu requester cannot invoke a service.

Figure 2-7 shows an example of a menu screen.

Figure 2-7. A Menu Screen



027

Custom Applications The second type of application that can be produced with the Pathmaker product is a custom application. Custom applications are created to handle the crucial data processing needs of a business, including high volume OLTP applications.

Examples of custom applications include:

- A payroll processing application
- An order entry application
- A vehicle registration application
- An airline reservations application

Operating a Custom Application

The screen for a custom application is similar in appearance to the screen for a DB requester application; however, the actions that the end user initiates by pressing the function keys of a custom application can be far more sophisticated than the database access tasks available in a DB requester application.

In addition to initiating database access, the end user of a custom application can use the function keys to initiate the execution of tasks performed by custom services such as:

- Enforcing integrity rules on data
- Complex editing
- Data manipulations
- Calculations

Custom applications are the most frequently used applications in a typical business environment. A custom application that is properly designed and implemented can provide an efficient way for the end user to accomplish crucial business tasks and, at the same time, can protect the integrity of the data the business relies upon.

Figure 2-8 is an example of a screen for a custom application used to maintain vehicle information.

Figure 2-8. Custom Application Screen

Vehicle Detail Page 1/1

Vehicle ID _____

Make _____ Model _____ Year _____

Engine Number _____

Registration Date _____ Sales Price _____

Tax _____

F3-Help	F5-Calculate Tax	F11-Owner List	F14-Update
F4-List	F10-Add	F12-Delete	SF16-Exit

028

A custom application was created for this business task to provide more sophisticated processing and database integrity than is available with a DB requester application. For example, pressing F5 (Calculate Tax) will calculate and display the tax on the sales price entered. Pressing F12 (Delete) will only delete the vehicle after all references to it are removed from other places in the database.

Components of a Custom Application

A custom Pathmaker application is usually composed of:

- One or more menu requesters (optional)
- One or more transaction (**TRNS**) requesters
- One or more **custom services** (packaged into custom servers)

All of these components are created by application developers using the Pathmaker product.

Menu Requester. The menu requesters used as components of custom Pathmaker applications are the same as those described for DB requester applications.

Transaction (TRNS) Requester. A transaction (TRNS) requester displays a screen similar in appearance to the screen for a DB requester. The important difference is that an end user can only invoke a custom service by using a transaction requester.

Note Transaction requesters *cannot* use the standard services.

Custom Service. A custom service can perform database access, data manipulation, editing, and calculations. The Pathmaker product provides a framework for each custom service. Application developers must supply the COBOL85 or C code that does the work. For custom services that access data in NonStop SQL tables, the Pathmaker product can be used to generate some SQL database access statements.

COBOL85 Custom Service. Using special EDIT files, application developers supply the COBOL85 code for each service for the:

- Working-Storage Section
- Extended-Storage Section
- Procedure Division

Application developers use the Pathmaker product to package custom COBOL85 services into custom COBOL85 servers and then generate and compile those servers. (When a Pathmaker server is generated, each service in it is included as a separate COBOL85 subprogram.)

When a COBOL85 server generated by the Pathmaker product is run, it receives a message from a requester, invokes the appropriate subprogram (that is, service) to process the request, and returns a reply to the requester. To make this series of actions work, each custom COBOL85 server contains the following components:

- File description entries (FDs)
- Overall program control logic
- Code to handle messages in the \$RECEIVE file
- Error logic
- File opens and closes (if applicable)
- Host variables (for NonStop SQL tables)

C Custom Service. A Pathmaker service written in C consists of:

- A custom source file
- A generated source file
- An object file

A custom source file is created by the Pathmaker product when the application developer adds a C service. A custom source file initially contains empty definitions. Application developers add data declarations and functions to this file to perform the service's business task.

The Pathmaker product supplies a common service utility library that contains functions, constants, and definitions that can be referenced by code in a custom source file. The common service library includes constants for common Enscribe and NonStop SQL file system error codes, and functions for formatting error messages, manipulating certain character strings, and so on.

A generated source file is created by the Pathmaker product when a C custom service is generated. A generated source file contains C code that the Pathmaker product creates for such items as the service's request and reply message data structure definitions, data structure definitions for each logical file accessed by the service, and host variables for NonStop SQL tables. The generated source file for a service references the service's custom source file.

The Pathmaker product uses the generated source file as the primary input file when compiling a C service. C services can be generated and compiled separately outside of any server or they can be generated and compiled as part of a server generation and compile operation. In either case, a separate nonrunnable object file is created for each C service compiled.

Application developers package custom C services into custom C servers and then initiate the generation and compilation of those servers. When a C server is compiled, the object files for the included C services are bound into the server.

When a C server generated by the Pathmaker product is run, it receives a message from a requester, invokes the appropriate function (that is, service) to process the request, and returns a reply to the requester. To make this series of actions work, the Pathmaker product provides the following in each custom C server it generates:

- Overall program control logic
- Code to handle messages in the \$RECEIVE file
- Error logic
- Logical file opens, including the passing of file descriptors to the services using control parameters
- Logical file closes

Support for MAKE Utilities. The Pathmaker product provides limited support for compiling C services and servers using an external MAKE utility. (A MAKE utility is a tool commonly found in C program development environments that is used to compile only those objects that are not current with respect to other objects that they depend upon.) Although the Pathmaker product does not supply a MAKE utility, it does support the use of such a utility by updating the timestamp of a generated source file when the definition of the service or server in the project catalog has been changed, and by allowing application developers to generate a service or server without immediate compilation.

Other Capabilities In addition to creating Pathway requesters and servers, the Pathmaker product has other capabilities. This subsection briefly introduces these capabilities and, where appropriate, provides references to manuals containing additional information.

Accessing Pathway Applications From a Pathmaker Application It is possible for a Pathmaker application to access a Pathway application built outside the Pathmaker product.

To allow this access, an application developer must **register** every outside requester that is directly called by the Pathmaker application and must place the pseudocode for all of the outside requesters in the SCREEN COBOL library associated with the Pathmaker application.

Servers that are not created with the Pathmaker product can also be registered. Although registering servers is not necessary to access them from a Pathmaker application, doing so is beneficial in several ways. Because the Pathmaker catalog contains information about registered servers and the physical files that they access, the Pathmaker product can generate the statements necessary for running them in the Pathway command files. In addition, because the Pathmaker catalog contains the location of the object code for the registered servers, the Pathmaker product is able to move these files whenever the other components of the associated Pathmaker application are moved.

Pathway Open Environment Toolkit (POET) The Pathway Open Environment Toolkit (POET) is a Tandem product that assists in the creation and running of client/transaction server applications for Tandem systems. Application developers use POET to generate workstation clients that interact with Pathway servers. The Pathmaker product, version C31 and later, has been enhanced to provide support for the creation and running of client/transaction server applications. POET is not required to use the Pathmaker product. If you want more information about using POET and the Pathmaker product to develop client/transaction server applications, refer to the *Pathway Open Environment Toolkit (POET) Programming Manual*.

3 Using Pathmaker—An Overview

In the two previous sections, the benefits and features of the Pathmaker product and other related Tandem products, the types of Pathmaker applications that can be produced, and how Pathmaker applications look and behave were explained.

In this section:

- The major Pathmaker components are introduced.
- The term Pathmaker project is defined.
- Several general approaches for using the product are discussed.
- The phases of a formal Pathmaker application life cycle are suggested.

This information introduces many of the procedures that are detailed in the *Pathmaker Programming Guide*.

