

IBM Informix High-Performance Loader

User's Guide

Version 9.4
March 2003
Part No. CT1T3NA

Note:

Before using this information and the product it supports, read the information in the appendix entitled "Notices."

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1996, 2003. All rights reserved.

US Government User Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

Introduction

In This Introduction	3
About This Manual	3
Types of Users	4
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Databases	5
New Features	6
Documentation Conventions	7
Typographical Conventions	7
Icon Conventions	8
Command-Line Conventions	10
Sample-Code Conventions.	12
Screen-Illustration Conventions	12
Additional Documentation	13
Related Reading	15
Compliance with Industry Standards	16
IBM Welcomes Your Comments	16

Chapter 1

High-Performance Loader Overview

In This Chapter	1-3
Overview of Features	1-3
Data Load	1-4
Data Unload	1-6
Loading Modes	1-7
Express Mode	1-7
Deluxe Mode	1-7

HPL Components	1-8
onpload Utility	1-8
ipload Utility	1-8
onpladm Utility	1-9
onpload Database	1-9
Starting onpload	1-9
Relationships Among the Parts of the HPL	1-10
Environment Variables	1-11
DBONPLOAD Environment Variable.	1-12
PLCONFIG Environment Variable.	1-12
PLOAD_SHMBASE Environment Variable	1-13
PLOAD_LO_PATH Environment Variable	1-14
Architecture of the onpload Utility	1-14
Deluxe-Mode Loads.	1-15
Express-Mode Loads	1-17
Unloads	1-20

Chapter 2

Getting Started

In This Chapter	2-3
Data-Load Example	2-3
Preparing to Use ipload	2-3
Creating a File of Data	2-4
Creating a Database	2-4
The ipload Utility	2-5
Starting ipload	2-5
Choosing a Project	2-6
Checking the Defaults	2-6
Load Job Windows	2-7
Load Job Select Window	2-7
Load Job Window	2-10
Device-Array Windows	2-10
Device Array Selection Window	2-11
Device-Array Definition Window	2-12
Format Windows	2-13
Format Views Window.	2-14
Record Formats Window	2-15
Format-Definition Window	2-17

Filter, Discard Records, and Logfile Text Boxes	2-19
Filter Text Box	2-19
Discard Records Text Box	2-19
Logfile Text Box.	2-19
Map Windows	2-20
Map Views Window	2-20
Load Record Maps Window	2-23
Map-Definition Window.	2-23
Load Options Window	2-28
Run Option	2-29
Active Job Window	2-30
Verifying the Data Transfer	2-31
Performing a Level-0 Backup	2-31
Generate Options	2-31
Starting the Example	2-32
Preparing the Unload-Job Window	2-32
Performing the Unload	2-37

Chapter 3

Using the High-Performance Loader Windows

In This Chapter	3-3
Starting ipload	3-3
Using the ipload GUI	3-3
HPL Main Window	3-4
Component-Selection Windows	3-6
Component-Definition Windows.	3-9
Load Job and Unload Job Windows	3-13
Views Windows	3-14
Selection-List Windows	3-18
Message Windows.	3-19
Using the HPL Buttons	3-19
Toolbar Buttons.	3-20
Icon Buttons	3-28
Buttons	3-30
Using the Online Help	3-31
Using UNIX Keyboard Commands to Move the Cursor	3-32

Chapter 4	Defining Projects	
	In This Chapter	4-3
	Project Organization	4-3
	Projects Window	4-6
	Creating a New Project.	4-7
	Selecting a Project	4-7
Chapter 5	Configuring the High-Performance Loader	
	In This Chapter	5-3
	Configuring ipload	5-3
	Selecting a Database Server	5-3
	Using the Connect Server Window.	5-4
	Creating the onpload Database	5-5
	Modifying the onpload Defaults	5-5
	Defaults Window.	5-5
	Changing the onpload Defaults.	5-7
	Selecting a Driver	5-8
	Drivers Window	5-9
	Using the Drivers Window	5-10
	Modifying the Machine Description	5-11
	Machines Window	5-12
	Using the Machines Window.	5-13
Chapter 6	Defining Device Arrays	
	In This Chapter	6-3
	Device Arrays	6-3
	Using Multiple Devices in a Device Array	6-3
	Using the Device-Array Selection Window	6-4
	Using the Device-Array Definition Window	6-6
Chapter 7	Defining Formats	
	In This Chapter	7-3
	Formats	7-3
	Fixed-Length Records	7-4
	Creating a Fixed Format	7-5
	Editing a Format	7-10
	Creating a Fixed Format That Uses Carriage Returns	7-12
	Creating a Fixed Format That Includes BYTE or TEXT Data	7-13
	Creating a Fixed Format That Includes Ext Type or Simple LO Data	7-16

Delimited Records	7-19
Creating a Delimited Format	7-19
Creating a Delimited Format That Includes BYTE or TEXT Data	7-21
Creating a Delimited Format with Extended Data Types.	7-22
COBOL Records	7-24
Creating a COBOL Format	7-25
Picture and Usage Descriptions	7-26
Other Formats	7-27
Fast Format	7-27
Fast Job	7-27
Format Options	7-28
Modifying Fixed and COBOL Formats.	7-28
Modifying Delimited-Format Options	7-30
Format Views Window	7-32

Chapter 8

Defining Queries

In This Chapter	8-3
Queries	8-4
Creating a Query	8-4
Using the Table Button	8-7
Editing the WHERE Clause.	8-11
Editing a Query	8-12
Exporting and Importing Queries	8-13
Importing a Query.	8-13
Exporting a Query.	8-16
Database Views Window	8-17

Chapter 9

Defining Maps

In This Chapter	9-3
Load and Unload Maps	9-3
Using the Map-Definition Window	9-4
Creating a Load Map	9-6
Unload Maps	9-10
Creating an Unload Map	9-10
Unloading Data Using Functions.	9-13
Mapping Options	9-14
Using Mapping Options.	9-15
Setting the Mapping Options	9-16

Editing Options	9-18
Using the Delete Button	9-18
Using the Find Button	9-19
Using the Specs Button.	9-21
Map Views Window	9-22

Chapter 10

Defining Filters

In This Chapter	10-3
Using a Filter	10-4
Creating a Filter	10-5
Editing a Filter	10-9
Filter Views	10-11
Filters with Code-Set Conversion	10-12

Chapter 11

Unloading Data from a Database

In This Chapter	11-3
Components of the Unload Job	11-3
Choosing the Database Server	11-4
Running Multiple Jobs	11-4
Unload Job Windows	11-5
Creating an Unload Job	11-5
Running the Unload Job	11-8
Specifying to Write to the End of the Tape	11-10
Using the Command-Line Information	11-11
Changing the Unload Options	11-12
Editing an Unload Job	11-14
Generate Options	11-15

Chapter 12

Loading Data to a Database Table

In This Chapter	12-3
Components of the Load Job	12-3
Choosing the Database Server	12-4
Running Multiple Jobs	12-4
Preparing User Privileges and the Violations Table	12-5
Load Job Windows	12-7
Creating a Load Job	12-8
Running the Load Job	12-10
Specifying to Read to the End of the Tape	12-12
Using the Command-Line Information	12-13

Changing the Load Options	12-14
Editing a Load Job	12-16
Generate Options	12-16

Chapter 13 Generate Options

In This Chapter	13-3
Types of Generate Tasks	13-3
Generating from the Load Job Window	13-4
Using the Autogenerate Load Components Window	13-4
Generating from the Unload Job Window	13-6
Using the Autogenerate Unload Components Window	13-6
Generating from the Components Menu	13-10
Generate Window	13-10
Generating Load and Unload Components	13-12
Using the No Conversion Job Option	13-14

Chapter 14 Browsing

In This Chapter	14-3
Browsing Options	14-3
Previewing Data-File Records	14-3
Reviewing Records That the Conversion Rejected	14-6
Viewing the Violations Table	14-7
Viewing the Status of a Load Job or Unload Job.	14-9

Chapter 15 Managing the High-Performance Loader

In This Chapter	15-3
Modes	15-3
Deluxe Mode	15-4
Express Mode	15-5
How Express Mode and Deluxe Mode Work.	15-6
Violations	15-9
Rejected Records from the Input File	15-9
Constraint Violations	15-10
Viewing Error Records	15-10
Performance	15-10
Configuration Parameters	15-11
Express-Mode Limitations	15-12
onstat Options for onpload	15-12
Devices for the Device Array	15-13

Usage Models	15-13
Performance Hints	15-16
Limitation When Using the Excalibur Text DataBlade Module	
Indexes	15-19

Chapter 16 The onpload Utility

In This Chapter	16-3
Understanding the onpload Utility	16-3
Starting the onpload Utility	16-3
Using the onpload Utility	16-4
The onpload Utility Syntax	16-4

Chapter 17 The onpladm Utility

In This Chapter	17-5
Description of the onpladm Utility	17-5
The onpladm Utility Features	17-5
Specification-File Conventions	17-6
Error Handling	17-9
Defining Jobs	17-10
Creating Jobs	17-10
Modifying a Job by Using a Detailed Specification File	17-21
Describing a Job	17-22
Listing All Jobs in a Project	17-23
Running a Job	17-24
Deleting a Job	17-25
Defining Device Arrays	17-26
Creating a Device Array	17-26
Modifying a Device Array	17-27
Describing a Device Array	17-27
Listing Project Device Arrays	17-28
Deleting a Device Array	17-29
Defining Maps	17-29
Creating Maps.	17-29
Deleting a Map	17-37
Describing a Map	17-38
Modifying a Map Using A Detailed Specification File	17-38
Listing All Maps in a Project	17-39
Defining Formats	17-39
Creating a Format	17-39
Modifying a Format Using a Specification File.	17-43

Describing a Format	17-44
Listing all Formats in a Project	17-44
Deleting a Format	17-46
Defining Queries	17-46
Creating a Query	17-46
Modifying a Query	17-47
Describing a Query	17-48
Listing all Queries in a Project	17-48
Deleting a Query	17-50
Defining Filters	17-50
Creating a Filter.	17-50
Modifying a Filter	17-51
Describing a Filter	17-52
Listing all Filters in a Project	17-52
Deleting a Filter.	17-54
Defining Projects	17-55
Creating a New Project	17-55
Running All Jobs in a Project	17-56
Listing all Projects	17-57
Deleting a Project	17-57
Defining Machine Types	17-58
Creating a Machine Type	17-58
Modifying a Machine Type	17-59
Describing a Machine.	17-60
Listing all Existing Machine Types	17-61
Deleting a Machine Type	17-61
Defining Database Operations	17-62
Creating a Database Project.	17-62
Configuring Target-Server Attributes	17-65

Appendix A	onpload Database
Appendix B	High-Performance Loader Configuration File
Appendix C	Picture Strings
Appendix D	Match Condition Operators and Characters
Appendix E	Custom Conversion Functions

Appendix F	onstat -j Option
Appendix G	HPL Log-File and Pop-Up Messages
Appendix H	Custom Drivers
Appendix I	Running Load and Unload Jobs on Windows Computers
Appendix J	Notices
	Index

Introduction

In This Introduction	3
About This Manual.	3
Types of Users	4
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Databases	5
New Features.	6
Documentation Conventions	7
Typographical Conventions	7
Icon Conventions	8
Comment Icons	8
Feature, Product, and Platform Icons	9
Compliance Icons	9
Command-Line Conventions	10
Sample-Code Conventions	12
Screen-Illustration Conventions	12
Additional Documentation	13
Related Reading	15
Compliance with Industry Standards	16
IBM Welcomes Your Comments	16

In This Introduction

This introduction provides an overview of the information in this manual and describes the conventions it uses.

About This Manual

This manual describes how to use the High-Performance Loader (HPL) to load and unload large quantities of data efficiently to or from an Informix database. It describes the architecture of HPL, the **onpload** utility that allows you to control HPL from a script, the **onpload** database that maintains information about load and unload jobs that you have prepared, the **ipload** graphical user interface (GUI), and the **onpladm** utility. The **ipload** utility is a UNIX application that helps users prepare load and unload jobs for both UNIX and Windows. The **onpladm** utility is a command-line version of the **ipload** utility that operates on both UNIX and Windows.

The first chapter introduces the HPL, provides a general overview of the tasks that the HPL performs, describes the architecture of the HPL, and includes two tutorial examples that take you through the process of loading and unloading data.

The second chapter introduces the graphical user interface, the **ipload** utility, which you can use to set the parameters for the HPL.

Subsequent chapters give details about developing the **onpload** database by using the individual components of the **ipload** and **onpladm** utilities.

[Appendix A, “onpload Database,”](#) describes the tables of the **onpload** database.

Types of Users

This manual is written for the following users:

- Database administrators
- Database server administrators

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with computer programming
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, see *IBM Informix Dynamic Server Getting Started Guide* for a list of supplementary titles.

Software Dependencies

This manual assumes that you are using IBM Informix Dynamic Server, Version 9.40.

Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

This manual assumes that you use the U.S. 8859-1 English locale as the default locale. The default is **en_us.8859-1** (ISO 8859-1) on UNIX platforms or **en_us.1252** (Microsoft 1252) for Windows environments. This locale supports U.S. English format conventions for dates, times, and currency, and also supports the ISO 8859-1 or Microsoft 1252 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

Demonstration Databases

The DB-Access utility, which is provided with your IBM Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix manuals are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines. ♦

For information about how to create and populate the demonstration databases, see the *IBM Informix DB-Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **\$INFORMIXDIR/bin** directory on UNIX platforms and in the **%INFORMIXDIR%\bin** directory in Windows environments.

New Features

For a comprehensive list of new database server features, see the *Getting Started Guide*.

High-Performance Loader (HPL) in Version 9.40 of Dynamic Server adds support for the following new features:

- Write to or read from the tape until the end of the device. When the end of the device is reached, you are prompted to provide another tape. You can specify this feature in the following ways:
 - Check the **Write/read to/from tape until end of device** checkbox on the Load Select Job or Unload Select Job windows. For more information, see [“Specifying to Write to the End of the Tape” on page 11-10](#) or [“Specifying to Read to the End of the Tape” on page 12-12](#).
 - Specify the **-Z** option with the **onpload** or **onpladm run job** commands. For more information, see [“The onpload Utility Syntax” on page 16-4](#) or [“Running a Job” on page 17-24](#).
- **HPL_DYNAMIC_LIB_PATH** configuration parameter sets the location of the shared library file that can contain custom drivers and conversion functions. For more information, see [“HPL_DYNAMIC_LIB_PATH” on page B-8](#).
- **HPLAPIVERSION** configuration parameter specifies whether to use custom conversion or driver functions with three or four arguments. Using four arguments allows different lengths for data in the input and output buffers. For more information, see [“HPLAPIVERSION” on page B-7](#).
- New log file and pop-up messages for the preceding new features.

This manual now contains the **onpladm** utility documentation that was previously supplied as HTML files. For more information, see [Chapter 17, “The onpladm Utility.”](#)

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Icon conventions
- Command-line conventions
- Sample-code conventions
- Screen-illustration conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
<i>italics</i> italics <i>italics</i>	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface boldface	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
monospace <i>monospace</i>	Information that the product displays and information that you enter appear in a monospace typeface.

(1 of 2)



Convention	Meaning
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
◆	This symbol indicates the end of one or more product- or platform-specific paragraphs.
→	This symbol indicates a menu item. For example, “Choose Tools→Options ” means choose the Options item from the Tools menu.

(2 of 2)

Tip: When you are instructed to “enter” characters or to “execute” a command, immediately press RETURN after the entry. When you are instructed to “type” the text or to “press” other keys, no RETURN is required.

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.



Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
	Warning:	Identifies paragraphs that contain vital instructions, cautions, or critical information
	Important:	Identifies paragraphs that contain significant information about the feature or operation that is being described
	Tip:	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Feature, Product, and Platform Icons


Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
	Identifies information that relates to the IBM Informix Global Language Support (GLS) feature
	Identifies information that is specific to UNIX platforms

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

Compliance Icons

Compliance icons indicate paragraphs that provide guidelines for complying with a standard.

Icon	Description
	Identifies information that is specific to an ANSI-compliant database

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

Command-Line Conventions

This section defines and illustrates the format of commands that are available in IBM Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled IBM Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters.
<i>variable</i>	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name, or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as .sql or .cob , might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products.
(. , ; + * - /)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.

(1 of 2)

Element	Description
' '	Single quotes are literal symbols that you must enter as shown.
<div>Privileges</div> <div>p. 5-17</div> <div>Privileges</div>	A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.
— ALL —	A shaded option is the default action.
→ →	Syntax within a pair of arrows indicates a subdiagram.
—	The vertical line terminates the command.
-f — OFF — ON —	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
— , — — variable —	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
— , — — 3 size —	A gate (/3) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. You can specify <i>size</i> no more than three times within this statement segment.

(2 of 2)

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single IBM Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
      WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.



Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Screen-Illustration Conventions

The illustrations in this manual represent a generic rendition of various windowing environments. The details of dialog boxes, controls, and windows were deleted or redesigned to provide this generic look. Therefore, the illustrations in this manual depict your product interface a little differently than the way it appears on your screen.

Additional Documentation

IBM Informix Dynamic Server documentation is provided in a variety of formats:

- **Online manuals.** The IBM Informix OnLine Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/> contains manuals for your use. This Web site enables you to print chapters or entire books.
- **Online help.** This facility provides context-sensitive help, an error message reference, language syntax, and more.
- **Documentation notes and release notes.** Documentation notes, which contain additions and corrections to the manuals, and release notes are located in the directory where the product is installed.