Major Pathmaker Components

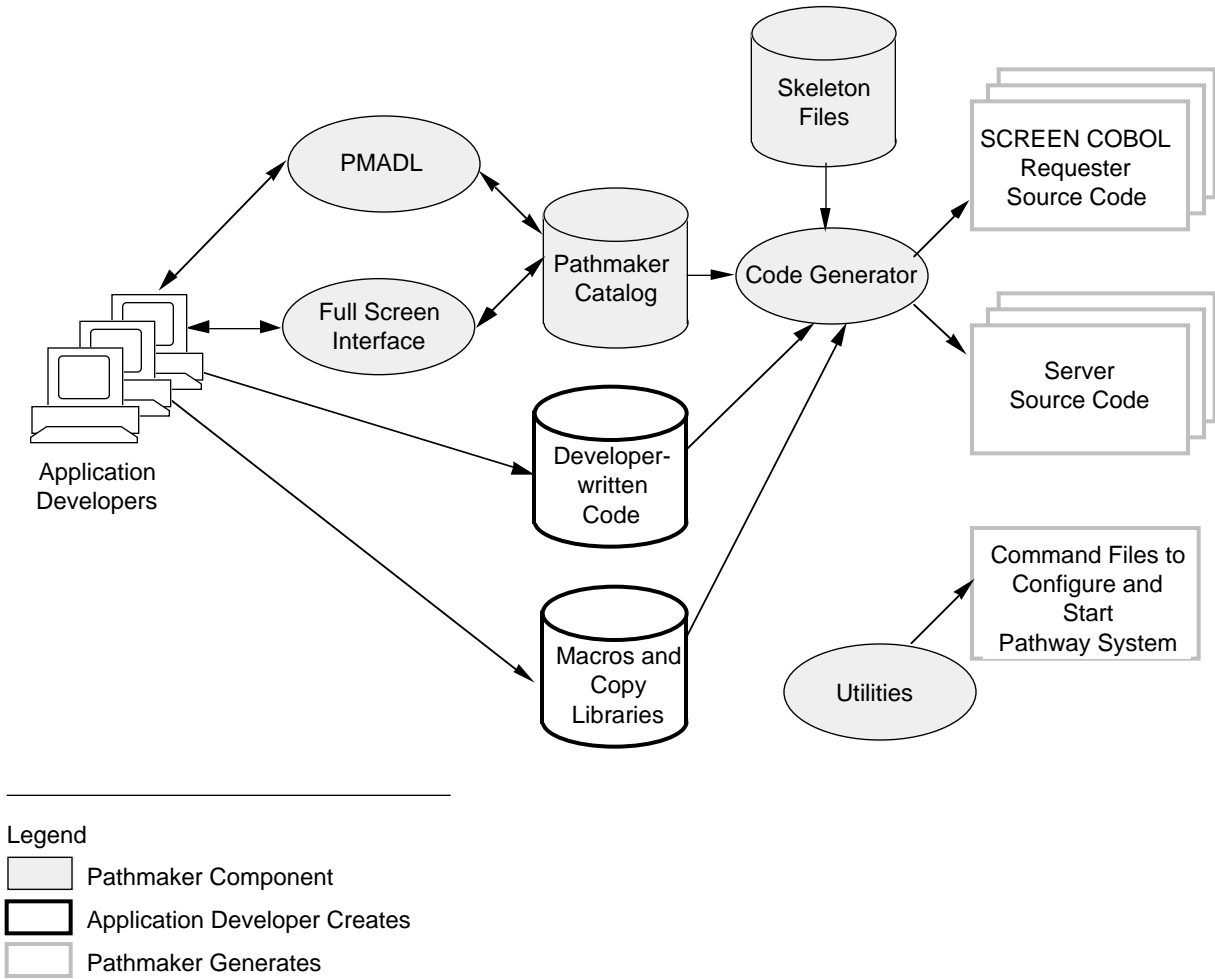
There are several major components of the Pathmaker product. This subsection briefly discusses and illustrates these components and provides general information about the purpose of each.

Pathmaker components include:

- A **catalog**
- A **full screen interface**
- The Pathmaker Application Definition Language (PMADL)
- Skeleton files and **code generators**
- Utilities**

Figure 3-1 illustrates the major components of the Pathmaker product.

Figure 3-1. Major Components of the Pathmaker Product



031

Pathmaker Catalog A Pathmaker catalog is a special group of files used to store information about a Pathmaker application.

A Pathmaker catalog is created when a Pathmaker **project** is created. In general, a single Pathmaker project results in a single Pathmaker application. (Pathmaker projects are explained in more detail later in this section.)

- Full Screen Interface** The Pathmaker full screen interface is, itself, a running Pathway system composed of requesters, server processes, server classes, and other Pathway entities. The full screen interface is used to enter information into a Pathmaker catalog. The interface presents a series of screens where information about the requesters, services, and servers that make up an application is entered by application developers. The Pathmaker full screen interface can run in block mode on the 6520/6530 family of Tandem terminals, including PCs with 6530 emulators. The full screen interface includes:
- A screen painter
 - A simulation feature
 - A help screen
 - A utility menu that provides interactive access to other Tandem products such as TEDIT, FUP, and so on
- Pathmaker Application Definition Language (PMADL)** The Pathmaker product provides the Pathmaker Application Definition Language (PMADL), which is a conversational or batch interface to a Pathmaker catalog. PMADL can be used in conjunction with, or as an alternative to, using the full screen interface. An existing Pathmaker catalog can be used as input to PMADL, which creates a text file that describes the contents of the catalog. This text file is then edited and used as input back into PMADL, which creates entries in a new catalog.
- PMADL is especially useful for making mass changes to a Pathmaker catalog. It can also be used to quickly create a Pathmaker application modeled after an existing Pathmaker application or to create a document that details the contents of a Pathmaker catalog.
- Skeleton Files and Code Generators** The Pathmaker product includes skeleton files and code generators that are used, in conjunction with the information entered into a Pathmaker catalog, to create SCREEN COBOL requester source code (including the code describing the end user screen) and the COBOL85 or C server source code for an application. In addition to the Pathmaker skeleton files, the following files are used as input to the code generators:
- Files created by the application developer containing additional SCREEN COBOL requester code.
 - Developer-written COBOL85 or C code that directs custom services to perform database access, data manipulation, calculations, and input validation.
 - Macros** and copy libraries.
- A macro exists as source code in a TEDIT file and is written in the Pathmaker macro language, which is unique to the Pathmaker product.

Utilities The Pathmaker product also includes several utilities, as follows:

- PMINSTAL, which assists in the installation of the Pathmaker product on a development system.
- HELPUTIL, which is used to modify the help text for an application created with the Pathmaker product.
- PMPROJECT, which is used to create and manage Pathmaker projects. (The term Pathmaker project is defined in the next subsection.) Some of the functions of PMPROJECT include adding a Pathmaker project and its associated catalog, altering project attributes, and moving a project to another system.

Detailed information about the Pathmaker utilities can be located in the *Pathmaker Reference Manual*.

A Pathmaker Project Project is a term that has several different meanings. A project can be thought of as all the files needed to complete an application development effort using the Pathmaker product. These files can include those that the Pathmaker product creates for the development of an application, the files that application developers create for the application (such as copy libraries and other source code files), and the compiled object code for the application.

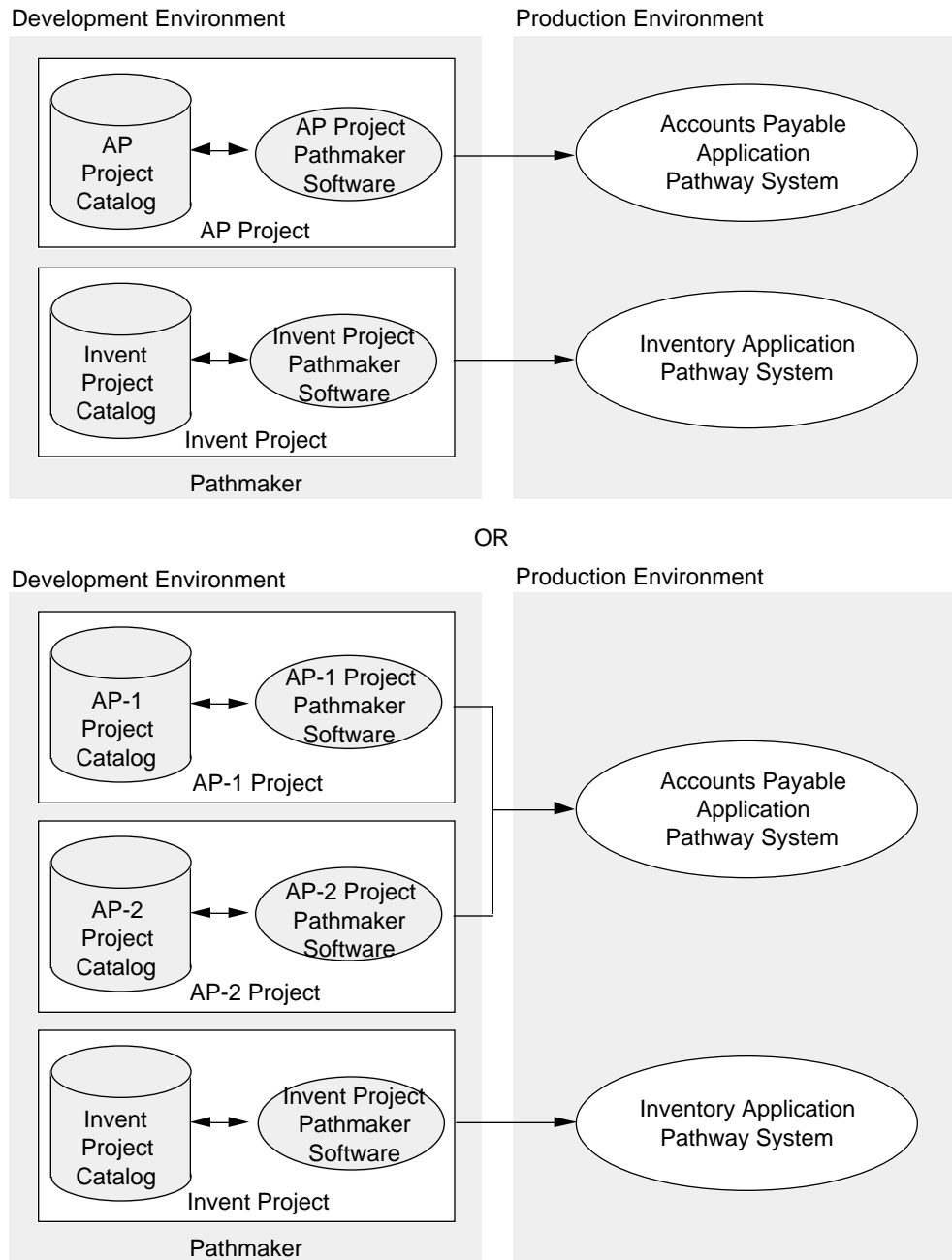
A project can also be thought of as the Pathmaker software interacting with the project files just described. Viewed this way, a Pathmaker project eventually results in an application for end users. The resulting application is often referred to as the **target application**.

In general, a single Pathmaker project results in a single Pathway application for the end user. For example, in a business environment, end users might need both a new accounts payable application and a new inventory application. To meet these needs effectively, it is likely that two separate Pathmaker projects would be added to the development system.

Alternatively, a very large application can be split into two or more Pathmaker projects during development and then merged together later. Developing a Pathmaker application in this manner requires special planning and procedures.

Figure 3-2 illustrates possible ways to set up Pathmaker projects that will be used to create two separate end-user applications.

Figure 3-2. Pathmaker Projects



Project Catalog During the preparation for a Pathmaker application development effort, Pathmaker software is installed on the development system and one or more Pathmaker projects are added.

When a Pathmaker project is added, a special group of files called a Pathmaker **project catalog** is created for that particular project. The files in a project catalog fall into several major categories, including administrative files, DDL dictionary files, help database files, and others. Each Pathmaker project catalog *must* be protected with TMF.

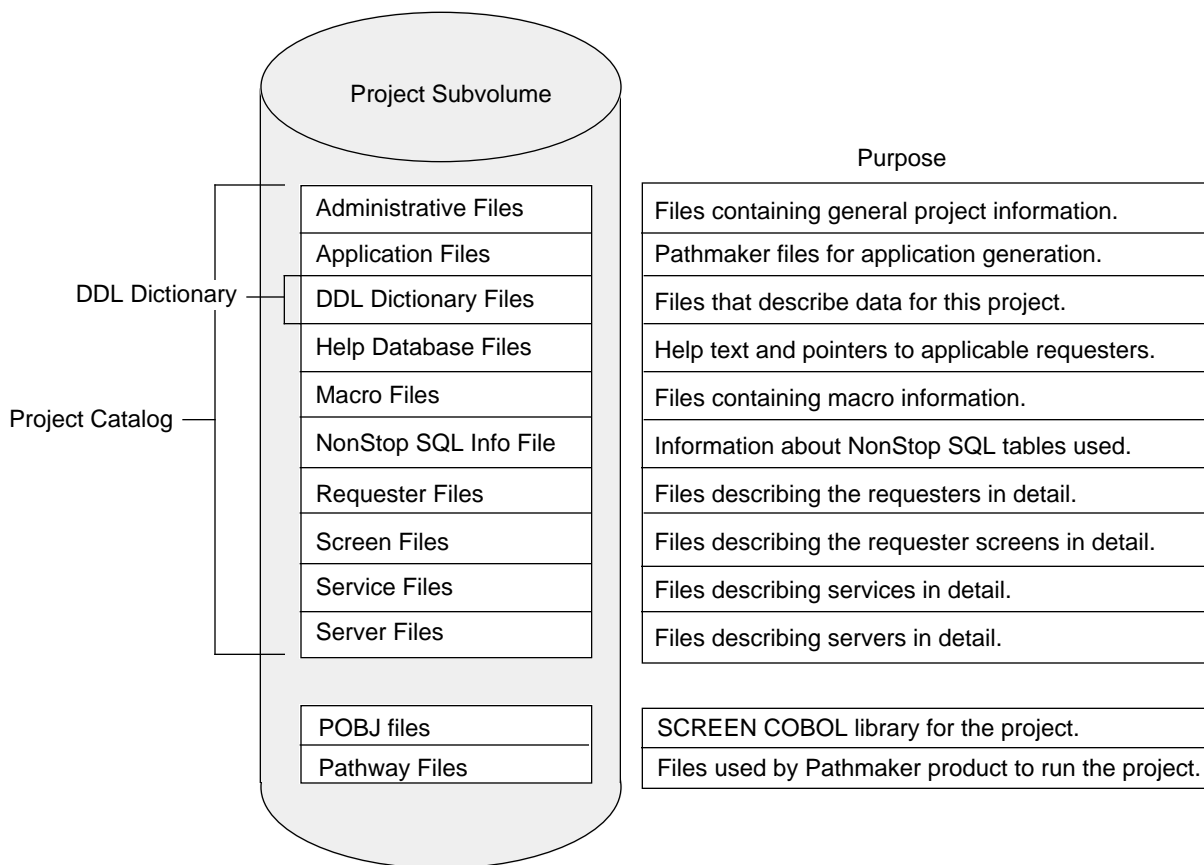
The project catalog is used to store information about the Pathmaker **objects** that make up this project. A Pathmaker object can be one of the following types:

- Registered NonStop SQL table
- DDL definition or record
- Requester
- Service
- Server
- Macro

Project Subvolume The project catalog resides on the **project subvolume**, which is specified when the project is first added to the development system. In addition to the project catalog files, each project subvolume also contains several other files associated with a particular project.

The contents of a project subvolume are illustrated in Figure 3-3. For more detailed information on the files in a Pathmaker project catalog, see the *Pathmaker Reference Manual*.

Figure 3-3. A Project Subvolume



Approaches for Using Pathmaker

The Pathmaker product was designed to provide application developers and designers with a great deal of flexibility. This flexibility allows the product to be used in different ways for different situations.

Not only can the Pathmaker product be used to create production applications in a formal and controlled manner, but it can also be used to design screen layouts before a database exists or to quickly create a prototype of a production application. This subsection presents several approaches for using the Pathmaker product and discusses the best use of each approach.

Creating Production Applications

In general, if the Pathmaker applications being created will be used to perform functions that are critical to the operation of a business, a formal approach to application development should be used. This is especially important if the proposed application is large or if many application developers will be working together to develop the application.

A formal approach generally means that a thorough and complete analysis and design has been completed, and the database for the application has been designed before work on creating the application begins. This approach also implies that the development of the application is handled in an orderly and controlled manner that allows the progress of application development to be tracked and documented.

Later in this section, a formal **life cycle** that can be used to develop production applications is suggested. The phases of this life cycle encompass multiple tasks, many of which can be accomplished with the Pathmaker product. The *Pathmaker Programming Guide* provides the details of the tasks that should be completed using this formal approach.

Designing Screens

Although the Pathmaker product's main focus is simplifying the creation of Pathway requester-server applications, it can be also used as a screen design tool. After a few tasks to prepare the environment (such as installing the Pathmaker software and adding a Pathmaker project) have been performed, application designers or developers can quickly create Pathmaker requesters and then use the screen painter to draw a proposed application's screens. These procedures can even be done before the application database exists.

If the data for the application has already been defined in the Pathmaker catalog, the Pathmaker product can use this information to create a default screen that application designers or developers can modify with the screen painter. If no data has been described to the Pathmaker product, application designers or developers can simply use the screen painter to paint the screens, using decorations (for example, underscores) to represent the data fields.

There are several advantages to designing screen layouts after data for the application has been defined in the Pathmaker catalog. These advantages are as follows:

- Allowing the Pathmaker product to create the data fields and their associated headings on requester screens from information in the catalog is much faster than painting alone and allows the requester to be used for a production application with much less modification.
- When simulating a requester screen that contains data fields created from information in the catalog, it is possible to key data into the data fields (although the data is not retained.)