Please examine these files because they contain vital information about application and performance issues. The following table describes these files.

UNIX

On UNIX platforms, the following online files appear in the \$INFORMIXDIR/release/en_us/0333 directory.

Online File	Purpose
ids_hpl_docnotes_9.40.html	The documentation notes file for your version of this manual describes topics that are not covered in the manual or that were modified since publication.
ids_unix_release_notes_9.40.html	The release notes file describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
ids_machine_notes_9.40.txt	The machine notes file describes any special actions that you must take to configure and use IBM Informix products on your computer. Machine notes are named for the product described.



Windows

The following items appear in the **Informix** folder. To display this folder, choose **Start→Programs→Informix Dynamic Server→ Documentation Notes or Release Notes** from the taskbar.

Program Group Item	Description
Documentation Notes	This item includes additions or corrections to manuals with information about features that might not be covered in the manuals or that have been modified since publication.
Release Notes	This item describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.

Machine notes do not apply to Windows platforms. ♦

- IBM Informix software products provide ASCII files that contain all of the error messages and their corrective actions. For a detailed description of these error messages, refer to *IBM Informix Error Messages* in the IBM Informix Online Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/>.

To read the error messages on UNIX, you can use the **finderr** command to display the error messages online. ♦

To read error messages and corrective actions on Windows, use the **Informix Error Messages** utility. To display this utility, choose **Start→Programs→Informix** from the taskbar. ♦

UNIX

Windows

Related Reading

For a list of publications that provide an introduction to database servers and operating-system platforms, refer to your *Getting Started Guide*.

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

IBM Welcomes Your Comments

To help us with future versions of our manuals, let us know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of your manual
- Any comments that you have about the manual
- Your name, address, and phone number

Send electronic mail to us at the following address:

`docinf@us.ibm.com`

This address is reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact Customer Services.

High-Performance Loader Overview

In This Chapter	1-3
Overview of Features	1-3
Data Load	1-4
Data Unload	1-6
Loading Modes	1-7
Express Mode	1-7
Deluxe Mode	1-7
HPL Components	1-8
onpload Utility	1-8
ipload Utility	1-8
onpladm Utility	1-9
onpload Database	1-9
Starting onpload	1-9
Relationships Among the Parts of the HPL	1-10
Environment Variables	1-11
DBONPLOAD Environment Variable	1-12
PLCONFIG Environment Variable	1-12
PLOAD_SHMBASE Environment Variable	1-13
Avoiding Shared-Memory Collision	1-13
Setting the PLOAD_SHMBASE Environment Variable	1-13
PLOAD_LO_PATH Environment Variable	1-14
Architecture of the onpload Utility	1-14
Deluxe-Mode Loads	1-15
Threads That the onpload Utility Uses	1-16
Threads That the Database Server Uses	1-17

Express-Mode Loads	1-17
Unloads	1-20

In This Chapter

This chapter introduces the High-Performance Loader (HPL), discusses the theory of the HPL, provides a general overview of the tasks that the HPL performs, and describes the architecture of the HPL.

Overview of Features

The HPL is a feature of the database server that allows you to load and unload large quantities of data efficiently to or from a database. The HPL lets you exchange data with tapes, data files, and programs, and converts data from these sources into a format compatible with Informix databases. The HPL also allows you to manipulate and filter the data as you perform load and unload operations.

The HPL includes the following features:

- The HPL supports COBOL, ASCII, multibyte, delimited, or binary data.
- The HPL can load and unload data of a different GLS locale from that of the database server.
- The client/server architecture of the HPL lets you use **ipload** to manage the **onpload** database on any database server on your network.
- The **ipload** utility provides a **Generate** option that lets you automatically generate the HPL components that are required for a load or unload job.

- The database-load feature lets you update your databases with data from any of the supported file types, while allowing you to control the format and selection of records from the input files. Data can be loaded from or unloaded to files, tapes, or application pipes (for UNIX), or to any combination of these three device types.
- The HPL provides synonym support for tables that are valid for the local database server. You can use synonyms for both the load and unload operations.
- The HPL provides support for unloading data with a query that accesses a view in its `SELECT` statement.
- The load and unload operations run in the background of your multitasking operating system. Once the operation begins, you can continue to use **ipload** to perform other functions.
- The **ipload** utility provides context-sensitive online help. The online help also includes a glossary.
- The **onpladm** utility is a command-line version of the **ipload** utility.
- Any database server on your network can use the **onpload** database, which contains parameters and controls that the HPL uses. This accessibility allows centralized management of your load and unload controls. These parameters and controls include the HPL components such as formats, maps, and projects.
- The HPL supports loading of raw tables in express mode.

Data Load

The *data-load* process reads a source data file, converts the data to a different format, and inserts the converted data into a database table. The source data can come from one or more of the following sources:

- Files
- Tapes
- Pipes (application-generated data) ♦

The HPL supports load and unload data greater than 2 GB to files and tapes.

During conversion, the source data is often manipulated so that the converted data displays different characteristics. Examples of this manipulation include:

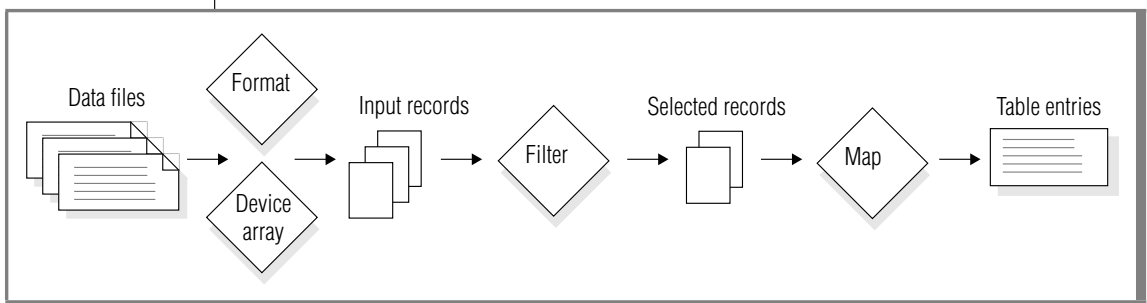
- Changing lowercase letters to uppercase letters
- Loading default values, loading certain table columns, or replacing nulls
- Masking the data to include only part of a value
- Converting from one data type to another, such as conversion of a numeric string to a float
- Converting from the code set of one locale to the code set of another locale ♦

When you prepare to run a data load using the HPL, you describe the actions that the HPL must take by defining a set of metadata *components*. The components describe different aspects of the load process. [Figure 1-1](#) shows the data load process. The HPL uses information from:

- The device array to find the set of the source-data files
- The format to define the data types and layout of the source data
- The filter to select the records from the source data that should be written to the database table
- The map to modify and reorganize the data

The **ipload** utility helps you prepare the components. [Chapter 12, “Loading Data to a Database Table,”](#) addresses the process of loading a file to a database.

Figure 1-1
The Data-Load Process



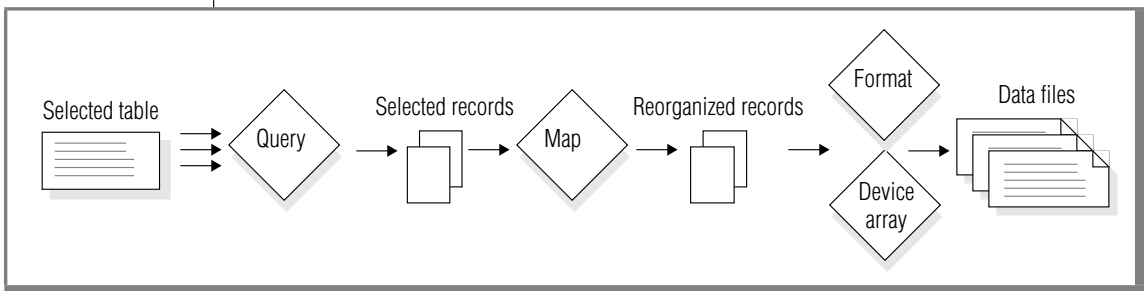
Data Unload

The *data-unload* process is essentially the same as the load process, but in reverse. The data-unload process extracts the source data from one or more database tables; converts the data to a new format; and writes the converted data to a file, tape, or on UNIX to a pipe (application). As in a load, you can manipulate the data from a database table so that the converted data displays different characteristics.

Figure 1-2 shows how the components of the HPL affect the data as it moves from a database to data files during the unload process. The HPL uses:

- The query to select records from the database
- The map to reorganize or modify the selected records
- The format to prepare the records for writing out to the data files
- The device array to find the location of the data files

Figure 1-2
The Data-Unload Process



The HPL uses the same components for an unload as for a load, with one exception. For an unload, **ipload** creates a Structured Query Language (SQL) query that extracts selected data from the table. As with a load, unload components are grouped together into an unload job. Unload jobs can be saved, retrieved, and rerun as many times as necessary. Unload jobs can be grouped together with load jobs in the same project.

For a description of the steps involved in an unload, see [Chapter 11, “Unloading Data from a Database.”](#)

Loading Modes

The HPL offers two load modes, deluxe mode and express mode. The express mode is faster, and the deluxe mode is more flexible. For a detailed comparison of the express and deluxe modes, see [Chapter 15, “Managing the High-Performance Loader.”](#)

Express Mode

Express-mode loads are significantly faster than deluxe-mode loads, but less flexible. In express mode you cannot update the table or read the new data entries until the load is complete. The express mode disables indexes, constraints, and triggers during the load. After the load, HPL rebuilds and reenables indexes; evaluates and reenables constraints, if possible; and reenables triggers. (HPL does not evaluate triggers with respect to the loaded data.)

You must perform a level-0 backup after an express-mode load.

Deluxe Mode

Deluxe-mode loads are not as fast as express-mode loads, but are more flexible. In deluxe mode, you can access and update the table that is being loaded. The deluxe mode updates indexes, performs constraint checking, and evaluates triggers as data is inserted into the table.

HPL Components

The HPL consists of the **onpload** utility, **ipload**, and the **onpload** database.

onpload Utility

The **onpload** utility has the following features:

- Converts, filters, and moves data between a database and a storage device
- Uses information from the **onpload** database to run the load and unload jobs and to convert the data
- Records information during a load about data records that do not meet the load criteria



Tip: The **onpload** utility must be located on the same network as the **onpload** database and on the same server as the target database.

You can start **onpload** using **ipload** or from the command line.

For more information about **onpload**, see [“Architecture of the onpload Utility” on page 1-14](#), and [Chapter 16, “The onpload Utility.”](#)

ipload Utility

The **ipload** utility is a UNIX-based graphical user interface that has the following features:

- Creates and manages the **onpload** database
- Creates and stores information for **onpload**
- Lets you create, edit, and group the components of the load and unload jobs
- Stores information about the load components in the database

For more information about **ipload**, see [“The ipload Utility” on page 2-5](#), [“Starting ipload” on page 3-3](#), and [“Configuring ipload” on page 5-3](#).

You can use **ipload** to manage **onpload** databases on both UNIX and Windows. For more information about managing **onpload** databases on Windows, see [Appendix I, “Running Load and Unload Jobs on Windows Computers.”](#)

onpladm Utility

The **onpladm** utility is a command-line utility for both UNIX and Windows with the same functionality as **ipload**.

For more information about **onpladm**, see [Chapter 17, “The onpladm Utility.”](#)

onpload Database

The **onpload** database has the following features:

- Contains information that the **onpload** utility requires to perform data loads and unloads
- Can reside on any database server on your network
- Can be used by any **onpload** utility on the same network.

In contrast, the target database must be located on the same server as the **onpload** utility.

Starting onpload

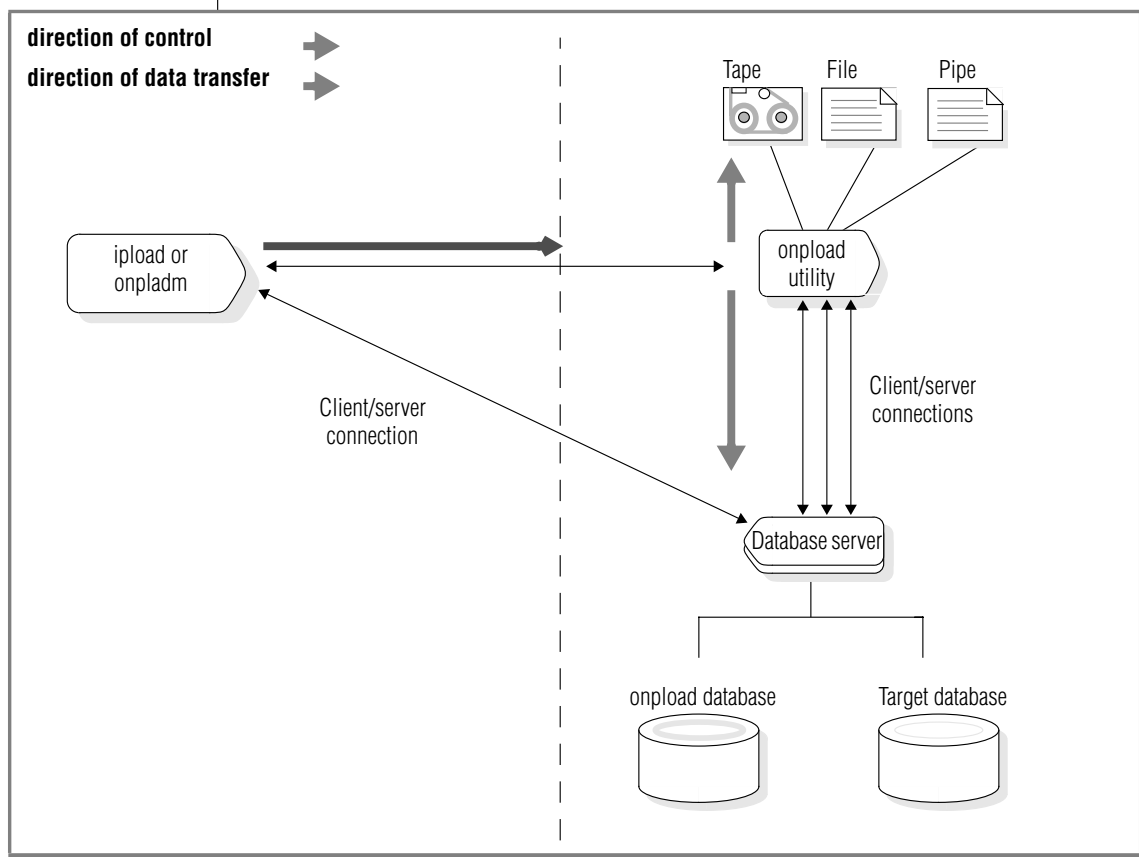
You can start **onpload** from either **ipload** or the **onpload** command line.

When you start **onpload** from **ipload**, **onpload** and **ipload** communicate using a socket connection. **onpload** executes completely independently from **ipload** (see the information to the right of the dashed line in [Figure 1-3](#)).

Relationships Among the Parts of the HPL

Figure 1-3 shows the relationships among the parts of the HPL. The **ipload** utility connects to the database server to populate the **onpload** database. The **ipload** utility controls the **onpload** utility, which uses the multithreaded architecture to make multiple connections to the database server and multiple I/O connections to tapes, files, or pipes.

Figure 1-3
Relationships Among the Parts of the HPL



The information to the right of the dashed line in Figure 1-3 shows **onpload** loading or unloading data with no interaction from **ipload**.

Environment Variables

The HPL is part of the database server, so you must start the database server before you use the HPL. Before you start the database server, you must set these environment variables:

- **INFORMIXDIR**
- **ONCONFIG**
- **INFORMIXSERVER**
- **LD_LIBRARY_PATH**

The **INFORMIXDIR**, **ONCONFIG**, and **INFORMIXSERVER** environment variables are documented in the *Administrator's Guide*.

Some computers use the **LD_LIBRARY_PATH** environment variable for shared libraries. See the machine notes for information about **LD_LIBRARY_PATH**.

In addition to the environment variables that you must always set when you use the database server, the following environment variables refer to the HPL:

- **DBONPLOAD**
- **PLCONFIG**
- **PLOAD_SHMBASE**
- **PLOAD_LO_PATH**



Tip: To maximize available memory and scan resources, HPL automatically sets the **PDQPRIORITY** environment variable to 100, if it is not already set. If the **PDQPRIORITY** environment variable is set, HPL uses that value. If the **PDQPRIORITY** environment variable is set to 0, then HPL cannot unload multiple devices. For more information on this environment variable, see the “IBM Informix Guide to SQL: Reference.”

DBONPLOAD Environment Variable

Each database server can have only one active **onpload** database. In most cases, you can use the default **onpload** database, which is named **onpload**.

If you choose to prepare multiple databases to hold different types of control information, you must set the **DBONPLOAD** environment variable before you can run a load or unload.

To prepare multiple onpload databases with ipload

1. Set the **DBONPLOAD** environment variable to the name of the alternative **onpload** database.
2. Restart **ipload**.
3. Use **ipload** to prepare the alternative database.

If you do not use an alternative **onpload** database, you do not need to set **DBONPLOAD**.

PLCONFIG Environment Variable

The default configuration file for the **onpload** utility is **plconfig.std**. The configuration file always resides in the **\$INFORMIXDIR/etc** directory. To use an alternative configuration file, you must set the **PLCONFIG** environment variable to the name of the alternative **onpload** configuration file. If you use **plconfig.std**, you do not need to set **PLCONFIG**.