Designing screen layouts after data for the application has been defined in the Pathmaker catalog is a good choice if an existing application is being converted or a new application for an existing database is being written.

If, however, the screen painter is used to paint the screens, using underscores or other characters to represent the data fields, the requesters for those screens can later be converted into functioning requesters after data for the application has been defined in the Pathmaker catalog.

Before deciding which of these methods to use, it is helpful to experiment with converting decorations that represent data fields to functioning data fields using the screen painter. Conversion can be a time-consuming process for a complex application with many screens. If many screens with data fields on each screen have been painted, it is probably more efficient to have the Pathmaker product re-create the default screens after data for the application has been defined in the Pathmaker catalog instead of having application developers convert each screen using the screen painter.

Using Simulation At any point after a requester has been defined, whether or not the data fields have been described to the Pathmaker product, the requester can be simulated for end users. After several requesters have been created, the navigation among their screens can also be simulated. To use this feature, simply select simulation on the Pathmaker Main Menu screen. The name of an existing requester must be entered; it is then possible to move through the application's user interface by using the function keys defined for screen navigation.

During simulation, the Pathmaker product first displays the requester whose name was entered on the Pathmaker Main Menu and then allows navigation to other requesters in the application. During this process, the appearance of the application can be demonstrated, but the requester cannot process data.

Simulating the appearance of the application screens and navigation among those screens is an important feature of the Pathmaker product that can be used during the design of an application, as well as during the development of an application. Potential end users can view the user interface and provide valuable feedback early in the application life cycle. Simulation is a good method of verifying the basic presentation of screens without the cost of creating services and configuring a Pathway system.

Simulating the interface before data for the application has been defined in the Pathmaker catalog can even help to identify fields that must be present in the database; however, simulation will not always help to identify relationships between the data elements.

Note The Pathmaker product is not intended as a database design tool; if prior database analysis or design has not been completed, it is possible to create a user interface that cannot be implemented effectively.

Creating Prototypes The screen painter and simulation features of the Pathmaker product are very effective for previewing the screens of a proposed application, but end users might also want to preview some of the functionality of a proposed application.

One of the best ways application developers can demonstrate application functionality is to create a prototype of a proposed production application. A prototype is a simplified working model on which a custom production application will be based.

Using Pathmaker DB requesters and the standard services and servers provided with the product, application developers can quickly create a working prototype that demonstrates basic application functionality such as adding, updating, and deleting information in the database.

A Suggested Life Cycle for a Pathmaker Application

Because the Pathmaker product is designed to be flexible, it does not always dictate a fixed order for completing the tasks involved in a Pathmaker application development effort. While this flexibility is often an advantage, it could cause confusion for first time Pathmaker users or for project teams that consist of many members. Pathmaker users need detailed information about what Pathmaker tasks need to be completed and the best time to complete them.

This subsection addresses many of these concerns by suggesting a life cycle that can be used as a high-level guide to a Pathmaker application development effort. Life cycle is defined as an orderly series of events, usually divided into phases, in the life of an application. Each phase should clearly define a set of objectives, the desired output, and the required input.

Details about the tasks in each phase of the life cycle where the Pathmaker product is used can be found in the *Pathmaker Programming Guide*. The major phases in the suggested life cycle are as follows:

- Definition
- Design
- Planning
- Setup
- Development
- Documentation
- Testing
- Move into production
- Maintenance

This suggested life cycle is intended for a formal Pathmaker application development effort that will result in a production application. Although this is certainly not the only possible life cycle, it is one that covers all of the important tasks that need to be completed to successfully create a Pathmaker application. You are encouraged to adapt this suggested life cycle to conform to the requirements of your particular environment.

Definition The first phase of an application life cycle is definition. In this phase, a formal written statement, clearly defining the application the end user wants and what the application designers agree to provide, must be produced. This statement is usually referred to as the problem specification and defines the job to be done. The Pathmaker product is not used for this life cycle phase.

Design The next phase is application design in which a design specification is created. The design specification is a written document that states the solution to the end user's business problem, which was identified in the definition phase. This document will become the focal point for application developers using the Pathmaker product.

When completing this phase, attention to detail and consideration for the architecture of a Pathmaker application is essential to the ultimate success of the application.

Work should not begin on the development of a production Pathmaker application until the design of the application is complete and has been thoroughly documented.

The design document should include the following information:

- Description of the database used by this application, as follows:
 - Names of NonStop SQL tables and Enscribe files (DEFINE names and the Pathmaker name for each)
 - Descriptions of columns and fields
 - Statement of integrity rules
 - Entity relationship diagram
- Chart of the application hierarchy
- Standards and naming conventions

Note DEFINE name is an alias that can be used to identify a file or table in a service. Later, when a server containing that service is run, the actual name of the file or table will be substituted by the system.

- Description of each transaction (service) in this application and the elements of each, including:
 - Name of service and description
 - Input needed (Request message)
 - Transaction processing instructions (input validation, integrity checks, calculations, database access, and so on)
 - Output needed (Reply messages)

- Description of each requester in this application, including:
 - Name of the requester
 - Screen layout
 - End user's terminal type
 - Function keys to be used and their associated actions
 - Data the requester expects to receive as parameters
 - Data the requester must send to each service invoked (request message)
 - Data the requester must pass as parameters to each requester invoked
 - Help text for end user
- Service packaging information
 - Name of the server
 - List of services that are to be packaged in each server

Note As an alternative to producing a written design document, the application designers can use Pathmaker development screens to record much of the design information. The resulting catalog can then serve as a starting point for the application developers.

Planning In this phase of the application life cycle, decisions must be made to allocate the resources needed to implement the design. These decisions should include:

- Identification of team members
- Assignment of specific tasks
- What equipment is to be used (system, volumes, processors, printers, and so on)
- Schedules
- Change procedures
- Training plan

The Pathmaker product is not generally used for this life cycle phase.

Setup This phase focuses on preparing the Pathmaker environment to implement the design.

Tasks during this phase include:

- Installing Pathmaker software
- Installing related products (TMF, Enform, and so on, if necessary)
- Assigning logons and subvolumes
- Preparing DDL schema
- Setting up the test database. (This can occur later for Enscribe files only.)
- Adding Pathmaker projects
- Customizing Pathmaker projects
- Compiling DDL schema to update DDL dictionary files
- Registering NonStop SQL tables
- Creating shared code (optional)

Development The development phase is the portion of the life cycle when the application design is implemented into requesters, services, and servers. The Pathmaker product's primary job is to simplify application development; however, to properly implement a production application, application design and planning must be done first. With the design specification and project plan as guides, application developers use the Pathmaker product to:

- Describe services and servers (for custom applications)
- Generate NonStop SQL data manipulation statements to include in service code (for custom applications that access NonStop SQL data)
- Write service code (for custom applications)
- Describe requesters and their screens
- Modify screens (where necessary)
- Generate and compile requester source code
- Package services into servers (for custom applications)
- Generate and compile server code (for custom applications)

Documentation During this phase (which should be done in parallel with development), information about the implemented application is recorded. This information will be used for training, maintenance, and auditing purposes and to track the progress of application development.

Documentation can be produced by printing hard copies of end user application screens, the help text for end users, and all Pathmaker screens used to develop the application. PMADL can be used to generate a detailed listing of the objects in a Pathmaker project. In addition, a number of Pathmaker reports can be produced that provide valuable information about an application from the information found in the Pathmaker catalog.

Testing After the application (or a portion of it) has been developed, it is time to verify that it works properly and according to the design. The type and depth of testing for an application should be defined during the planning phase. Pathmaker provides a utility (PMPROJECT) that creates Pathway command files that can be used to configure and start a Pathway system for testing.

Move Into Production After thorough testing and acceptance by the end users, auditors, and other concerned parties, the application is ready for production. The Pathmaker utility PMPROJECT creates a file that can be used to FUP DUP the needed application components into the production environment.

Maintenance After an application has been placed into production, changes to enhance or modify the application are inevitable. The Pathmaker product can be used to perform many application maintenance tasks.

Table 3-1 summarizes the phases in the life cycle of a Pathmaker application.