The **onpload** configuration file is described in [Appendix B, “High-Performance Loader Configuration File.”](#)



PLOAD_SHMBASE Environment Variable

The **PLOAD_SHMBASE** environment variable allows you to specify shared-memory address attachments specifically for **onpload** processes. You can set the **PLOAD_SHMBASE** environment variable to avoid shared-memory collisions between **onpload** and the database server. It is suggested that you do not set this variable, but rather allow the HPL to specify the attachments.

Tip: To use the **PLOAD_SHMBASE** environment variable, you must start **onpload** from the command line, not from within **ipload**.

Avoiding Shared-Memory Collision

Both the database server and **onpload** allocate shared memory. When **onpload** starts up, the database server allocates shared memory for buffers for **onpload** processes. **onpload** also allocates shared memory for its internal use. Because of the dynamic nature of shared-memory allocations, shared-memory collisions can occur between **onpload** and the database server. If this occurs, the **onpload** job can also fail and error messages are sent to the **onpload** log file or the database server log file.

To verify if a collision has occurred, use the **onstat -g seg** option. Check for overlap between the shared-memory segments that the database server is using and the SHMBASE reported in the **onpload** log file. For more information on the **onstat -g seg** option, see the *Administrator's Guide* and the *IBM Informix Dynamic Server Performance Guide*.

Setting the PLOAD_SHMBASE Environment Variable

To override the initial shared-memory allocation in case of a shared-memory collision between **onpload** and the database server, set the **PLOAD_SHMBASE** environment variable to a value much higher or much lower than that for the shared memory that the database server uses.

PLOAD_LO_PATH Environment Variable

The **PLOAD_LO_PATH** environment variable allows you to specify the location of the CLOB or BLOB output files. If this environment variable is not set, the database server puts the output files in **/tmp**.

For more information on environment variables, see the *IBM Informix Guide to SQL: Reference* and the *IBM Informix Dynamic Server Performance Guide*.

Architecture of the onpload Utility

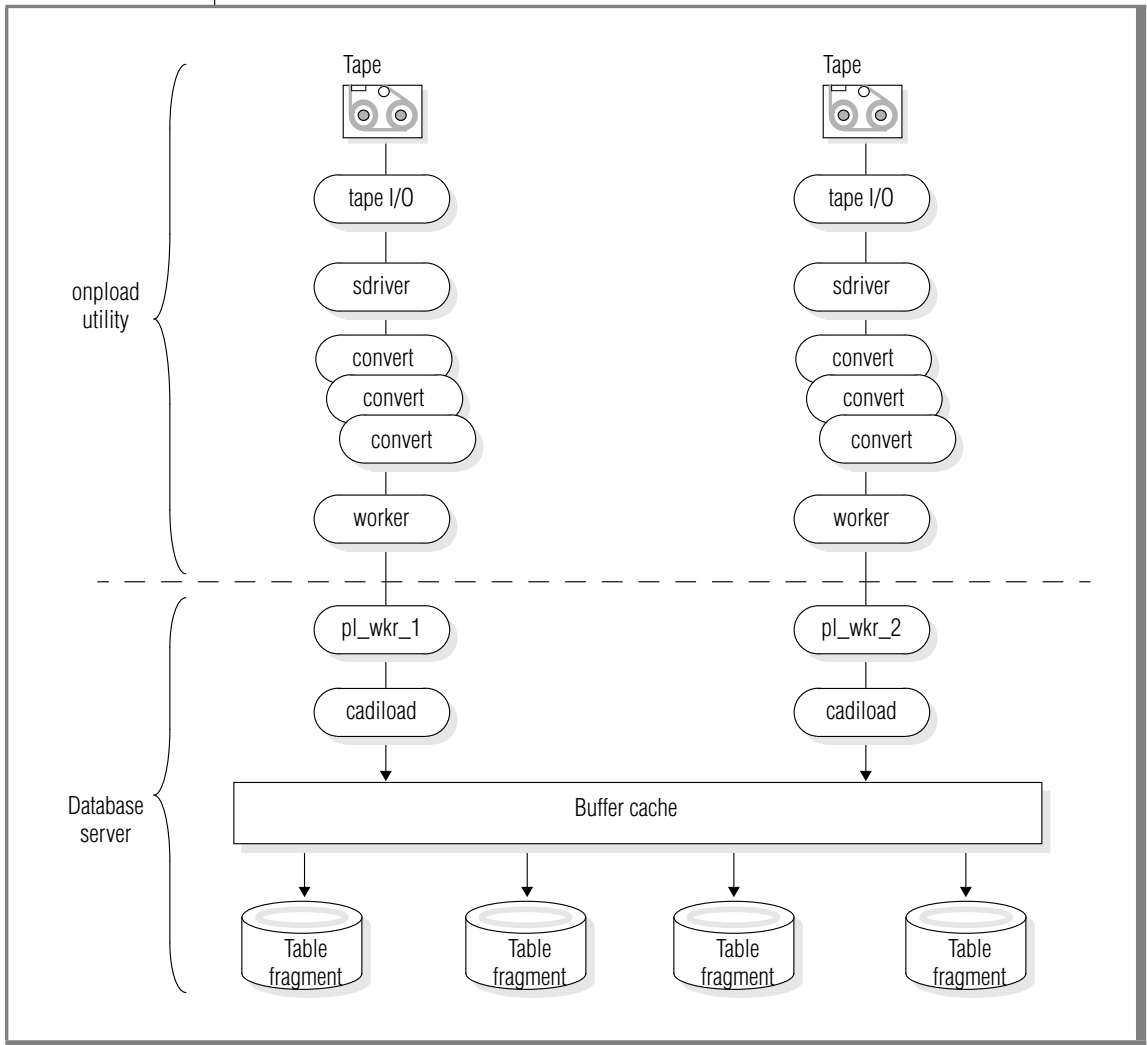
The **ipload** utility is the interface that allows you to prepare the parameters that the **onpload** utility uses. The **onpload** utility, which is a client application that attaches to the database server, actually loads and unloads the data.

The **onpload** utility can take advantage of parallel processing to perform both I/O and data conversion as efficiently as possible because it uses the same multithreading architecture that the database server uses. Multithreading is described in the *Administrator's Guide*. The following sections describe how **onpload** uses multithreading for deluxe loads, express loads, and unloads.

Deluxe-Mode Loads

Figure 1-4 shows the threads that **onpload** uses in a deluxe-mode load process. In deluxe mode, data is subject to the same constraints as if you were loading the data using SQL INSERT statements.

Figure 1-4
A Deluxe-Mode Load



Threads That the onpload Utility Uses

The **onpload** utility starts the following threads:

- **tape I/O threads**

The **onpload** utility starts one **tape I/O** thread for each tape device. It reads data from the tape device asynchronously.

onpload starts a similar thread for piped output. ♦

In the case of disk file input, **onpload** uses the multithreading AIO subsystem instead of a dedicated I/O thread.

- **sdriver threads**

The **worker** threads control I/O from input files. They handle device abstraction for the different device types handled. The **driver** threads also are responsible for passing out records from the input and passing records to the converters.

- **convert threads**

The **onpload** utility starts one or more **convert** threads for each device. These threads perform conversions on the input data such as uppercase to lowercase conversion or code-set conversion.

- **worker threads**

The **onpload** utility starts one **worker** thread for each input device. These threads communicate with the database server. The main responsibility of the **worker** thread is to pass data to the database server.

To see the status of the **onpload** threads, you must use the **-j** option of the **onstat** utility. This option is documented in [Appendix F](#).

Threads That the Database Server Uses

The database server uses the following threads to insert the data into the database:

- **pl_wkr** threads

Each **worker** thread of the **onpload** utility is paired with a **pl_wkr** thread in the database server. These threads receive the data from **onpload**.

In a utility that shows the database server status, the **pl_wkr** threads are named **pl_wkr_1**, **pl_wkr_2**, **pl_wkr_3**, and so forth.

- **cadiload** threads

The **cadiload** threads are the **insert** threads. The **insert** threads perform a normal insert into the database, just as if you were using an INSERT statement. The SQL optimizer governs the method that is used for inserting the data.

Express-Mode Loads

[Figure 1-5 on page 1-19](#) shows a single express-mode load process. In express mode, the data is inserted directly into an extent without any evaluation of objects such as constraints, indexes, or triggers.

The **onpload** utility behaves the same way during express loads and deluxe loads, as described in [“Threads That the onpload Utility Uses” on page 1-16](#). However, the database server behaves differently during an express load. The **pl_wkr** threads pass the data to **stream** threads (also called *fragmenter* threads) that decide where the data should be stored. The *fragmenter* threads pass the data to an exchange that distributes the data to **setrw** threads. The **setrw** threads write table rows to disk a page at a time, bypassing the buffer cache.

The number of input devices can be different from the number of table fragments. The exchange operator handles multiplexing of data. The data is processed in parallel with respect to the data read from the device array and also with respect to the data written out to table fragments on separate disks. Pipeline parallelism is also present in the data flow from input devices to table fragments on disk. Parallelism is the main mechanism for achieving high performance.

During express-mode load, the database server writes the data to new extents on disk, but those extents are not yet part of the table. At the end of an express-mode load, the database server adds the new extents to the table.

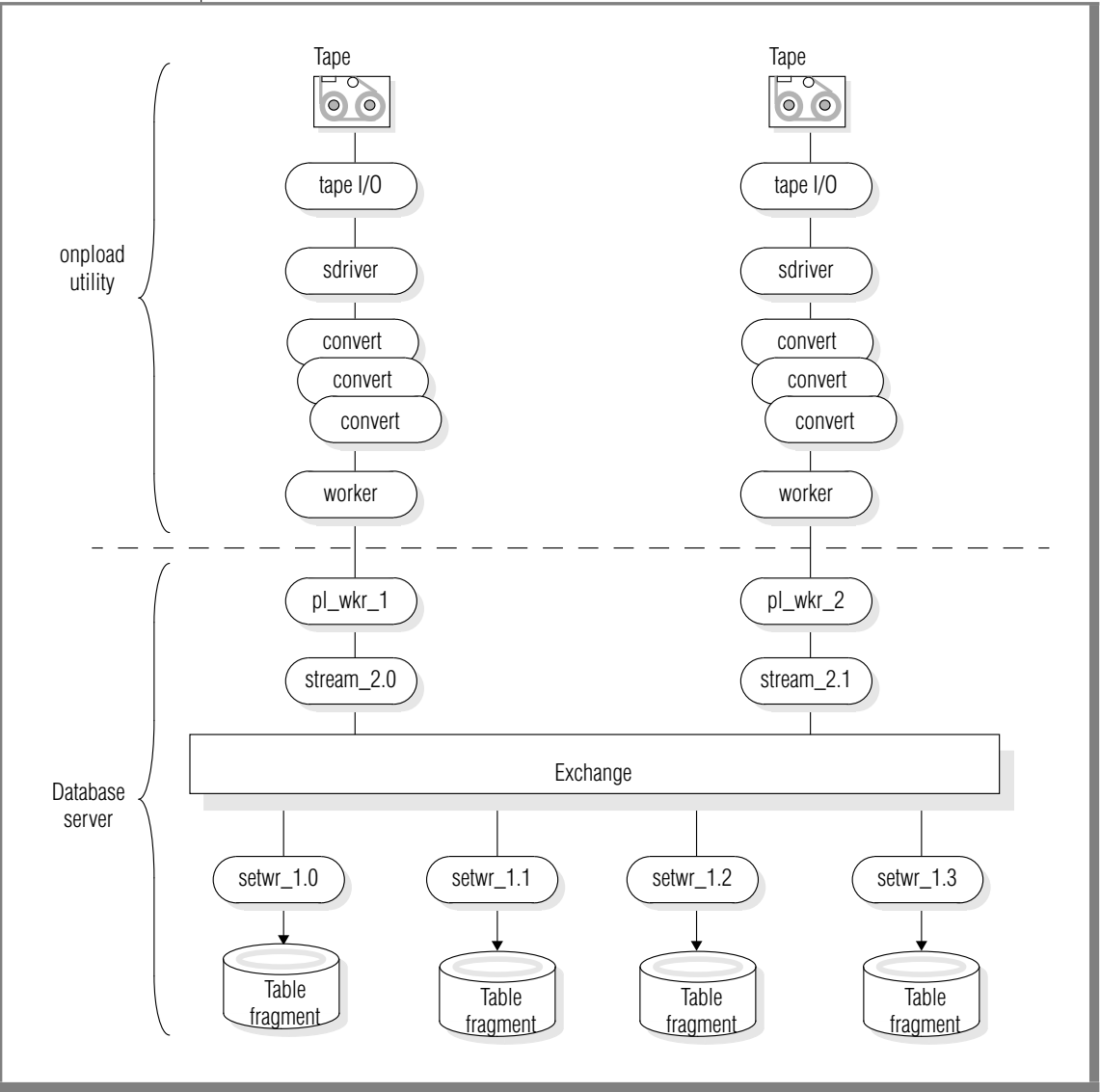
After the express-mode load, you must perform a level-0 backup before you can write to the target database. If you try to write to the table before you perform a level-0 backup, the database server issues ISAM error -197, as follows:

```
Partition recently appended to; can't open for write or logging.
```

ANSI

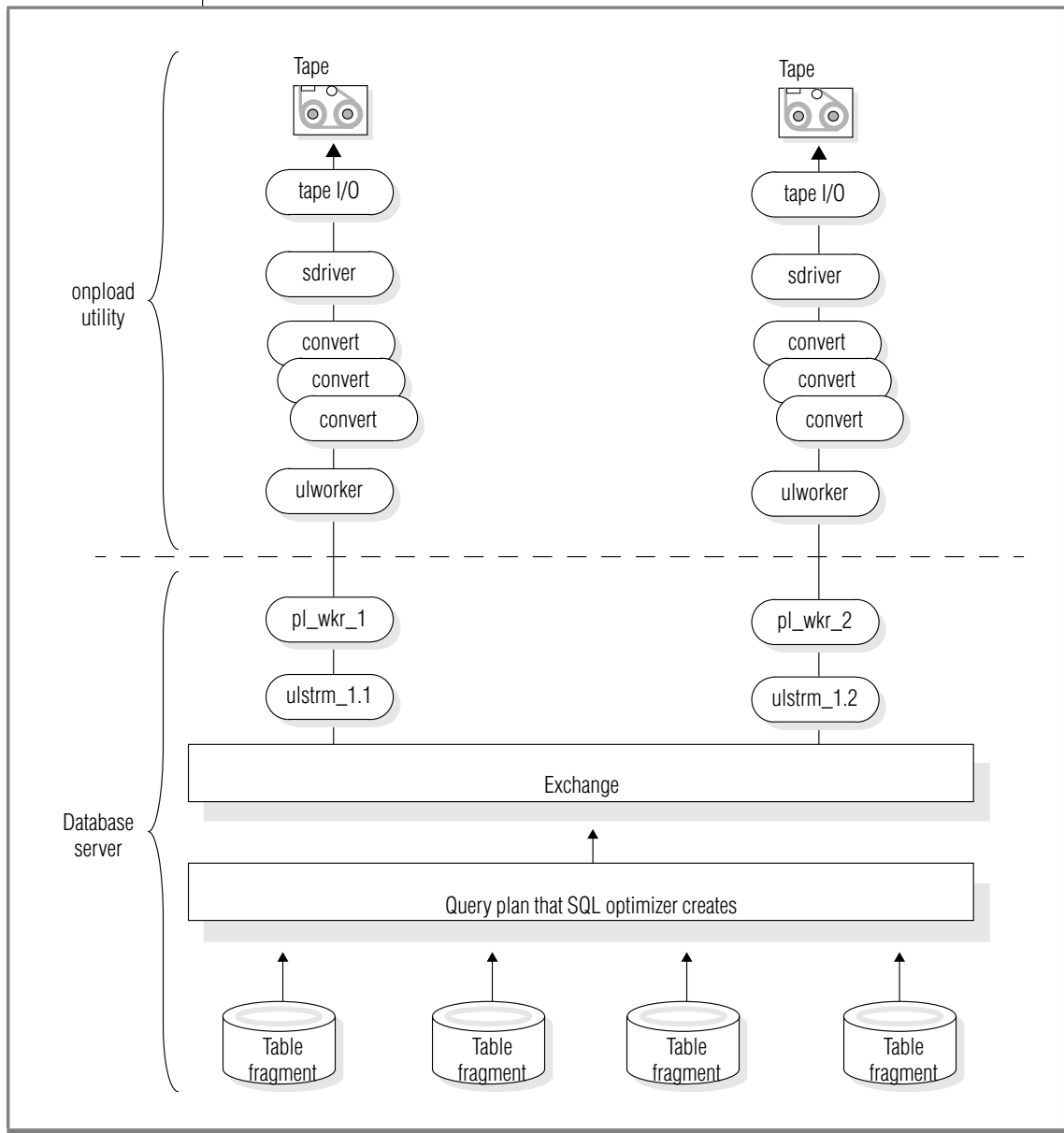
If your database is ANSI compliant, all access (both read and write) is denied until you perform a level-0 backup. Because data is not logged in express mode, the level-0 backup is necessary to allow for recovery in case of media failure. ♦

Figure 1-5
An Express-Mode Load



Unloads

Figure 1-6 on page 1-21 shows the **onpload** unload process. In the unload process, the behavior of **onpload** parallels the behavior described in “Threads That the onpload Utility Uses” on page 1-16 and “Threads That the Database Server Uses” on page 1-17, except that the threads are unloading the data instead of loading it.

Figure 1-6
The Unload Procedure

The **ulstrm** (unload-stream) thread packages data for output to the **onpload** client from the query plan. The SQL optimizer creates the query plan. The query plan behaves as if you were running a query from any other client, such as DB-Access. The exchange operator distributes the resulting data to the **ulworker** threads in a round-robin fashion, and **onpload** unloads the data onto tapes or files.

Parallelism with respect to the output device, the source table fragments, and the flow of the data is evident in [Figure 1-6 on page 1-21](#).

Getting Started

In This Chapter	2-3
Data-Load Example	2-3
Preparing to Use ipload	2-3
Creating a File of Data	2-4
Creating a Database	2-4
The ipload Utility	2-5
Starting ipload	2-5
Choosing a Project.	2-6
Checking the Defaults	2-6
Looking at the Defaults Window	2-6
Looking at the Machines Window	2-7
Load Job Windows	2-7
Load Job Select Window.	2-7
Load Job Window	2-10
Device-Array Windows	2-10
Device Array Selection Window	2-11
Device-Array Definition Window	2-12
Format Windows	2-13
Format Views Window	2-14
Record Formats Window	2-15
Format-Definition Window	2-17
Filter, Discard Records, and Logfile Text Boxes	2-19
Filter Text Box	2-19
Discard Records Text Box	2-19
Logfile Text Box	2-19

Map Windows	2-20
Map Views Window	2-20
Load Record Maps Window	2-23
Map-Definition Window.	2-23
Load Options Window.	2-28
Run Option	2-29
Active Job Window	2-30
Verifying the Data Transfer	2-31
Performing a Level-0 Backup	2-31
Generate Options	2-31
Starting the Example	2-32
Preparing the Unload-Job Window	2-32
Performing the Unload	2-37

In This Chapter

This chapter shows how the components of the High-Performance Loader (HPL) fit together. The chapter includes a step-by-step tutorial with two examples (a load and an unload) that use the **ipload** graphical user interface (GUI).

Data-Load Example

The illustrations in the first example use a database with only one table. The table contains three columns. The data to be loaded into the database is in a file that has only four records. In a real production environment, you would probably use the INSERT statement, the **dbimport** utility, or the LOAD statement for such a simple operation. However, by using an extremely simple example, the illustrations can show what happens at each step.

Preparing to Use ipload

The **ipload** utility is a part of UNIX database servers. Before you can use **ipload**, you must have *installed* the database server. If the database that you want to load or unload is on the computer where you plan to run **ipload**, you must also *start* the database server.

You can also use **ipload** to manage load and unload operations for a database server on a different computer. For example, you can use **ipload** on a UNIX computer to manage the **onpload** database on a Windows computer. For more information about **onpload** databases on Windows, see [Appendix I, “Running Load and Unload Jobs on Windows Computers.”](#)

Important: *The first time you run **ipload**, you must be user **informix**.*



Creating a File of Data

The illustrations in this chapter assume that the data to be loaded is in a file named **/work/mydata**. Create a file that contains the following data:

```
1|a|50
2|b|25
3|c|10
4|d|5
```

Creating a Database

The HPL loads data into an *existing* table in an *existing* database. The examples in this chapter load the information from the file **/work/mydata** into a three-column table named **tab1** in a database named **testdb**. You can use DB-Access to prepare the database and table, as follows:

```
CREATE DATABASE testdb;
CREATE TABLE tab1
(
    col1 INTEGER,
    col2 CHAR(1),
    col3 INTEGER
);
GRANT ALL ON tab1 TO PUBLIC;
GRANT CONNECT TO PUBLIC;
```

After you finish preparing the database for the examples, exit from DB-Access.

The *ipload* Utility

The HPL uses information from the **onpload** database to control loading and unloading of data. Theoretically, you could create the **onpload** database and use DB-Access or some other database tool to populate it. However, it is recommended that you always use **ipload** to manage the **onpload** database.

Starting *ipload*

To start **ipload**, type the following at the system prompt:

```
ipload
```

A decorative splash screen appears and stays on the display while **ipload** finishes loading. If you do not want to see the splash screen, use the **-n** flag, as follows:

```
ipload -n
```

The first time you start **ipload**, it automatically creates the **onpload** database. The **ipload** utility also puts certain default values into the database. [Appendix A, “onpload Database,”](#) describes the database tables.

When **ipload** starts, the HPL main window appears, as [Figure 2-1](#) shows.

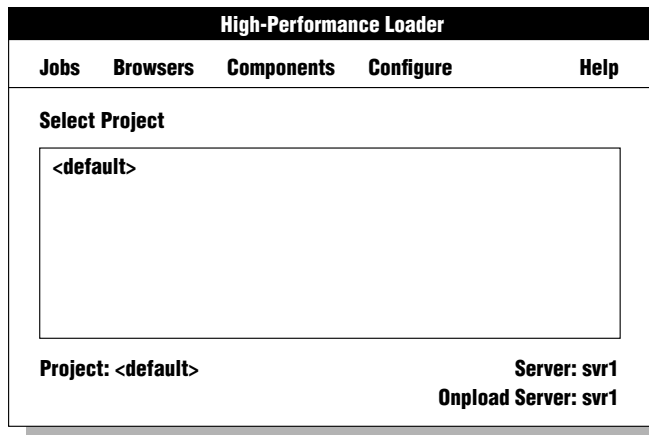


Figure 2-1
The HPL Main Window



Tip: To exit from *ipload*, choose *Exit* from the *Jobs* menu.

Choosing a Project

You use the HPL by preparing *load jobs* or *unload jobs* that import or export data. You can assign the load and unload jobs to various *projects* to organize the jobs into functional groups. Projects are described in [Chapter 4, “Defining Projects.”](#)

The **ipload** utility automatically creates a project named **<default>**. If you choose not to organize your work into projects, you can put all of the load and unload jobs in the default project.

For this example, you can use the default project. Click **<default>** on the HPL main window to choose the default project.

Checking the Defaults

The default values that **ipload** selects when it is first started specify machine type, character code set for character-type data, and other operating characteristics. In most cases, the only default that you might need to change is the machine type. [Chapter 5, “Configuring the High-Performance Loader,”](#) describes the **ipload** defaults.

Looking at the Defaults Window

Choose **Configure→Defaults** to see the current default values. After you check the defaults, you can click **Cancel** to exit from the Defaults window. If you need to change one of the default values, see [“Modifying the onpload Defaults” on page 5-5.](#)

Looking at the Machines Window

Choose **Configure**→**Machines** to see the current default values. Make sure that the machine type displayed in the Machines window matches the current machine type. Click the **Machine Type** down arrow to select a machine type.

If you need to change one of the values, see [“Modifying the Machine Description” on page 5-11](#). Click **Cancel** to exit from the Machines window.

Load Job Windows

A *load job* is a collection of the specific pieces of information that you require to move data from a data file into a database. The Load Job window, illustrated in [Figure 2-3 on page 2-9](#), shows a flowchart that includes all of the components of a load job. After you become familiar with **ipload**, you can create or modify the individual components of a load or unload job with direct menu choices from the HPL main window.

Load Job Select Window

The data-load example in this chapter takes records from **/work/mydata** and loads them into **tab1** of the **testdb** database. To perform the load, you need to create a load job.

To create a load job

1. Choose **Jobs→Load** from the HPL window.

The Load Job Select window appears, as [Figure 2-2](#) shows.

Figure 2-2
The Load Job Select Window

Load Job Select

Delete **Notes** **Connect**

Selection Type

☐ Open ☒ Create

Job Name:

Command Line:

☐ Write/read to/from tape until end of device.

Job Information

Job	Type	Status	Server	Map	Datasource

Notes

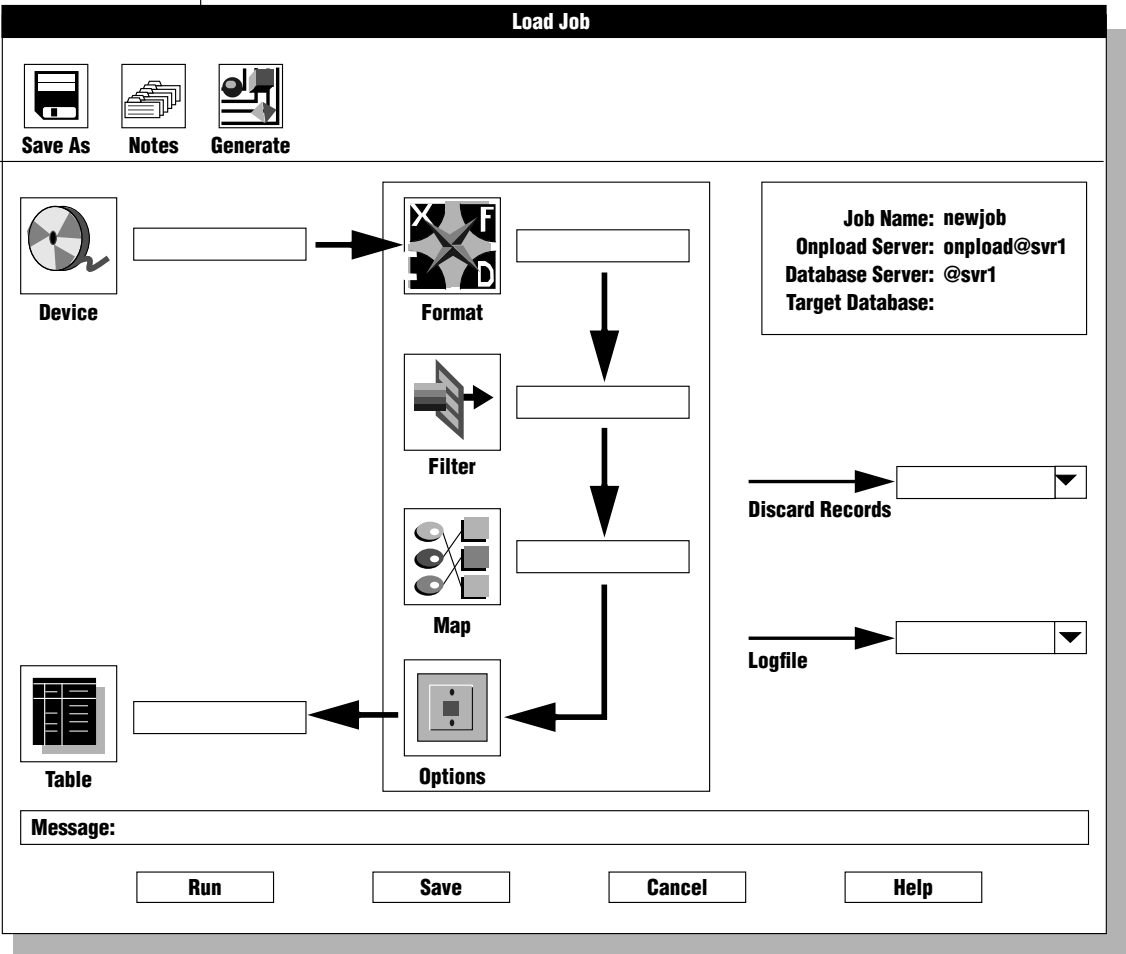
Message: Enter a job name to create

OK **Cancel** **Help**

2. Click **Create** in the **Selection Type** group.
3. Choose a name for the load job and type it in the **Job Name** text box.
This example uses **newjob**.

- 4. Click OK.
The Load Job window appears, as [Figure 2-3](#) shows.

Figure 2-3
The Load Job Window



Load Job Window

The thick arrows on the Load Job window indicate the steps that you take as you build a load job. The icons indicate the task at each step. The thin arrows indicate filenames for error recording. To create a load job, you need to complete the following tasks:

Task	Click
Specify the source of the data	Device
Describe the data	Format
Tell ipload which data you want to discard (optional)	Filter
Specify association between input fields and load-table columns	Map
Specify options for the load job	Options
Specify the database table to load	Table
Record rejected data records (optional)	Discard Records
Record information about the job (optional)	Logfile

Device-Array Windows

A *device array* is a collection of files, tape devices, and pipes that **onpload** uses for input and output. Pipes are only supported on UNIX.

The Device Array Selection and device-array definition windows let you create a device array and specify the location of the input data. In this example, the input data is the **/work/mydata** file that you created in [“Creating a File of Data” on page 2-4](#).

Device Array Selection Window




To create a device array

1. Click **Device** in the Load Job window.

The Device Array Selection window appears, as [Figure 2-4](#) shows.

Figure 2-4
The Device Array Selection Window

Device Array Selection

 **Copy**  **Delete**  **Print**

Selection Type

☐ Open ☒ Create **Device Array:**

Current Arrays

Notes

Message:

2. Click **Create** in the **Selection Type** group.

3. Select a name for the device array and type it in the **Device Array** text box.

This example uses **an_array**.

4. Click **OK**.

The device-array definition window appears, as [Figure 2-5](#) shows.

Device-Array Definition Window

The title bar of the device-array definition window shows the array name that you typed in the **Device Array** text box.

Figure 2-5

The Device-Array Definition Window with One Array Item

The screenshot shows a window titled "an_array". At the top left are two icons: a printer labeled "Print" and a notepad labeled "Notes". Below these are two sections. The "Array Item Type" section has three radio buttons: "Tape", "File", and "Pipe". The "Tape Parameters" section has two text boxes for "Block Size:" and "Tape Size:", followed by two radio buttons for "MB" and "GB". Below these is the "Array Items" section, which contains a list box with the text "FILE /work/mydata" and a "Perform" button. To the right of the list box are three radio buttons: "Add" (selected), "Edit", and "Delete". At the bottom of the window is a message box that says "Message: Select item to edit, or add new device item". Below the message box are three buttons: "OK", "Cancel", and "Help".

an_array

Print **Notes**

Array Item Type

☐ **Tape** ☐ **File** ☐ **Pipe**

File Name:

Tape Parameters

Block Size: **Tape Size:** ☐ **MB** ☐ **GB**

Array Items

FILE /work/mydata

Perform

☒ **Add**
☐ **Edit**
☐ **Delete**

Message: Select item to edit, or add new device item

OK **Cancel** **Help**

To add a device to the array

1. Click **Add** in the **Perform** group (lower right).
2. Click **File** in the **Array Item Type** group (upper left).
3. Type the full pathname of a device in the **File Name** text box.
In this example, the device is **/work/mydata**.
4. Click **Perform** to add the **/work/mydata** file to the device array.
The **ipload** utility lists each item of the device array in the **Array Items** list box. [Figure 2-5](#) shows the window after you add **/work/mydata** to the array.
5. Click **OK**.
The display returns to the Load Job window. The **Device** list box now displays the device array name that you chose.

Format Windows

The format specifies the organization of the input data. In this example, the input data is in the file **/work/mydata**, which you created in [“Creating a File of Data” on page 2-4](#). Each record in the file has three fields.

Format Views Window

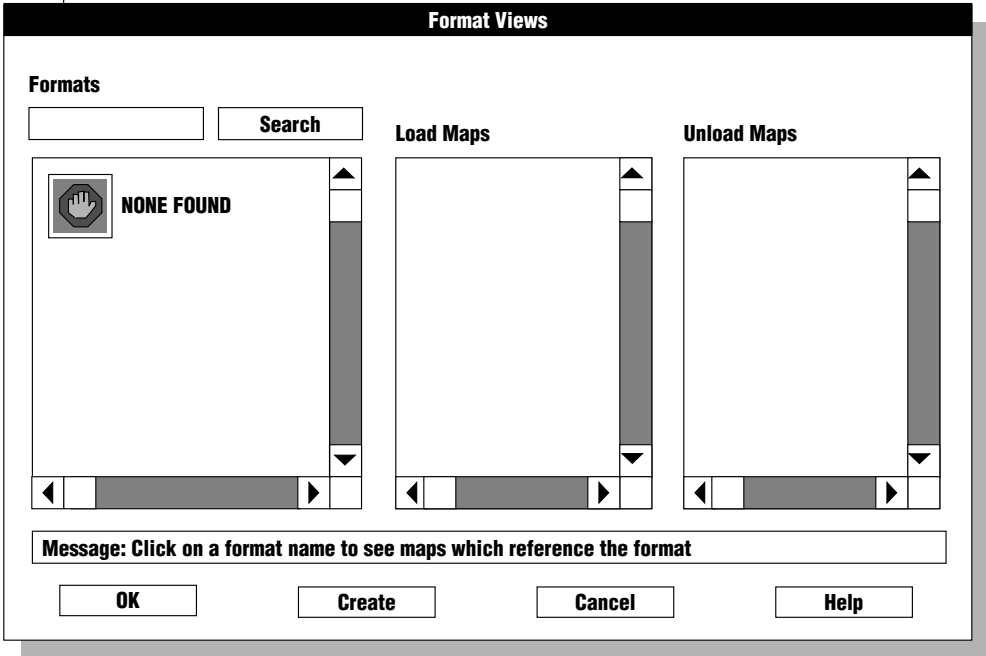
The Format Views window displays existing formats, so that you can choose the format to use in the load job.

To open the Format Views window

1. Click the **Format** button in the Load Job window.

The Format Views window appears, as [Figure 2-6](#) shows. When you first start **ipload**, no formats are defined, as the NONE FOUND icon illustrates.