Table 3-1. Life Cycle of a Pathmaker Application

Phase	Objective	Input	Output	Pathmaker Feature
Definition	To clearly define the business problem.	Information from end users	Problem specification	
Design	To describe the overall solution.	Problem specification Designer ideas End user feedback	Design specification	Simulation (Note: Setup must be done first)
Planning	To plan resources for implementing design.	Design specification Resource information	Project plan	
Setup	To prepare Pathmaker environment for implementing design.	Project plan	Pathmaker product, Pathmaker project(s) and catalog(s), Subvolumes, Logons, DDL schema, Updated DDL dictionary, NonStop SQL database	Pathmaker product installation utility (PMINSTAL)
Development	To implement design with Pathmaker product.	Design specification Project plan	Updated Pathmaker catalog, POBJ files, Server source code, Server object code, Pathway command files	Full screen interface or PMADL to add information to catalog Skeleton files and code generators to generate source code Utilities
Documentation	To record information about application for training and maintenance.	Updated Pathmaker catalog	Hard-copy documentation	Pathmaker print facility (screens and help text) Pathmaker reports PMADL
Testing	To verify that the application works properly and according to design.	Updated Pathmaker catalog (with application installation information) Test database Server object code Pathway command files POBJ files	Tested application	Pathmaker PMPROJECT utility (install option)
Move Into Production	To move from the test environment into the production environment.	DUPCODE command file NonStop SQL Compiler*	Copies of: Server object code, DB standard servers, Help server and files, POBJ files	Provides FUP DUP files Provides files for SQL compiling all SQL Servers
Maintenance	To enhance or change the application.	Pathmaker catalog	Updated catalog, Copy of changed object into production	Same as "Development" and "Move Into Production"

* Note: Servers containing SQL commands must be recompiled when they are moved into production.

A Final Note With the completion of the *Introduction to Pathmaker* manual, you have gained a foundation of basic knowledge about the Pathmaker product that will enable you to effectively use the information presented in the *Pathmaker Programming Guide* and the *Pathmaker Reference Manual*.

Glossary

application. A set of programs designed to perform a specified task, usually involving operations on a database.

application developer. A programmer using the Pathmaker product or some other means to create an application.

application screen. The data entry screen or menu screen for an application. Application screens are displayed to end users when they run a Pathmaker application.

An application screen can be designed entirely by setting screen layout parameters in the Requester Definition screen and in its related screens, or the application screen can be enhanced by using the screen painter.

catalog. See project catalog.

catalog subvolume. See project subvolume.

COBOL85. The Tandem compiler and run-time support for the American National Standard Programming Language, COBOL, X3.23-1985.

code generator. A component of the Pathmaker product used to generate Pathmaker requesters and servers.

code macro. See macro.

context-free. A term applied to services and servers that specifies that they do not save any information about a request to be used by subsequent requests. All Pathmaker services and servers must be context-free.

custom application. A type of Pathmaker application created to handle the high volume online transaction processing needs of a business. In addition to database access tasks, a custom application can be created to perform database integrity checks, complex editing, data manipulation, and calculations. A custom application is usually composed of one or more TRNS (transaction) requesters, one or more custom services (packaged into custom servers), and one or more menu requesters.

custom server. A Pathmaker server that contains code for executing custom services.

custom service. A unit of work performed by a server, corresponding to a single SEND operation in a Pathway environment. Custom services are written in COBOL85 or C to perform a specific transaction. A custom service might be one that inserts an airline flight reservation record or transfers money from one bank account record to another.

custom source file. An EDIT file that contains code written by an application developer for one or more Pathmaker custom services. In previous versions of the Pathmaker product, these files were called Transaction Copy Libraries.

Data Definition Language. See DDL.

data field. A portion of the application screen dedicated to accepting data typed in by the end user and displaying data from the application. A data field consists of a set of contiguous positions on the screen and can be underlined for emphasis. Data fields have an associated DDL or NonStop SQL data definition.

data field area. On a Pathmaker default screen, the area between the two rows of equal signs that contains all of the data fields and the headings that identify them.

data field attribute. A characteristic of a data field. Data field attributes can include: length, data type, default value, allowable values, video attributes, protection from data entry, display format, heading, upshifting of entered data, and help text.

data type. A category of data. Specific data types can vary from programming language to programming language. For example, in COBOL, data can be numeric, alphanumeric, alphabetic, alphanumeric edited, and numeric edited.

database. A structured set of information, stored on some relatively permanent medium, describing some entities such as the employees of a business. The physical database usually is kept in one or more disk files that application programs can add to, delete from, alter, or read.

database requester. See DB requester.

DB (database) requester. A type of Pathmaker requester that uses a set of predefined services to read, insert, delete, or update data. This kind of requester cannot call custom services but can call other requesters. A DB requester uses standard services contained in a standard server (sometimes called a general server) whose performance might not be as efficient as that of custom services; it is not the responsibility of the standard server to guarantee referential integrity.

DB requester application. A type of Pathmaker application created to add, maintain, or retrieve information from one or more related tables (or files) in a database. DB requester applications cannot perform calculations or data manipulations or, more importantly, perform complex integrity checks on the database. A DB requester application is usually composed of one or more DB (database) requesters, one or more standard services (packaged into standard servers), and one or more menu requesters.

DDL (Data Definition Language).

A Tandem product that provides both a language used to define data and a program that compiles the data definitions and generates various types of output. The output can include data dictionary entries, FUP commands, and source code that describes the data, in a variety of programming languages.

DDL dictionary. A database that contains the formal description of an application's data in accordance with the data definitions found in a schema. A DDL dictionary is part of every Pathmaker catalog. Several Tandem products, including the Pathmaker product, recognize the format of these dictionary files and use the information in them to perform various tasks, such as reporting, program generation, and the creation of SQL commands for table creation.

decoration. Any screen element that displays unchanging text on a screen. Decorations that the Pathmaker product automatically creates include the requester title and data field headings. Using the screen painter, lines, borders, boxes, column headings, or other displayed text can be manually added.

default screen. The application screen that the Pathmaker product creates from the screen layout information application developers provide.

developer. See application developer.

Encompass. A set of data management products offered by Tandem that includes the DDL, Enscribe, Transfer, Pathway, TMF, and Enform products.

end user. A person running a target application generated by the Pathmaker product.

Enform. A Tandem product that generates reports from information contained in a DDL dictionary.

Enscribe. A Tandem software product that provides high-level access to, and manipulation of, records in an Encompass database. As an integral part of the Tandem NonStop operating system, Enscribe helps ensure data integrity if a processor module, I/O channel, or disk drive fails.

Enscribe DDL. See DDL.

File Utility Program (FUP). A Tandem product used to interactively create, purge, and otherwise manipulate disk files.

full screen interface. A Pathmaker component used to enter information into a Pathmaker catalog. The full screen interface presents a series of screens where information about the requesters, services, and servers that make up an application is entered by application developers. The full screen interface is, itself, a running Pathway system.

function key list. A list that appears on an application screen showing available function keys (the list might show only a portion of the available function keys). On a Pathmaker default screen, the area beneath the second row of equal signs contains the function key list.

FUP. See File Utility Program.

general server. See standard server.

heading. The brief, explanatory text associated with a particular data field on a screen. For example, the heading for a data field used to enter and display an employee's last name might have Last Name as its heading.

help text. Online explanations of the use of a screen, the fields on the screen, and the function keys on the screen. Help text is stored in a set of key-sequenced files called the help database. Within the Pathmaker full screen interface, help text is accessed for each screen by pressing function key F15. The Pathmaker product allows application developers to create similar help text for the target application.

HELPUTIL. A Pathmaker utility used to modify the help text for an application created with the Pathmaker product.

hierarchical structure. The screen navigation structure that the Pathmaker product uses when creating an application. In an application with a hierarchical structure, the application's requesters are arranged in a fixed hierarchy, sometimes referred to as a tree. Typically, this hierarchy has a main menu requester at the top level with sub-menu requesters below leading to detail requesters (used for data entry or data retrieval) at the lowest levels.

host language. Any programming language whose statements can be combined with SQL statements in the same source program.

host program. A source program that contains statements written in both a host language and SQL.

host variable. A variable declared in a host program and referred to by an SQL statement embedded in the same source program. The Pathmaker product declares host variables if host variable names have been specified in the Logical File Name column of the Logical File Entries screen.

interprocess communication message. See IPC.

IPC (interprocess communication message). A unit of communication between requesters and servers. The Pathmaker product generates interprocess communication messages that have a standard header. IPC messages generated for Pathway Open Environment Toolkit (POET) applications have a different header.

JET6530 terminal. A terminal that supports Kanji and acts like a 6530 terminal.