Figure 2-6
The Format Views Window





2. Click **Create** to open the Record Formats window.


Record Formats Window


Figure 2-7 shows the Record Formats window. The Record Formats window lets you create a new format or open an existing format.

Record Formats

Copy

Delete

Print

Search

Mode

☐ Open

☒ Create

Type

☒ Fixed

☐ Delimited

☐ COBOL

☐ COBOL (byte)

Formats

Create Format:

Format

Type

Notes

Message:

OK

Cancel

Help

Figure 2-7
The Record Formats Window

To create a new format

1. Click **Create** in the **Mode** group.
2. Click **Delimited** in the **Type** group.

The input data file, **/work/mydata**, is in delimited format. Other formats are described in [Chapter 7, “Defining Formats.”](#)

3. Choose a name for the format and type it in the **Create Format** text box.

This example uses the name **a_format**.

4. Click **OK**.

The format-definition window appears. [Figure 2-8](#) shows a partially completed format-definition window. The title bar of the format-definition window shows the name that you chose for the new format.

Figure 2-8
The Format-Definition Window

Delimited Format—a_format

Save As Print Notes Options Browse

Field Name **Field Type**

input2 Chars

input1 Chars

Perform

Operation

☒ Add
☐ Insert
☐ Edit
☐ Delete

Message: Please enter field specification

OK Cancel Help

Format-Definition Window

In the format-definition window, you must make an entry for each field of the data records in the input file. The input file for this example (**/work/mydata**) has three fields of data in each record, so you must enter format information for three pieces of data.

To enter a format definition

1. Click **Add** in the **Operation** group.
2. In the **Field Name** text box, type a descriptive name for the first field of the data record.

You can choose any descriptive name. This example uses **input1**, **input2**, and **input3** for the three fields of **/work/mydata**.
3. In the **Field Type** text box, type the data type or click the down arrow for a list of selections.

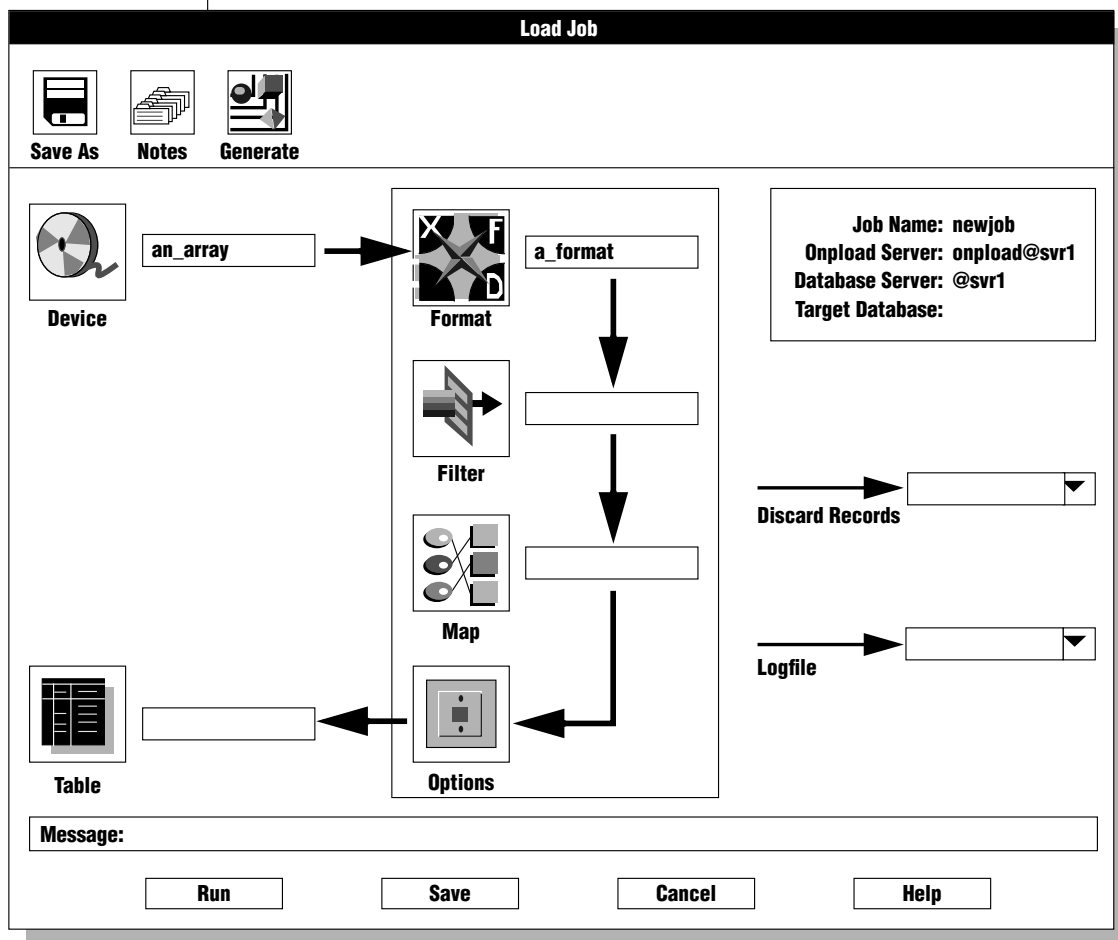
Because the data in **/work/mydata** is simple ASCII data, the type is **Chars**. Other data types are discussed in [Chapter 7, “Defining Formats.”](#)
4. Click **Perform**.

[Figure 2-8](#) shows the partially completed format-definition window. The entry for the first item is complete. The **Field Name** and **Field Type** for the second item are present and ready for you to click **Perform**.
5. Repeat steps 2 through 4 for each of the three input fields.
6. Click **OK** after you complete all of the input fields.

The display returns to the Format Views window, which displays the new format in the **Formats** list box.
7. Click **Cancel** to return to the Load Job window.

The Load Job window now displays the name of the device and the name of the format, as [Figure 2-9](#) shows.

Figure 2-9
Partially Completed Load Job Window



Filter, Discard Records, and Logfile Text Boxes

The Load Job window now has entries for a device and format. The next incomplete items in the Load Job window ([Figure 2-9](#)) are the **Filter** text box, the **Discard Records** text box, and the **Logfile** text box.

Filter Text Box

Use a filter to choose the records from the data file that should be inserted into the table. In this example, all of the records from the `/work/mydata` data file will be inserted into the database table. Therefore, you do not need to create a filter. For this example, you can leave the **Filter** text box blank.

[Chapter 10, “Defining Filters,”](#) describes how to create and use a filter.

Discard Records Text Box

The **Discard Records** text box specifies a file that keeps information about records that were rejected because of incorrect format or invalid data. For this example, you can leave the **Discard Records** text box blank.

[“Reviewing Records That the Conversion Rejected” on page 14-6](#) describes how to view rejected records.

Logfile Text Box

The **Logfile** text box specifies a file where a record of the load or unload job is kept. For this example, you can leave the **Logfile** text box blank.

[“Viewing the Status of a Load Job or Unload Job” on page 14-9](#) describes how to view the log file.

Map Windows

The Map windows let you create a map that specifies which data field from the input file (**/work/mydata**) is entered into which columns in the database table.

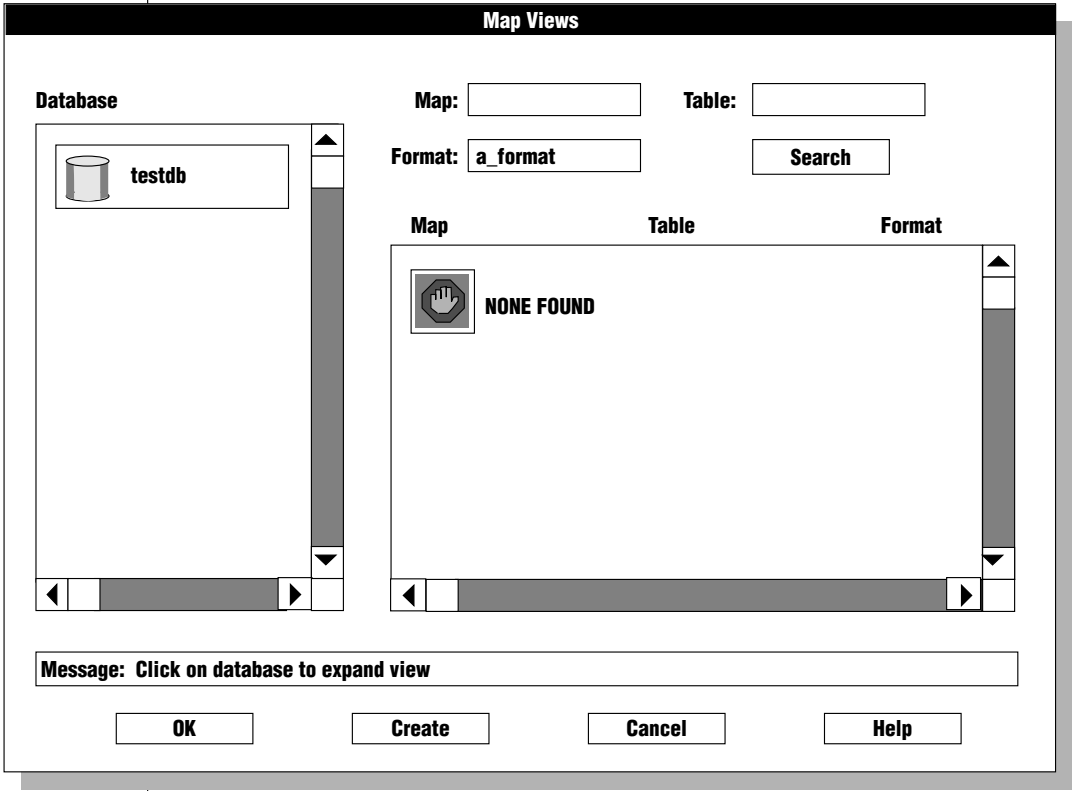
Map Views Window

The Map Views window lets you create a new map or edit an existing map. You need to create a map that shows how the input data that the record format **a_format** describes should be loaded into the table **tab1**.

To create a new map

- 1. Click the **Map** button in the Load Job window.
The Map Views window appears, as [Figure 2-10](#) shows.

Figure 2-10
The Map Views Window



The NONE FOUND icon in the Map column is appropriate. At this point, the Map Views window does not contain any information because you have not yet specified any relationships.

2. Click **Create**.

The Load Record Maps window appears, as [Figure 2-11](#) shows.

Figure 2-11

A Partially Completed Load Record Maps Window with an Open Selection List

The screenshot shows the 'Load Record Maps' window with a 'Select From List' dialog box open. The 'Load Record Maps' window has a title bar and a toolbar with icons for Copy, Delete, Print, and Search. Below the toolbar, there are sections for 'Selection Type' (with 'Open' and 'Create' radio buttons, where 'Create' is selected), 'Map Data' (with fields for Map Name: 'a_map', Database: 'testdb', Table: (empty), and Format: 'a_format'), and 'Notes' (a text area). At the bottom of the window is a 'Message: Enter a map name to create' field with 'OK', 'Cancel', and 'Help' buttons. The 'Select From List' dialog box has a title bar, an 'Items' list containing 'tab1', a 'Selection' text field, and 'OK' and 'Cancel' buttons.

Load Record Maps Window

The Load Record Maps window specifies the device array that holds the input data, the format that describes the input data, and the database and table where the input data will be stored.

To complete the Load Record Maps window

1. Click **Create** in the **Selection Type** group.
2. Select a name for the map and type it in the **Map Name** text box.
This example uses **a_map**.
3. Type the name of your database (**testdb**) in the **Database** text box, or click the down arrow to select a database from the selection list.
4. Click the down arrow beside the **Table** text box to see a list of tables in the selected database.

[Figure 2-11](#) shows the Load Record Maps window and the selection list at this point.

5. Select a table from the list and click **OK**.

Because you already filled in the **Format** text box on the Load Jobs window, the **Format** text box is already complete.

6. Click **OK** to open the map-definition window.

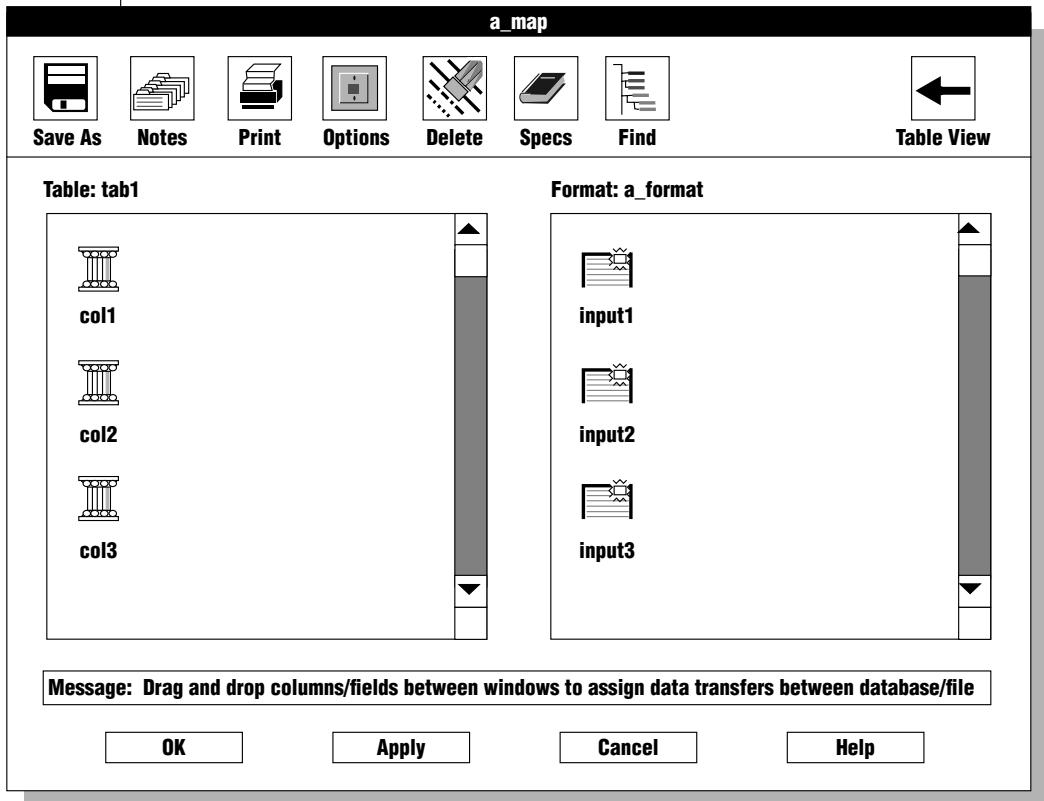
Map-Definition Window

The map-definition window lets you associate an input item with a table column. This example stores the data from **/work/mydata** as follows, using the field names assigned on [page 2-15](#).

Data from Input Field	Goes into Table Column
input1	col3
input2	col2
input3	col1

Figure 2-12 shows the map-definition window. The title bar of this window shows the map name that you chose.

Figure 2-12
The Map-Definition Window



To associate each input item with a column of the database table

1. Click the **col1** icon and *hold the mouse button down*.
A box appears around the icon and its name.
2. Drag the boxed icon to the **input3** icon in the right-hand pane.

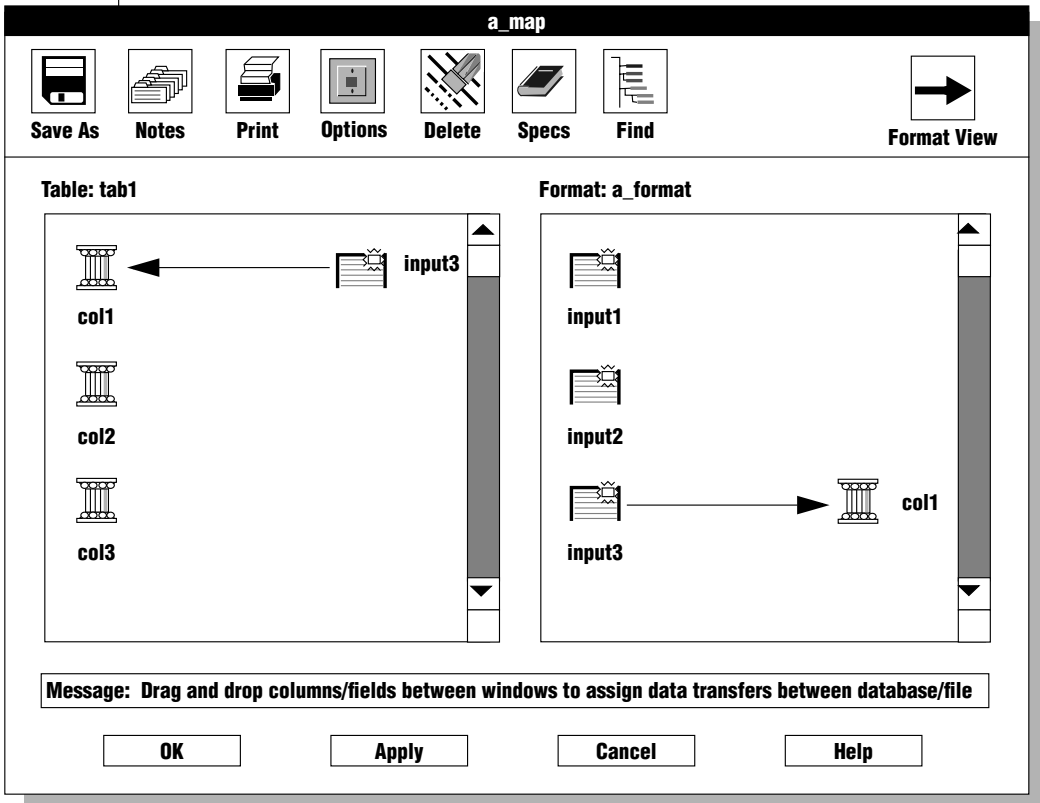
3. Release the mouse button.

The associated items appear in the second column of each pane.

Figure 2-13 shows the map-definition window with this step completed.

Figure 2-13

The Map-Definition Window with One Association Completed



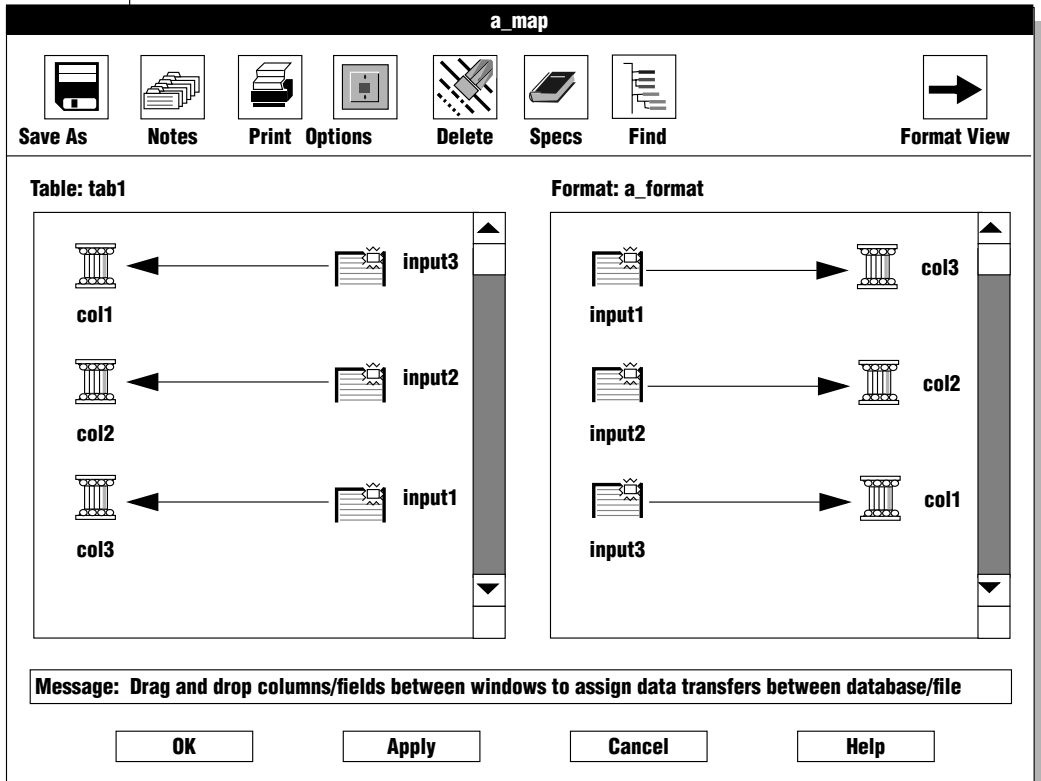
4. Connect col2 to input2.

5. Connect col3 to input1.

Figure 2-14 shows the window with all three connections completed.

Figure 2-14

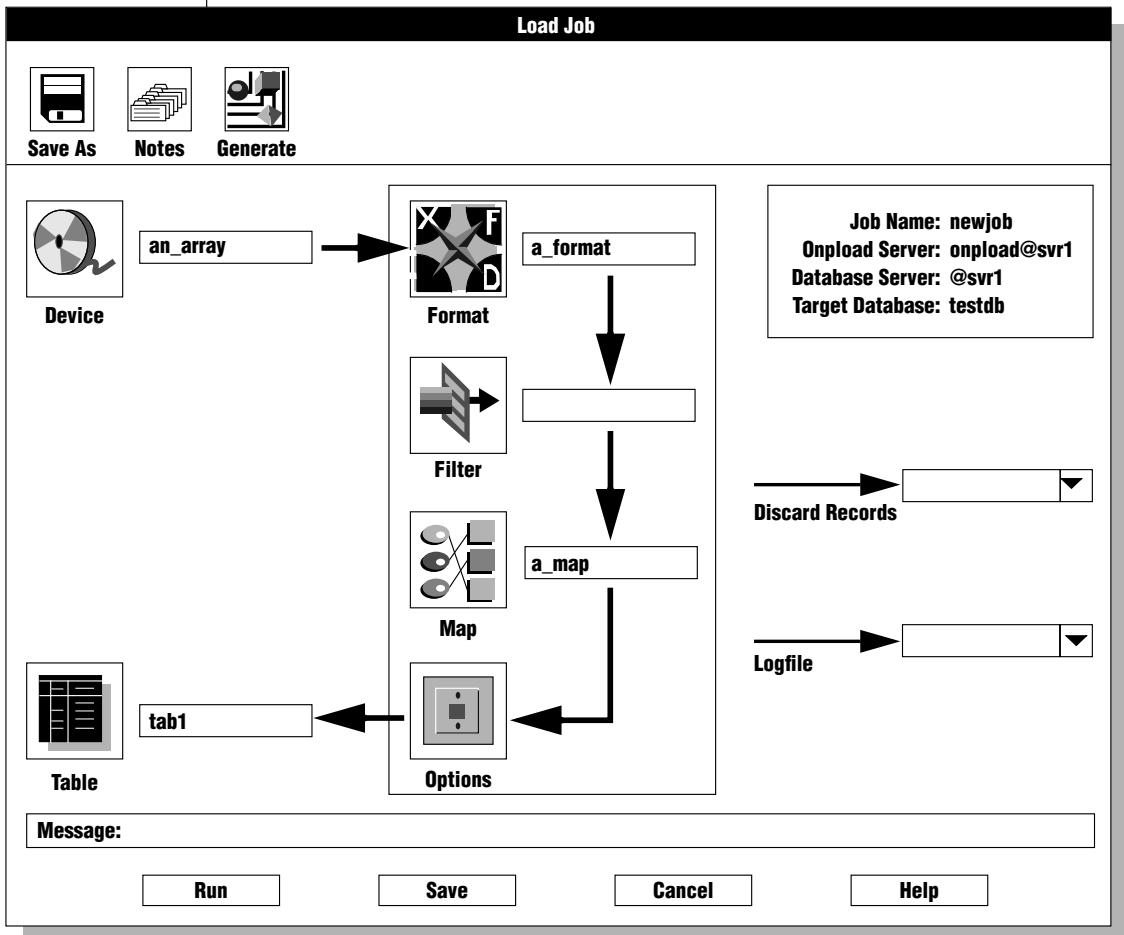
The Map-Definition Window with All Associations Completed



6. Click **OK** to return to the Map Views window.
7. Click **Cancel** to return to the Load Job window.

The Load Job window now has entries in all of the required areas, as [Figure 2-15](#) shows. The **ipload** utility was able to fill in the table name and the target database name (upper right area) because you specified the database and table as you built the map.

Figure 2-15
The Load Job Window with All Required Component Boxes Completed



You have finished all of the required parts of the Load Job window, but you might want to modify the options, as discussed in the next section.

Load Options Window

The HPL has three modes of operation: express, deluxe with replication, and deluxe without replication. The express mode is optimized for speed and the deluxe mode provides the full functionality of SQL inserts as data is loaded. For a detailed comparison of these modes, see [Chapter 15, “Managing the High-Performance Loader.”](#) This example uses the express mode.

To set the load-job options

1. Click **Options** in the Load Job window.

The Load Options window appears, as [Figure 2-16](#) shows.

Load Options

Load Mode: Express ▼

Generate Violations Records: Deluxe with Replication

Tapes: 0

Number Records: 0

Start Record: 0

Max Errors: 0

Commit Interval: 0

Message: Table is locked during Express mode load

OK **Cancel** **Help**

Figure 2-16
The Load
Options
Window

2. Select **Express** from the **Load Mode** list box.
3. Select **Yes** from the **Generate Violations Records** list box.
4. Make sure all of the other entries are 0.
5. Click **OK** to return to the Load Job window.

Run Option

You are now ready to perform the load.

To finish this example

1. Click **Save** to save the load job.

After you save the load job, the message line displays the following message:

```
Saved job successfully
```

You can now execute the job, or you can return at a later time and execute the job.

2. Click **Run** to execute the load job.

The Active Job window appears, as [Figure 2-17 on page 2-30](#) shows.

Tip: The *ipload* utility generates an *onpload* command and then executes the command to run your job. To see the command that *ipload* generates, look at the **Command Line** text box on the Load Job Select window. For more information, see [“Using the Command-Line Information” on page 12-13](#).



Active Job Window

The Active Job window reports the progress of your job. [Figure 2-17](#) shows the Active Job window after the load is complete.

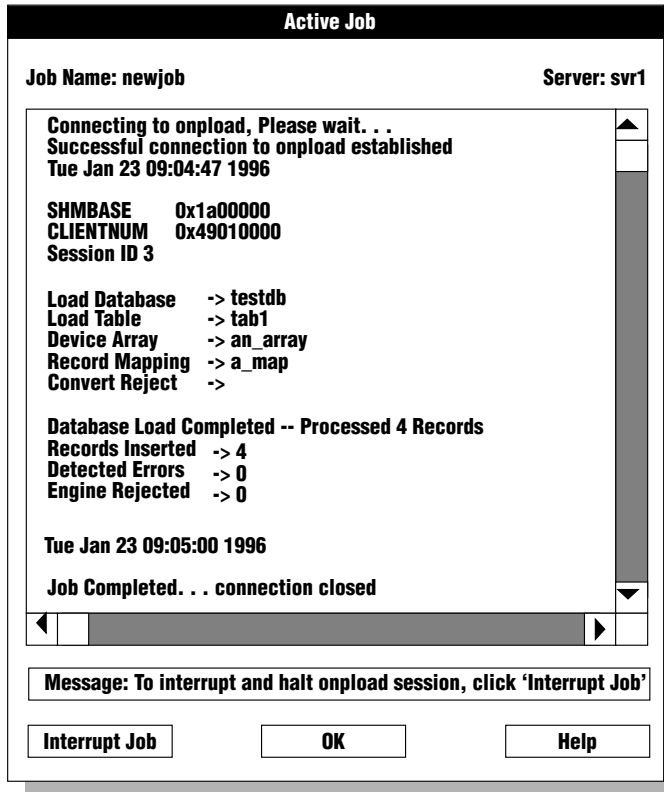


Figure 2-17
The Active Job Window

When the Active Job window reports that the load job is complete, click **OK** to return to the Load Job Select window.

Verifying the Data Transfer

If you want, you can use DB-Access to verify that the data from **/work/mydata** was transferred into your database.

Performing a Level-0 Backup

The **onpload** utility does not log the data that it writes to a table during an express-mode load. For safety, **onpload** flags the dbspaces that are associated with the table as read-only. To allow for data recovery in case of disk corruption, you must make a level-0 backup. A level-0 backup on the dbspaces affected by the express-mode load saves the data and unsets the read-only flags.

If you do not care about data recovery, you can make a level-0 backup using **/dev/null** as the backup device. This action unsets the read-only flag without backing up data to any real device.

Generate Options

The **ipload** utility has Generate options that you can use to automatically create a format, map, query, and device. After the components are generated, you can modify the components to meet your needs. The Generate options are described in [Chapter 13, “Generate Options.”](#)

This example uses the **Generate** button in the Unload Job window to create the components that are required for an unload job. After you create the components, you can use the **Run** option to execute the unload job.

Starting the Example

If you completed the first example in this chapter, your database server and **ipload** are ready for you to use. If you did not complete the example, you need to complete the following tasks as the first example describes:

- Start your database server ([page 2-3](#))
- Start **ipload** ([page 2-5](#))
- Check the defaults ([page 2-6](#))

Preparing the Unload-Job Window

An *unload job* is a collection of the specific pieces of information that are required to move data from a database into a data file. The Unload Job window ([Figure 2-3 on page 2-9](#)) shows a flowchart that includes all of the components of an unload job. You can use the **Generate** option to create the components of the unload job and to complete the items on the Unload Job window.

The generate example uses the **Generate** option to unload the contents of the **items** table of the **stores_demo** database into a file named **/work/items_out**. For instructions on how to create the **stores_demo** database and other demonstration databases, see the *IBM Informix DB-Access User's Guide*.

To generate the unload job


1. Choose **Jobs→Unload** from the HPL window.
The Unload Job Select window appears, as [Figure 2-18](#) shows.
2. Click **Create** in the **Selection Type** group.


3. Choose a name for the unload job and type it in the **Job Name** text box.


This example uses the name **unld**.

Figure 2-18
The Unload Job Select Window

Unload Job Select


Delete


Notes


Connect

Selection Type

☐ Open ☒ Create

Job Name:

Command Line:

☐ Write/read to/from tape until end of device.

Job Information

Job	Type	Status	Server	Map	Datasource

Notes

Message: Enter a job name to create

OK

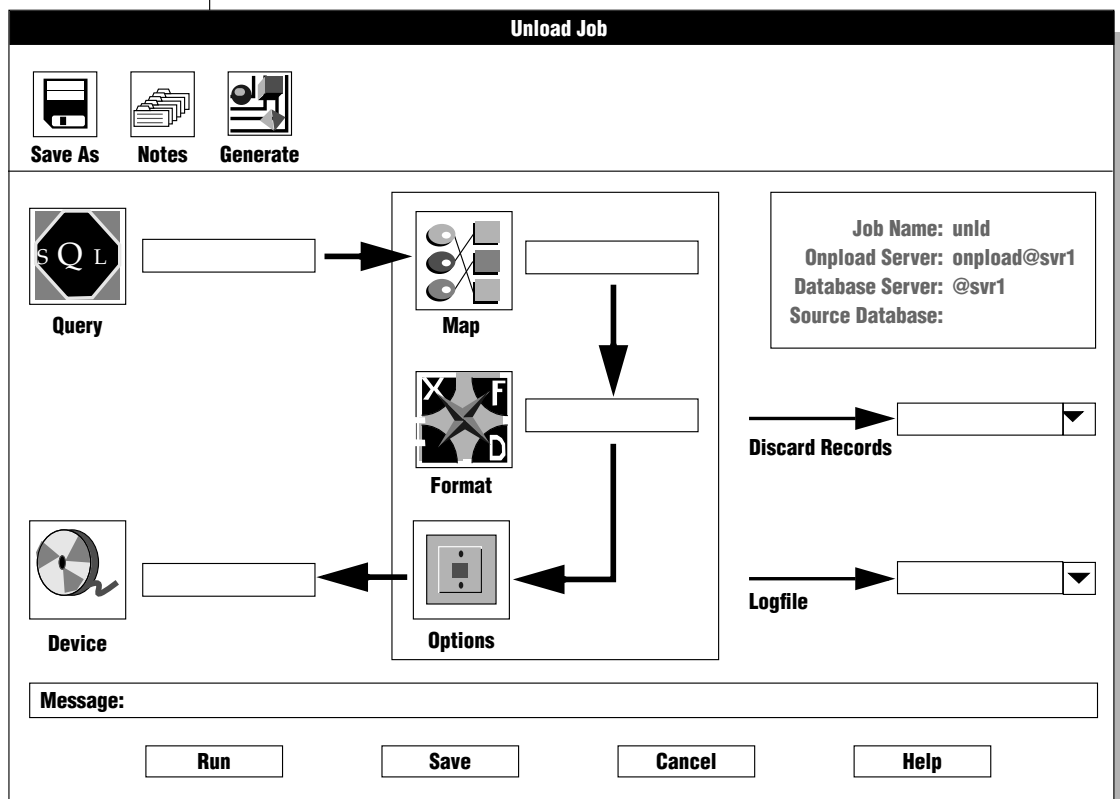
Cancel

Help

4. Click **OK**.

The Unload Job window appears, as [Figure 2-19](#) shows. The information box in the upper right part of the display shows the name of the unload job, the name of the database server where the **onpload** database is stored, and the name of the database server where **ipload** is running.

Figure 2-19
The Unload Job Window



5. Click the **Generate** button.

The Autogenerate Unload Components window appears. [Figure 2-20](#) on page 2-35 shows the completed window.

6. Click **Table**.

You can unload an entire database table or only selected records from the table. **Table** indicates that you want to unload the entire table.

Query indicates that you want to unload selected records.

7. Type `stores_demo` in the **Database** text box.

For this step and steps 8 and 10, you can click the down arrow to the right of the text box and select the entry from a selection list.

[Figure 2-11 on page 2-22](#) shows an example of a selection list.

8. Type `items` in the **Table** text box.

Autogenerate Unload Components

Unload from

☒ **Table** **Database:** `stores_demo` ▼

☐ **Query** **Table:** `items` ▼

Query: ▼

Unload to

☒ **Device Array** `/work/items_out` ▼

☐ **File**

Message: Enter database to unload

OK **Cancel** **Help**

Figure 2-20
The Autogenerate
Unload
Components
Window

9. Click **File**.

File indicates that you want to type the name of a file. If you choose **Device Array**, you must type the name of an already existing device array.

10. Type the full pathname of the file that will store the unloaded data. This file can be in any directory to which you have write access.