Kanji. A character set that is not phonetic used in written Japanese.

life cycle. An orderly series of events, usually divided into phases, in the life of an application.

logical screen. A logical screen defines the locations, lengths, and other attributes of fields displayed on the end user's screen. The logical screen is equivalent to the Screen Section of a SCREEN COBOL program unit. See also page.

macro. A unit of parameterized text that can be included in a generated source. A macro can be pure text, equivalent to a copy library, or can use a macro language to control the emitted text.

macro language. A language unique to the Pathmaker product used to create the requester skeleton, the server skeletons, and the C service skeleton supplied with the product. The language can be used by application developers to change the skeletons supplied with the product or to create other application files such as macros. The language consists of two types of statements: source lines and command lines.

menu (MENU) requester. A type of Pathmaker requester that allows data entry and navigation to other requesters but does not support transmittal of data to a server.

message area. A reserved portion of the screen where messages and errors are displayed to the end user. The message area on a Pathmaker default screen is below line 24.

monolithic application. A single program that performs all terminal handling, communications, and database management operations.

multifile DB requester. A type of Pathmaker DB requester that accesses multiple related tables or files.

NonStop SQL. The Tandem implementation of a distributed SQL (structured query language) relational database management system.

object. See Pathmaker object.

OLTP (online transaction processing) application. A set of programs designed to perform a specific set of tasks by processing transactions that change the database from one consistent state to another. These changes to the database occur immediately (online) instead of being deferred.

online transaction processing. See OLTP application.

page. All data fields and decorations displayed at one time on the application screen. A page might correspond to a subset of the Screen Section. A single logical screen might have one or many pages. See also logical screen.

paging area. An area within an application screen on which pages are displayed. The paging area on a default screen is initially the midportion of the screen, from the row of equal signs (=) below the screen title to the row of equal signs above the function key list. The screen painter can be used to redefine the paging area.

PATHCOM. A component of a running Pathway system. PATHCOM is a process that provides a command language interface, allowing a system operator to control and direct PATHMON.

Pathmaker. An application generator that increases the productivity of programmers developing applications for a Tandem NonStop computer system.

Pathmaker Application Definition Language. See PMADL.

Pathmaker object. An object defined in a Pathmaker project catalog. A Pathmaker object can be one of the following types: registered NonStop SQL table, DDL definition or record, requester, server, service, or macro.

Pathmaker version. The Pathmaker software (program object files, data files, help text files, and support programs) plus compatible versions of other Tandem products used with the Pathmaker product, such as DDL, Enform, and Pathway. Each software release contains a new version.

All versions of the Pathmaker product installed on a system are registered in a single file, \$SYSTEM.PATHMAKR.INSTALLS, on that system.

PATHMON. The supervising process of a Pathway environment. PATHMON manages all other processes in a running Pathway system.

Pathway. A Tandem product that helps simplify the development and control of online transaction processing applications. Pathway includes SCREEN COBOL for requester development, terminal control processes (TCPs) that use the SCREEN COBOL code to control terminals, PATHMON for central control of all processes, and the PATHMON command interpreter.

Pathway command file. A file containing commands that define and add the Pathway objects required to execute an application. This file can contain all the commands needed to start a Pathway system.

Pathway Open Environment Toolkit. See POET.

PMADL (Pathmaker Application Definition Language). A component of the Pathmaker product that provides a conversational or batch interface to a Pathmaker catalog.

PMINSTAL. A Pathmaker utility that assists in the installation of the Pathmaker product on a system.

PMPROJECT. A Pathmaker utility used to create and manage Pathmaker projects.

POBJCOD. A required file in the SCREEN COBOL library that contains requester object programs.

POBJDIR. A required file in the SCREEN COBOL library that contains the directory to the code files in POBJCOD.

POBJSYM. An optional file in the SCREEN COBOL library that contains the symbols tables for the requester object programs.

POET (Pathway Open Environment Toolkit). A Tandem product that assists in the creation and running of client/server applications for a Tandem system.

process. In a Tandem computer system, a process is essentially any running program.

project. 1. All the files needed to complete an application development effort using the Pathmaker product. 2. The Pathmaker product interacting with the project files resulting in a Pathmaker application.

project catalog. A set of files in which the Pathmaker product maintains information about a specific ongoing project. These files reside on the project's subvolume and are associated with only that project.

project subvolume. The subvolume in which the Pathmaker product creates a project catalog and other project files. The project subvolume is specified when a Pathmaker project is added using the PMPROJECT utility.

prototype. A simplified working model of an application on which a custom application will be based. A prototype is not intended for use in production.

registered (REG) requester. A requester written outside of the Pathmaker environment but known to a Pathmaker project. To create a Pathmaker application that accesses such requesters, a registered requester object must be created in the Pathmaker catalog for every outside requester directly called by the Pathmaker application. In addition, the psuedocode for all of the outside requesters, even those not directly called, must be placed in the SCREEN COBOL library associated with the Pathmaker application.

registered (REG) service/server. A service or server written outside of the Pathmaker environment but known to a Pathmaker project. It is possible to create a Pathmaker application that accesses outside servers without registering them; however, registering them allows the Pathmaker product to move their associated files during import and export operations. Identifying the files used by a server through service registration allows the Pathmaker product to generate statements in the Pathway command file for running the servers.

release number. A code that designates the release date of a Tandem product. The release number begins with a letter and is followed by a two-digit number. For example, C20 is the release number for some products released in 1989. Look on the title page of this manual to find out which release of the Pathmaker product this edition documents.

reply message. A type of interprocess communication (IPC) message formatted by a service and sent back to a requester after the service has completed its work. A reply message is initiated in the COBOL85 server code by a WRITE statement to the \$RECEIVE file.

report. A listing of information from a project's catalog or from an application's database. The Pathmaker product provides a number of standard Enform reports showing such information as object types defined in the catalog, service and server structure, relationships between requesters, and many more details. To create reports that extract information from an application database, application developers must create requesters and servers for this purpose.

requester. 1. A process controlling the interaction between servers and end users. A requester can control terminal displays; accept, interpret, and validate input from the end user; request database access through servers; and return information to the end user. In Pathmaker and other Pathway environments, terminal control processes (TCPs) interpret SCREEN COBOL programs to serve as requesters. 2. A Pathmaker application module corresponding to a SCREEN COBOL program unit. A Pathmaker requester contains presentation logic for a single logical screen (which can contain several pages), manages terminal context associated with that screen, communicates with services, and contains navigation logic. There are four types of Pathmaker requesters: database (DB), transaction (TRNS), menu (MENU), and registered (REG).

requester copy library. An optional EDIT file that contains additional SCREEN COBOL code written by an application developer for inclusion in a Pathmaker requester.

requester-server application. An application divided into two parts: the requesters, which perform data collection, and the servers, which access and modify the database. Requesters and servers communicate with interprocess communication messages.

requester skeleton. The framework for all generated requesters. The requester skeleton consists of SCREEN COBOL source text lines plus special macro language command lines.

The requester skeleton consists of two files: REQSKL and REQDBSKL. REQDBSKL contains sections that only pertain to DB requesters. In discussions in which the term requester skeleton is used without specifying a file, REQSKL is meant.

request message. A type of interprocess communication (IPC) message sent from a requester and received by a particular service. When an end user presses a function key associated with a business task, the requester formats a message that identifies the service and includes the data needed by the service to do its work. A request message is initiated in the SCREEN COBOL requester code by a SEND statement. Pathway routes the request to a server process containing the desired service.

schema. An EDIT file that contains a formal representation of data composed of DDL statements that describe the data as fields, groups of fields, or records.

screen. 1. That part of the terminal on which screen images are displayed. 2. A screen image. 3. The logical screen associated with a requester, corresponding to the SCREEN section of a SCREEN COBOL program unit. The logical screen defines the locations, lengths, and other attributes of fields displayed on the end user's screen.

SCREEN COBOL. A language similar to COBOL designed specifically for writing requester programs, including descriptions of screen images. This language is supplied as part of Pathway. The Pathmaker product generates SCREEN COBOL programs from the requester information in a project catalog.