11. Click **OK**.

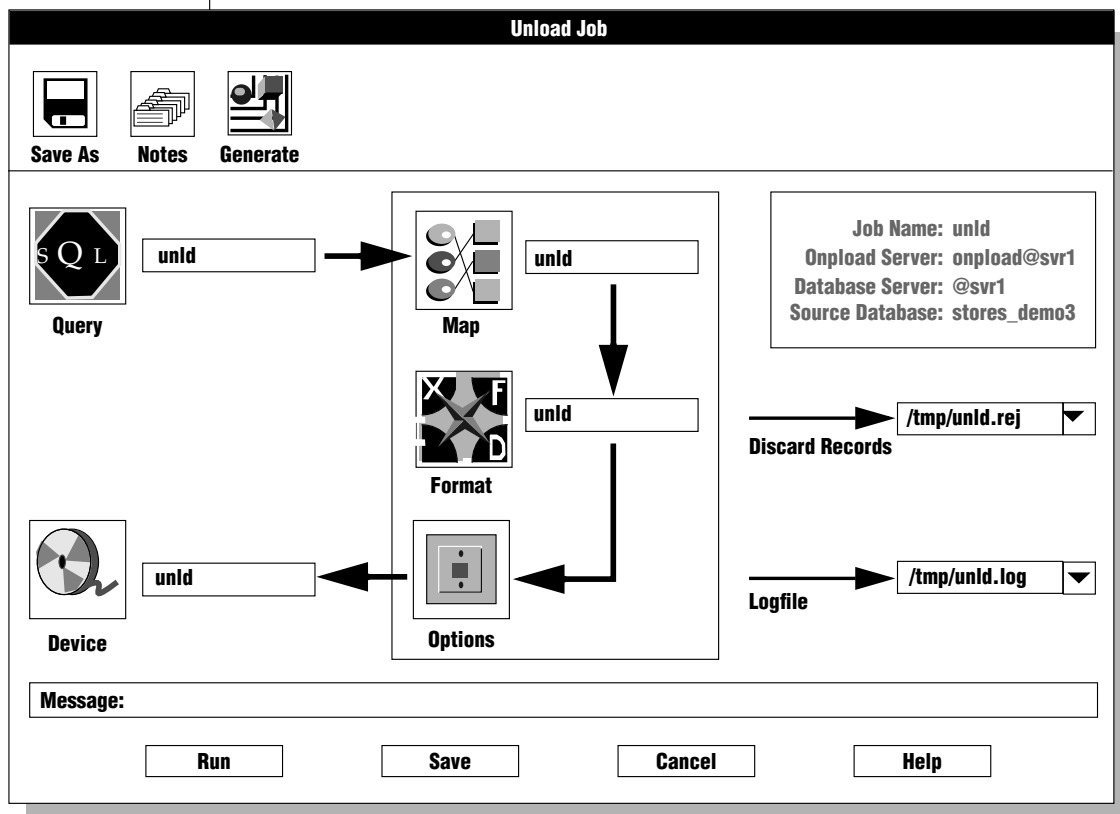


The **Generate** option creates the Query, Format, and Map components for the unload job and fills in the Unload Job window. These components are all named **unld**. The **Generate** option also creates a device array named **unld** and puts the file that you specified (**/work/items_out**) into that array.

Tip: After you finish this exercise, you can choose **Components→Devices** from the HPL window and examine the **unld** device array.

Figure 2-21 shows the Unload Job window as completed by the **Generate** option.

Figure 2-21
The Unload Job Window



In addition to completing the main flow of the Unload Job window, the **Generate** option also fills in the Source Database information in the upper right-hand corner and creates pathnames for the Discard Records file and the Logfile. [Chapter 14, “Browsing,”](#) describes the rejected records file and the log file.

Performing the Unload

You are now ready to perform the unload.



Tip: At this point, you might want to preview the records that the query will select. See [“Previewing Data-File Records” on page 14-3](#).

To finish this example

1. Click **Save** to save the unload job.

After you save the unload job, the Message line displays the following message:

```
Saved job successfully
```

You can now execute the job, or you can return at a later time and execute the job.

2. Click **Run** to execute the unload job.

The Active Job window appears. This window reports the progress of your unload job. The Active Job window for an unload job is similar to the Active Job window for a load job, which [Figure 2-17 on page 2-30](#) shows.

3. When the Active Job window reports that the unload job is finished, click OK to return to the Unload Job window.
4. Click **Cancel** to return to the HPL main window.
5. You can create another job or choose **Jobs→Exit** to exit **ipload**.

Using the High-Performance Loader Windows

In This Chapter	3-3
Starting ipload	3-3
Using the ipload GUI	3-3
HPL Main Window	3-4
Initial Options on the HPL Main Window	3-4
Options of the HPL Main Window	3-5
Component-Selection Windows	3-6
Toolbar Buttons	3-7
Selection Type	3-8
Component-Name Text Box	3-8
Component List Box	3-8
Notes Area	3-8
Message Line	3-8
Buttons	3-9
Component-Definition Windows	3-9
Toolbar Buttons	3-11
Item-Selection Group	3-11
Item-Name Text Box	3-11
Special-Parameters Group	3-11
Item List Box	3-12
Perform Group	3-12
Message Line	3-12
Buttons	3-12
Load Job and Unload Job Windows	3-13
Views Windows	3-14
Accessing Views Windows	3-14
Available Options in a Views Window	3-15
Selection-List Windows	3-18
Message Windows	3-19

Using the HPL Buttons	3-19
Toolbar Buttons	3-20
Browse Button	3-21
Copy Button	3-21
Delete Button	3-23
Notes Button	3-24
Print Button.	3-27
Icon Buttons	3-28
Buttons.	3-30
Using the Online Help.	3-31
Using UNIX Keyboard Commands to Move the Cursor	3-32

In This Chapter

This chapter describes the **ipload** graphical user interface (GUI) and discusses the **ipload** windows, buttons, online help, and keyboard commands.

Starting ipload

To start **ipload**, type the following word at the system prompt:

```
ipload
```

A decorative splash screen appears and stays on the display while **ipload** finishes loading. If you do not want to see the splash screen, use the **-n** flag, as follows:

```
ipload -n
```

Using the ipload GUI

The **ipload** utility has the following types of displays:

- HPL main window
- Component-selection windows
- Component-definition windows
- Load Job and Unload Job windows
- Views windows
- Selection-list windows
- Message windows

HPL Main Window

When you start **ipload**, the HPL main window appears, as [Figure 3-1](#) shows. The HPL main window is the focus of the user interface. You return to the main window after each task and choose a new option.

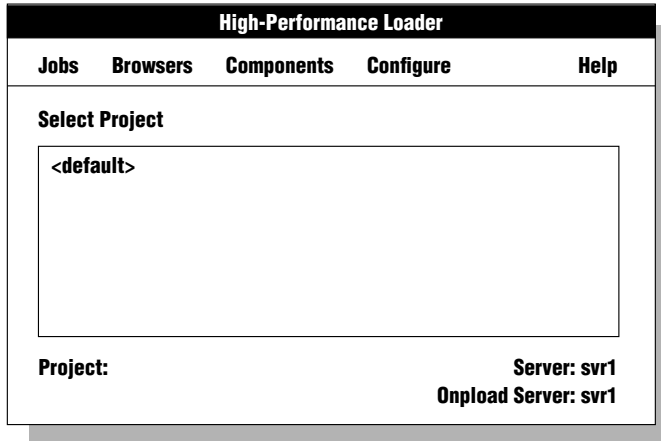


Figure 3-1
The HPL Main Window

Initial Options on the HPL Main Window

When you first enter **ipload**, you can:

- Select the default project in the Select Project list.
- Choose **Configure**→**Project** to create a new project.
For information on how to create a project, see [“Creating a New Project” on page 4-7](#).
- Choose **Configure**→**Server** to select a database server and an **onpload** database server.
For information about how to choose a database server, see [“Selecting a Database Server” on page 5-3](#).
- Choose **Help** to look at the online help.
- Choose **Jobs**→**Exit** to exit from **ipload**.

Options of the HPL Main Window

After you select a project, you can choose options from any of the menus on the HPL main window. The following list describes each of the menu options.

Main Menu Option	Submenu Option	Purpose	Page
Jobs	Load	Create a load job and use the Load Job window to load data into a database	12-3
	Unload	Create an unload job and use the Unload Job window to unload data from a database to a file	11-5
	Exit	Exit from the user interface	2-5
Browsers	Record	Review records in a specified format, search the list of available formats, or edit a format	14-6
	Violations	View records that passed the filter and conversion but were rejected by the database	14-7
	Logfile	View load status and see where any errors occurred	14-9
Components	Formats	Create or modify data-file formats	7-3
	Maps	Create or modify maps that show the relationship between data-file fields and database columns	9-3
	Query	Build, modify, or retrieve SQL-based queries	8-4
	Filter	Create or modify filters that determine source data-file records for conversion and load	10-3
	Devices	Specify a set of files, tapes, or pipes (only on UNIX) that will be read simultaneously for loading or unloading the database	6-3
	Generate Job	Automatically generate the components for load and unload jobs	13-3

(1 of 2)

Main Menu Option	Submenu Option	Purpose	Page
Configure	Server	Select the database servers that hold the onpload database and the target database	5-3
	Project	Create a project under which formats, filters, queries, maps, and load and unload jobs are stored	4-3
	Defaults	Specify the default character sets for the data file and databases	5-5
	Machines	Specify the machine parameters that are used to convert binary data	5-11
Help	Glossary	View definitions of terms that pertain to the HPL	3-31
	Contents	View the main contents page that directs you to discussions of various HPL topics	3-31

(2 of 2)

Component-Selection Windows

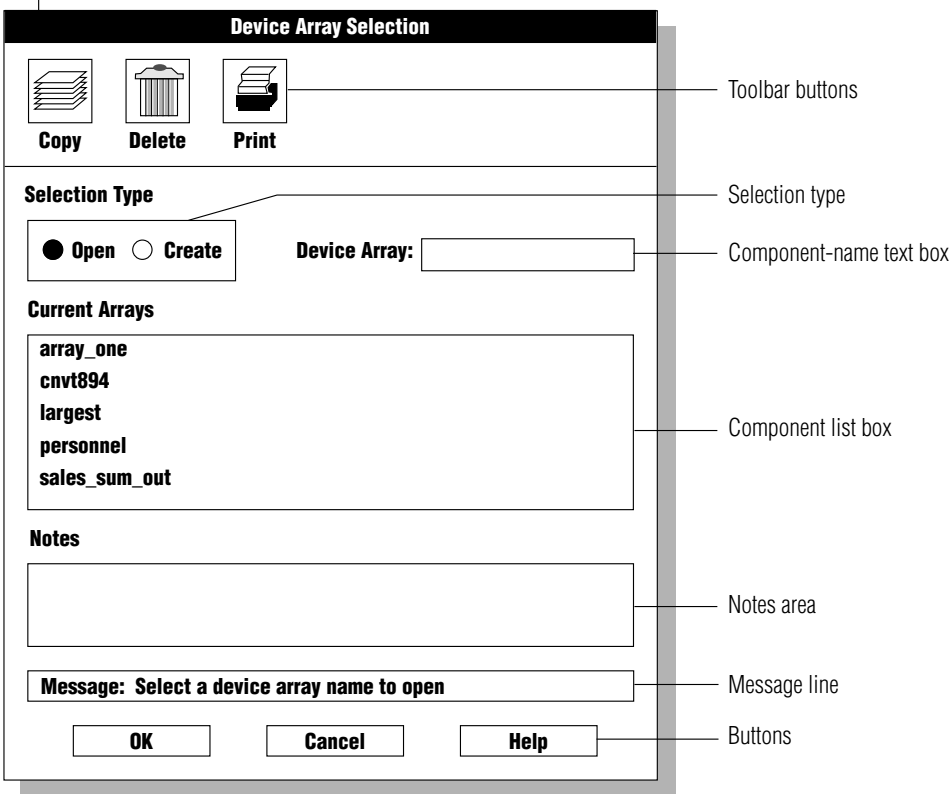
The windows for creating or modifying components often (but not always) come in pairs. The first window, the component-selection window, lets you create a new component or select an existing component to modify. This window also lets you view notes and copy, delete, or print information about a component. The second window, the component-definition window, lets you make the actual changes.

The details of a selection window vary depending on the operation that you are performing. However, the component-selection windows have the following standard features:

- Toolbar buttons
- Selection type
- Component-name text box
- Component list box
- Notes area
- Message line
- Buttons

Figure 3-2 shows the Device Array Selection window to illustrate the standard features of component-selection windows.

Figure 3-2
The Device Array Selection Window



Toolbar Buttons

The buttons across the top of the display represent actions that you can take after you select a component from the component list. For example, in [Figure 3-2](#) (the Device Array Selection window), the toolbar buttons indicate that you can copy, delete, or print an array. [“Using the HPL Buttons” on page 3-19](#) explains how to use these buttons.

Selection Type

The selection type allows you to specify the action that you want to take. In most of the displays, you can either open an already existing component or create a new component.

Component-Name Text Box

If you click **Create**, you must type a name for the new component in the **Device-Array** text box. (In [Figure 3-2](#), you must give a name for the new device array.)

Before you can type a name in the **Device Array** text box, you must click inside that text box to activate it. When the text box is active, it has a narrow black border. If you type a character that is not valid, the interface beeps at you, displays a message on the message line, and refuses to display the invalid character.

Component List Box

The component list box lists the components that currently exist in this project. If you click **Open** in the selection group, you must select a component from this list.

Notes Area

The notes area displays stored comments about the selected component. This area is not an active area. To store a comment about a component, you must select a component and use the **Notes** button. For more information about notes, see [“Notes Button” on page 3-24](#).

Message Line

The message line primarily gives instructions for the next logical action. The message line also gives an error message when an action fails or a completion message when a process is finished.

Buttons

The buttons across the bottom of the display let you indicate your next action. For a more complete discussion, see [“Using the HPL Buttons” on page 3-19](#).

Component-Definition Windows

After you select a component to create or modify and click **OK** in the component-selection window, you typically see the component-definition window. The component-definition window allows you to enter, edit, or delete values or items that describe the component. The device-array definition window has the following elements that are common to most of the other component-definition windows:

- Toolbar buttons
- Item-selection group
- Item-name text box
- Special-parameters group
- Item list box
- Perform group
- Message line
- Buttons

[Figure 3-3](#) shows an example of a component-definition window: the device-array definition window.

Figure 3-3
The Device-Array Definition Window

The screenshot shows a window titled "array_one" with the following components and annotations:

- Toolbar buttons:** "Print" and "Notes" icons at the top left.
- Item-selection type:** "Array Item Type" section with radio buttons for "Tape", "File" (selected), and "Pipe".
- Item-name text box:** "File Name:" field containing "/work/data/more_data".
- Special parameters:** "Tape Parameters" section with "Block Size" and "Tape Size" input fields, and radio buttons for "MB" and "GB".
- Items list box:** "Array Items" section containing a list of items: "FILE /work/data/some_data", "FILE /work/data/some_data", "PIPE /work/data/some_data", and "TAPE /work/data/some_data".
- Perform group:** "Perform" button and a group of radio buttons for "Add" (selected), "Edit", and "Delete".
- Message line:** "Message: Enter file name to load from or to create".
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom.

Toolbar Buttons

The buttons across the top of the display represent actions that you can take after you select a component from the component list box. For example, in [Figure 3-3](#) (the device-array definition window), the toolbar buttons indicate that you can print or make a note about an item. “[Using the HPL Buttons](#)” on [page 3-19](#) explains how to use these buttons.

Item-Selection Group

The item-selection group lets you specify the type of item that you want to edit or the type of action that you want to take. After you specify a choice in the item-selection group, other options become active. In the device-array definition window, the item-selection group is labeled **Array Item Type**. After you select **Tape**, **File**, or **Pipe** (only on UNIX), other options become active.

Item-Name Text Box

The item-name text box lets you specify the name or description of one of the items that makes up the component. For example, in the device-array definition window, you type the full pathname of a device in the item-name text box.

In the device-array definition window, the item-name text box is labeled **Tape Name**, **File Name**, or **Pipe Name** (only on UNIX), depending on the type of component that you select from the **Array Item Type** group.

Special-Parameters Group

When a component-definition window first appears, some of the choices are inactive (shown in gray letters). In general, the inactive choices are not meaningful until you specify some other characteristic of the component that you are editing.

The special-parameters group in the device-array definition window ([Figure 3-3 on page 3-10](#)) is the **Tape Parameters** group. The items in the special-parameters group are meaningful only for tapes. The choices in the **Tape Parameters** group become active only if you select **Tape** from the **Array Item Type** group. The choices in [Figure 3-3](#) are gray because **File** is selected in the **Array Item Type** group.

Item List Box

The item list box shows items that you already created to define the component. In the device-array definition window, this list is labeled **Array Items** and shows the tapes, files, and pipes (only on UNIX) that are already part of the current device array.

Perform Group

The **Perform** group lets you specify the action that you want to take. After you select an item and an action, you must click **Perform** to complete the action. For example, to add a new device in the device-array definition window, you must specify the name or description of the device and then click **Perform** to add it to the **Array List**.



Important: Remember to click **Perform** to complete the action that you designated in the **Perform** group.

Message Line

The message line primarily gives instructions for the next logical action. The message line also gives an error message when an action fails or a completion message when a process is finished.