SCREEN COBOL library. A set of disk files that contains the requester programs after they have been compiled by the SCOBOLX compiler provided with Pathway. The library consists of these files: POBJDIR, POBJCOD, and, optionally, POBJSYM.

screen navigation. The ability to move between the requester screens in an application.

screen painter. A component of the Pathmaker full screen interface that allows application developers to modify application screens generated by the Pathmaker product. With the screen painter, data items can be moved to different screen locations and borders drawn to improve a screen's readability.

server. A program comprising one or more services. Servers communicate with requesters and access and manipulate the database. Each server is a context-free process that does not require memory of a previous transaction in order to process the current transaction. See also registered service/server, custom server, and standard server.

server class. A named family of identical server processes, each of which is created from the same object file on disk. Server classes are created by Pathway at run time.

server process. A running database management program that handles IPC messages for one or more services. Pathway uses object code to create running server processes.

server skeleton. A file, consisting of source text lines plus special macro language command lines, that the Pathmaker product uses to generate server source code. There are two server skeletons provided with the Pathmaker product—one for generating COBOL85 servers and one for generating C servers.

service. A unit of work to be performed by a server, corresponding to a single SEND operation in a Pathway environment. See also custom service, registered service/server, and standard service.

service skeleton. A file, consisting of source text lines plus special macro language command lines, that the Pathmaker product uses to generate C service source code.

single-file DB requester. A type of Pathmaker DB requester that accesses only one table or file.

simulation. The displaying of sequences of screens that provides a preview of an application in its final form. Through simulation, end users can see what an application will look like before it is actually coded and installed.

skeleton files. Files included with the Pathmaker product that are used in conjunction with the code generators to generate requester source code, COBOL85 and C server source code, and C service source code.

skeleton language. See macro language.

standard server. A server supplied by Tandem that contains code for executing the Pathmaker standard services. Standard services can only be invoked from DB requesters. Tandem provides a standard server for Enscribe databases (DBSERVER) and a standard server for NonStop SQL databases (SQLGS).

standard service. A predefined service provided by the Pathmaker product to do basic operations on an application database. Standard services include reading the next record or inserting a new record. See also custom service and service.

target application. The application created as a result of a Pathmaker application development effort.

target terminal. The type of terminal on which an application screen is displayed. Each type of terminal has certain characteristics, such as supported video attributes, that limit allowable screen images. The Pathmaker product supports 6520, 6530, JET6530, and 3270-compatible terminals.

TCP (Terminal Control Process). A Pathway requester process responsible for terminal management and transaction control. A TCP interprets SCREEN COBOL object code and communicates with server processes.

TEDIT. The Tandem text editor.

Terminal Control Process. See TCP.

test configuration file. One of a set of files created by the PMPROJECT utility when a Pathmaker target application is installed. The test configuration files are used to create the application test database (for Enscribe only) and to configure and start the application's test Pathway system.

title area. On a Pathmaker default screen, the area above the first row of equal signs. This area contains the requester title, which is a brief description that helps the end user identify the major purpose of a particular requester screen and a page designator (for example, Page 1/3), which shows the number of the current page and the total number of pages for this screen.

TMF (Transaction Monitoring Facility). A Tandem product that controls transaction concurrency and protects data against system failures.

transaction. A group of computer operations that reflect a particular commercial interaction by changing a database from one consistent state to another consistent state. These operations, including the entry of data about the event, the processing of that data, and the resulting change in the database, provide a computer model of the real events making up the transaction. A transaction might be one that inserts an airline flight reservation record or transfers money from one bank account record to another.

Transaction Copy Library. A term used in previous versions of the Pathmaker product to refer to an EDIT file that contained the Working-Storage Section, the Procedure Division, and, optionally, the Extended-Storage Section, for one or more Pathmaker COBOL85 services. These files are now called Custom Source Files.

Transaction Monitoring Facility. See TMF.

transaction (TRNS) requester. A type of Pathmaker requester that accepts data entry and calls services to process the data entered. A TRNS requester is more versatile than a database (DB) requester because it can call custom services.

user. See end user.

user interface. See full screen interface.

utilities. Programs included with the Pathmaker product that assist with miscellaneous tasks. These programs include: PMINSTAL, which assists in the installation of the Pathmaker product on a system; HELPUTIL, which is used to modify help text for a Pathmaker application; and PMPROJECT, which is used to create and manage Pathmaker projects.

video attribute. A property of a screen element that determines the visual appearance of the screen-image locations occupied by the element. An element can be bright or dim, steady or blinking, light-on-dark or reversed, underlined or not, visible or hidden.

3270 terminal. A family of terminals manufactured by IBM and other companies but sometimes used with NonStop systems.

6520 terminal. A terminal model formerly sold by Tandem. Its black-and-white screen and its keyboard occupy a single cabinet.

6530 terminal. A terminal family manufactured and sold by Tandem. Terminals in this family have green phosphor screens, and their keyboards are detached.

Index

A

Application

- definition of 1-4, Glossary-1
- design features of Pathmaker 1-33
- developer 1-15, 1-26, 1-27, Glossary-1
- life cycle 3-9
 - definition phase 3-12
 - design phase 3-13
 - development phase 3-15
 - documentation phase 3-16
 - maintenance phase 3-16
 - planning phase 3-14
 - production phase 3-16
 - setup phase 3-15
 - testing phase 3-16

monolithic 1-10

OLTP (online transaction processing)

- creating with Pathway 1-11, 1-15
- definition of 1-4
- functions of 1-6
- processing requirements of 1-8
- running with Pathway 1-18

Pathmaker

See Pathmaker, application

prototype 3-11

screen 1-2, 2-1, 2-9, 2-11, 3-9, Glossary-1

Application Definition Language (PMADL) 1-2, 1-29, 1-32, 3-3

Architecture of Pathmaker application 2-1

Attributes of a data field 2-6

Auditing requirements for Pathmaker 1-41, 3-7

B

BEGIN-TRANSACTION statement 1-41

Benefits

- of Pathmaker 1-1, 1-26
- of Pathway 1-9

C

- C custom service 2-20
- Catalog 1-26, 3-2, 3-7
 - reports 1-32
 - subvolume 3-8
- Client/transaction server applications 2-22
- COBOL85
 - custom service 2-20
 - definition of Glossary-1
- Code generators 3-3
- Code macros 1-32, 3-3
- Code sharing 1-32, 3-3
- Command files
 - See* Pathway, command files
- Common service utility library 2-21
- Communication message 1-11, 1-16
- Configuration file 1-15, 1-25, 1-27, 1-30
- Context-free service 1-22
- Copy libraries 3-3
- Custom application
 - components 2-19
 - operation 2-18
 - screen 2-19
- Custom server 2-19
- Custom service
 - C 2-20
 - COBOL85 2-20
 - SQL statement generation 1-26
- Custom source file 2-20

D

- Data
 - entering in a Pathmaker application 2-7
 - receiving from a Pathmaker application 2-8
- Data Definition Language
 - See* DDL
- Data field
 - area 2-1
 - attributes 2-6
- Data type 2-6, 2-7, 2-8, Glossary-2

-
- Database
 - comparison of Enscribe and NonStop SQL 1-38
 - creation 1-40
 - definition of 1-19
 - effect on Pathmaker 1-37
 - Enscribe
 - creating 1-35
 - definition of 1-34
 - keys 2-14
 - NonStop SQL
 - definition of 1-34
 - keys 2-14
 - table registration 1-37
 - types 1-34
 - Database requester
 - See DB requester
 - DB requester
 - application 2-13
 - components 2-16
 - keys 2-14
 - operation 2-14
 - screen 2-15
 - multifile and single-file 2-16
 - DDL (Data Definition Language)
 - definition of 1-35, Glossary-2
 - dictionary 1-36, 1-37
 - schema 1-35, 1-37
 - Decoration 2-11, 3-9, Glossary-2
 - Default screen 2-11, Glossary-2
 - Defaults for Pathmaker application 2-8
 - Definition phase 3-12
 - Design phase 3-13
 - Development phase 3-15
 - Dictionary, DDL 1-36, 1-37
 - Documentation phase 3-16

E

- Encompass 1-34, Glossary-3
 - End user Glossary-3
 - End user screen 1-2, 2-1, 2-9, 2-11, 3-9
 - END-TRANSACTION statement 1-41
 - Enform
 - definition of Glossary-3
 - reports 1-32
 - Enscribe
 - comparison with NonStop SQL 1-38
 - creating a database with 1-35
 - DDL
 - See DDL
 - definition of 1-34
 - keys 2-14
-

F

- File Utility Program
 - See FUP
 - Full screen interface 1-2, 1-28, 3-3
 - Function key list 2-1
 - FUP
 - creating Enscribe files with 1-35
 - definition of Glossary-3
-

G

- General server 2-16
 - Generated source file 2-21
-

H

- Heading Glossary-3
- Help 2-4, 2-12
- HELPUTIL utility 2-12, 3-4
- Hierarchical application structure 2-9
- Host
 - language Glossary-4
 - program Glossary-4
 - variables
 - for a C custom service 2-21
 - for a COBOL85 custom service 2-20

I

Interface for Pathmaker 1-2, 1-28, 1-29, 1-32, 3-3

Interprocess communication message

See IPC

IPC 1-11, 1-16

J

Japanese character set 2-1

JET6530 terminal 2-1, Glossary-4

K

Kanji 2-1, Glossary-4

Keys 2-14

L

Language

See Programming language

Life cycle 3-9

 definition phase 3-12

 design phase 3-13

 development phase 3-15

 documentation phase 3-16

 maintenance phase 3-16

 planning phase 3-14

 production phase 3-16

 setup phase 3-15

 testing phase 3-16

M

Macro language 1-32, Glossary-4

Macros 1-32, 3-3

Maintenance phase 3-16

MAKE utility support 2-21

Manuals

 Pathmaker xiii

 related products xv

Menu requester 2-17, 2-19

Message

- area 2-1
 - from a Pathmaker application 2-8
 - interprocess communication (IPC) 1-11, 1-16
-

N

NonStop SQL

- comparison with Enscribe 1-38
 - definition of 1-34
 - keys 2-14
 - statement generation 1-26
 - table registration 1-37
-

O

Object 3-7

OLTP application

- creating with Pathway 1-11, 1-15
- definition of 1-4, Glossary-5
- functions of 1-6
- processing requirements of 1-8
- running with Pathway 1-18

Online processing 1-4

Online transaction processing application

- See OLTP application

Overlay 2-2

P

Page 2-2

Paging area 2-2

PATHCOM 1-18

Pathmaker

application

- accessing other Pathway applications 2-22
- architecture 2-1
- command files 1-15, 1-25, 1-27, 1-30
- configuration file 1-15, 1-25, 1-27, 1-30
- creation 1-26, 3-9
- custom 2-18
- customization 2-11, 2-12
- data field area 2-1
- data field attributes 2-6

Pathmaker

application (continued)

- DB requester 2-13
- displaying defaults 2-8
- entering data 2-7
- function key actions 2-4
- function key list 2-1
- help text 2-4, 2-12
- hierarchical structure 2-9
- Kanji support 2-1
- life cycle 3-9, 3-12
- message area 2-1
- messages 2-12
- operating 2-2, 2-7
- overlay 2-2
- page 2-2
- paging area 2-2
- prototype 2-13, 3-11
- receiving data and messages 2-8
- screen design 3-9
- screen layout 2-1, 2-11
- screen navigation 2-9
- screen painter 2-11
- screen preview 1-2
- terminal 2-1
- title area 2-1
- application definition language (PMADL) 1-2, 1-29, 1-32, 3-3
- application types
 - See DB requester application and Custom application
- approaches for using 3-9
- benefits 1-1, 1-26
- catalog 1-26, 1-32, 3-2, 3-7, 3-8
- code generators 3-3
- components 3-1
- copy libraries 3-3
- database types 1-37
- DDL dictionary 1-37
- definition of 1-1
- development environment customization 1-2
- full screen interface 1-2, 1-28, 3-3
- how used 1-2
- interface 1-2, 1-28, 1-29, 1-32, 3-3

- Pathmaker (continued)
 - large development team support 1-2
 - macro language 1-32
 - macros 1-32, 3-3
 - management assistance 1-32
 - manual set xiii
 - manuals for related products xv
 - object 3-7
 - PMADL 1-2, 1-29, 1-32, 3-3
 - production application creation 3-9
 - project 3-2, 3-5
 - project catalog 3-7
 - project subvolume 3-8
 - reports 1-32
 - simulation 1-33, 3-10
 - skeleton files 3-3
 - SQL statement generation 1-26
 - subvolume 3-8
 - target application 3-5
 - terminal
 - application developer 3-3
 - end user 2-1
 - TMF requirements 1-41
 - user interface 1-2, 1-28, 1-29, 1-32, 3-3
 - utilities
 - HELPUTIL 2-12
 - PMINSTAL 3-4
 - PMPROJECT 1-40
 - version xiii, 2-22, Glossary-5
- Pathmaker Application Definition Language
 - See PMADL
- PATHMON 1-18
- Pathway
 - accessing other applications from a Pathmaker application 2-22
 - benefits 1-9
 - command files 1-15, 1-25, 1-27, 1-30, Glossary-6
 - components 1-18
 - configuration file 1-15, 1-25, 1-27, 1-30
 - creating an application with 1-11, 1-15
 - creating an application without 1-10
 - definition of 1-9

Pathway (continued)
 how used 1-10
 interprocess communication message (IPC) 1-11, 1-16
 running an application with 1-18
Pathway Open Environment Toolkit
 See POET
Planning phase 3-14
PMADL 1-2, 1-29, 1-32, 3-3
PMINSTAL utility 3-4
PMPROJECT utility 1-40, 3-4
POBJCOD file 1-18
POBJDIR file 1-18
POBJSYM file 1-18
POET, relationship with Pathmaker 2-22
Previewing screens
 See Simulation
Process
 definition of 1-18
 server 1-19
Production application
 See Pathmaker application
Production phase 3-16
Programming language
 for requesters
 See SCREEN COBOL
 for servers 1-13
Project 3-2, 3-5
 catalog 3-7
 subvolume 3-8
Prototype 2-13, 3-11

R

Registered (REG) requester 2-22
Registered (REG) server 2-22
Release number xiii, Glossary-7
Reply message 1-11, 1-16
Reports 1-32
Request message 1-11, 1-16

Requester 1-11
 benefits 1-14
 copy library 2-4, Glossary-7
 DB
 See DB requester
 definition of 1-13
 functions of 1-13
 menu 2-17, 2-19
 programming language
 See SCREEN COBOL
 registered (REG) 2-22
 SCREEN COBOL library 1-18
 skeleton 1-2, 3-3, Glossary-7
 transaction (TRNS) 2-20
Requester-server application
 See OLTP application

S

Schema 1-35, 1-37
SCOBOL
 See SCREEN COBOL
SCOBOLX compiler 1-13
Screen 1-2, 2-1, 2-9, 2-11, 3-9
SCREEN COBOL
 compiler 1-13
 definition of 1-13
 library 1-13, 1-18
Screen interface 1-2, 1-28, 3-3
Screen painter 2-11
SEND statement 1-16
Server 1-11
 benefits 1-14
 criterion for packaging services 1-13, 1-14
 definition of 1-13
 functions of 1-13
 programming languages 1-13
 registered (REG) 2-22
 skeleton 1-2, 3-3, Glossary-8
Server class 1-21
Server process 1-19

Service

- benefits 1-14
 - context-free 1-22
 - criterion for packaging 1-13, 1-14
 - custom 2-20
 - definition of 1-14, 1-19
 - functions of 1-14
 - registered (REG) 2-22
 - skeleton 1-2, 3-3, Glossary-9
 - SQL statement generation 1-26
 - standard 2-16
- Setup phase 3-15
- Shared code 1-32, 3-3
- Simulation 1-2, 1-33, 3-10
- Skeleton
- files 1-2, 3-3
 - language 1-32
- SQL
- See NonStop SQL
- Standard service 2-16
- Statement generation, NonStop SQL 1-26

T

- Table registration, NonStop SQL 1-37
- Target
- application 3-5
 - terminal 2-1, Glossary-9
- TCP 1-18
- TEDIT 1-15, 1-26, 1-28, 1-35, 3-3, Glossary-9
- Terminal
- 3270 2-1, Glossary-10
 - 6520 2-1, 3-3, Glossary-10
 - 6530 2-1, 3-3, Glossary-10
 - application developer 3-3
 - end user 2-1
 - JET6530 2-1, Glossary-4
- Terminal Control Process (TCP) 1-18
- Test configuration file 1-15, 1-25, 1-27, 1-30
- Testing phase 3-16
- Title area 2-1

TMF

- BEGIN-TRANSACTION statement 1-41
- database consistency 1-40
- definition of 1-40
- END-TRANSACTION statement 1-41
- functions of 1-40
- protecting an application with 1-41
- requirements for Pathmaker 1-41, 3-7
- Transaction 1-4
- Transaction (TRNS) requester 2-20
- Transaction copy library Glossary-10
- Transaction Monitoring Facility
 - See TMF

U

- User interface 1-2, 1-28, 1-29, 1-32, 3-3
- Utilities
 - HELPUTIL 2-12, 3-4
 - PMINSTAL 3-4
 - PMPROJECT 1-40

V

- Video attribute 2-6, 2-8, 2-11, Glossary-10

Special characters

- \$RECEIVE file 1-16