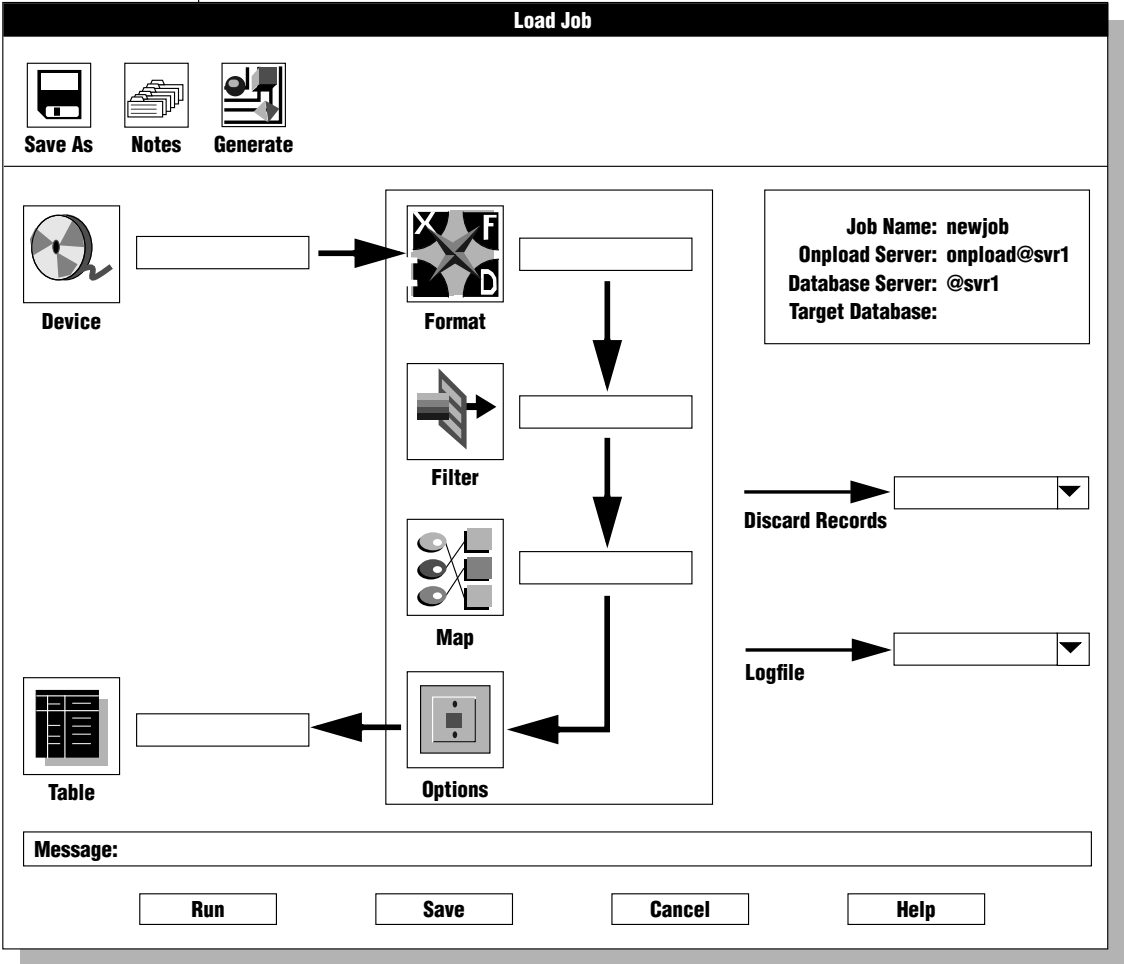
Buttons

The buttons across the bottom of the display let you indicate your next action. For a more complete discussion, see [“Using the HPL Buttons” on page 3-19](#).

Load Job and Unload Job Windows

The Load Job and Unload Job windows provide a visual presentation of the basic components that you choose for each job. [Figure 3-4](#) shows the Load Job window. The Load Job window and its functions are discussed in [Chapter 12](#). The Unload Job window and its functions are discussed in [Chapter 11](#).

Figure 3-4
The Load Job Window



Views Windows

A views window shows a graphic representation of the relationships among various **ipload** components. From a views window, you can search for specific components, select an existing component for editing, or create a new component. A views window does not allow you to change any values. To change the values of a component, you must display a component-definition window.

Accessing Views Windows

The following table lists the four types of views windows and gives instructions for how to access each view.

Window Name	Purpose	How to Access	Page
Format Views	Show the load and unload maps that are associated with a particular format	<ul style="list-style-type: none">■ Click Search in the Record Formats window■ Click Format in the Load Job window*■ Click Format in the Unload Job window*	2-15
Map Views	Show the databases, tables, queries, and formats that are associated with a map	<ul style="list-style-type: none">■ Click Search in the Load Record Maps window■ Click Search in the Unload Record Maps window■ Click Map in the Load Job window*■ Click Map in the Unload Job window*	9-7
Database Views	Show the tables in the database or the queries that are associated with the database	<ul style="list-style-type: none">■ Click Search in the Query window■ Click Table in the Load Job window*■ Click Query in the Unload Job window*	8-5
Filter Views	Show the formats that are associated with a particular filter	<ul style="list-style-type: none">■ Click Search in the Filter window■ Click Filter in the Load Job window*	10-11

* These options display the view window only if the corresponding text box is empty. If the text box includes the name of a component, the component-definition window is displayed.

Available Options in a Views Window

The four types of views windows operate in a similar manner. When a views window appears, you have the following options:

- Type in a component name and search for the component.
- Click a label associated with an icon to expand the view and see related components.
- Click an icon to open the component-definition window that allows you to edit the component values.
- Click **Create** to display the component-selection window that allows you to create a new component.

Searching for a Component

You can use the **Search** button in a views window to locate a specific component. Type the component name in the search text box and then click **Search**. The view displays only the component names that match the text string.

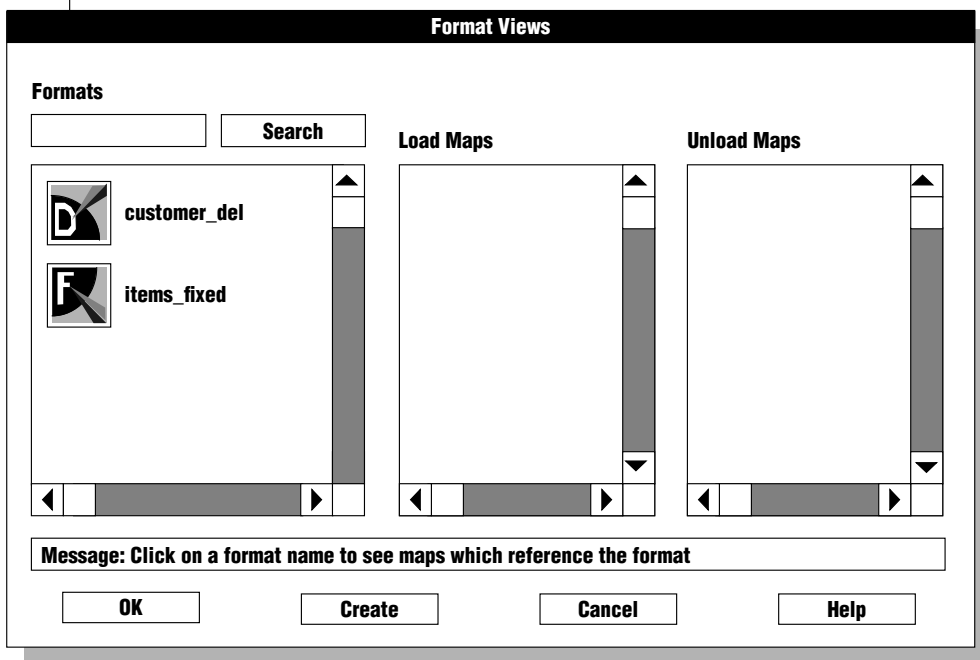
You can use the following wildcard search characters in the search text string.

Wildcard Symbol	Effect
?	Matches any <i>single</i> character
*	Matches any <i>string</i> of characters

Expanding the View

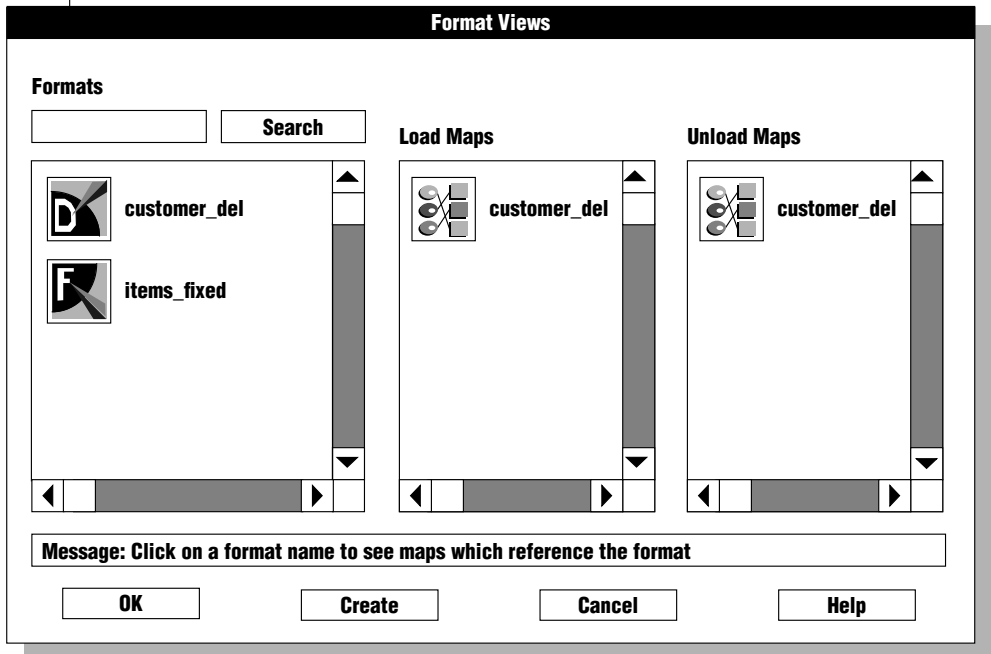
Three of the views windows expand their views to show related components. To expand the view, click an icon label (for example, **customer_del**) in the first pane. In the Database Views window, click an icon label in the second pane to expand the view further. [Figure 3-5](#) shows the Format Views window.

Figure 3-5
The Format Views Window



When you click an icon label in the Formats pane, the view expands to show maps that are related to your choice. [Figure 3-6](#) shows the expanded view.

Figure 3-6
Expanded View of a Format



You can click the desired icon to display a definition window for any format or map that is shown.

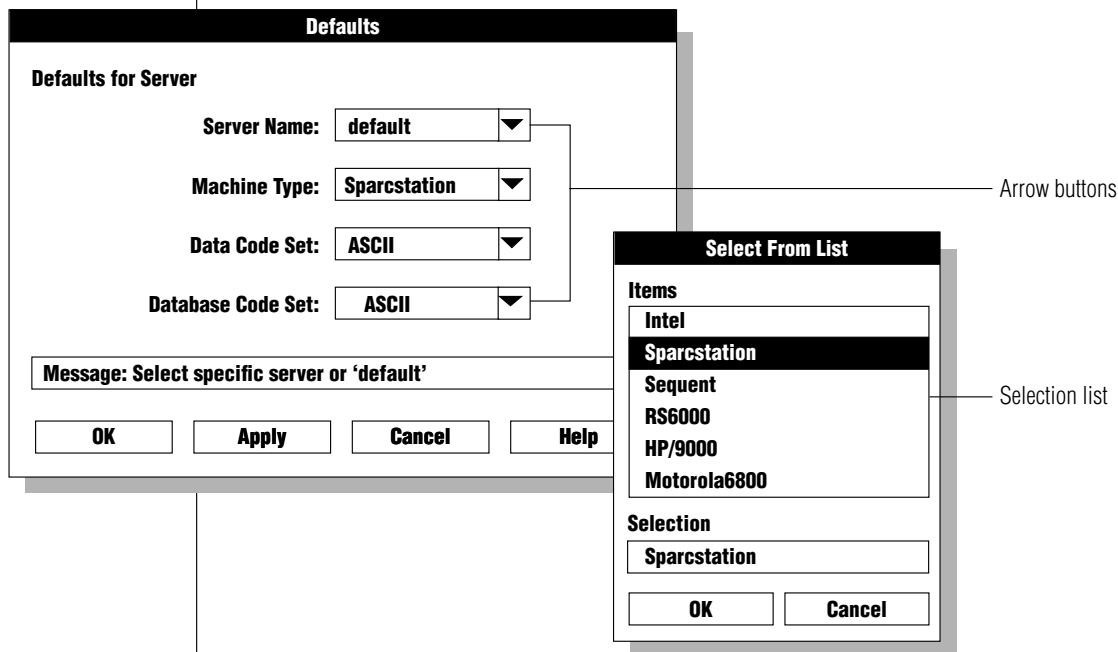
Selection-List Windows

A selection-list window lists the possible values for a text box. A down arrow that follows a text box indicates that you can use a selection list to see and select possible values for the text box. When you click the down arrow, the corresponding selection-list window appears.

Figure 3-7 shows the selection list that is available for the **Machine Type** text box in the Defaults window. After you select an item in the list box, click **OK**, and the item appears in the text box on the original window.

Figure 3-7

The Defaults Window and a Selection List



Tip: If your entry is rejected, look at the selection list. Your entry is invalid if it is not available in the selection list.

Selection-list windows are available for many text boxes throughout the HPL user interface. These windows have various names, but this document refers to them as selection lists.

Message Windows

A message window typically contains either a warning or an information update. A warning lets you verify or cancel the action that you have just chosen. An update informs you about the successful completion of an operation or explains why an operation failed. [Figure 3-8](#) shows a typical error message.



Figure 3-8
The Message Window










Using the HPL Buttons

After you move beyond the HPL main window, every window has at least one button to help you move through the interface. In general, buttons appear in three locations:






- Toolbar buttons appear across the top of the display.
- Icon buttons appear in the middle section of the display.
- Buttons appear across the bottom of the display.

Toolbar Buttons

Toolbar buttons appear at the top of many windows. The function of the window determines which buttons appear. The following sections describe the toolbar buttons. Buttons that appear in only one window are described with the specific window.

Button	Button Name	Purpose	Page
	Browse	Displays the Browse window	14-3
	Copy	Copies the selected component (format, map, query, filter, device, or project) to a new item	3-21
	Connect	Lets you reattach to an active unload (or load) job from the Unload (or Load) Job Select window	12-8
	Delete (trashcan)	Deletes the selected component (format, map, and so on)	3-23
	Delete (eraser)	Breaks the association between a database column and a data-file field	9-18
	File	Displays the Import/Export File Selection Window	8-13
	Find	Allows you to quickly locate a particular field or column in a map window	9-19
	Generate	Lets you generate jobs automatically	13-3
	Notes	Allows you to type descriptive text for an item	3-24

(1 of 2)

Button	Button Name	Purpose	Page
	Options	Displays an options window where you can change default values or supply additional parameters	11-12, 12-14
	Print	Prints the parameters for the selected item	3-27
	Save As	Saves a copy of the currently selected item (behaves in the same way as the Copy button)	3-21
	Search	Displays a views window where you can see the relationships among components	3-14
	Specs	Displays the Specifications window, where you can view the attributes for selected columns or fields	9-21

(2 of 2)

Browse Button

The **Browse** button lets you look through the files that show information about the load or unload jobs and any problems that the **onpload** utility found. For more information about the **Browse** button, see [“Browsing Options” on page 14-3](#).

Copy Button

The **Copy** button lets you copy a selected component. This feature can save you time when you are creating a new component. You can copy an existing component and then modify the copy with your changes.

You can copy one component at a time, or you can select and copy multiple components at the same time. You can copy components that are grouped under a project (filters, formats, maps, and queries) within the same project or to a different project.



If you copy a component within a project, you must give the copy a different name. If you copy a component to a different project, you can retain the name for the copy or give the copy a different name. If you copy multiple components, you must copy them to a different project. When you copy multiple components, the components retain their names.

Important: *Devices are not project specific. When you copy a device, you must give the copy a new name.*

To copy an existing format to a new format

1. In the HPL main window, select the project that includes the format that you want to copy.
2. Choose **Components→Formats** to access the Record Formats window.

For an example, see [“Creating a Fixed Format” on page 7-5](#).

3. Select the format that you want to copy.

This example assumes that the format to copy is **some_format**.

4. Click the **Copy** button.

The Copy Data window appears, as [Figure 3-9](#) shows. The Copy Data window displays a list of existing projects. The **Copy To** text box shows the name of the format that you are copying.

Figure 3-9
The Copy
Data
Window

5. Select the project to which you want to copy the format.
6. Type the name that you want to give to the copied format in the **Copy To** text box.

If you are copying the format to another project, you can keep the same name. You must change the name, however, if you are copying the format to the same project.

7. Click **OK**.
The display returns to the Record Formats window.
8. Click **Cancel** to return to the HPL main window.

Delete Button

The **Delete** button lets you delete one or more selected components.

To delete an existing format

1. In the HPL main window, select the project that includes the format that you want to delete.
2. Choose **Components→Formats** to access the Record Formats window.
For an example, see [“Creating a Fixed Format” on page 7-5](#).
3. Select the format that you want to delete.

4. Click the **Delete** button.

The Confirm Delete window appears, as [Figure 3-10](#) shows. The Confirm Delete window describes the impact of deleting this format. The text in this window is different for each of the component types.

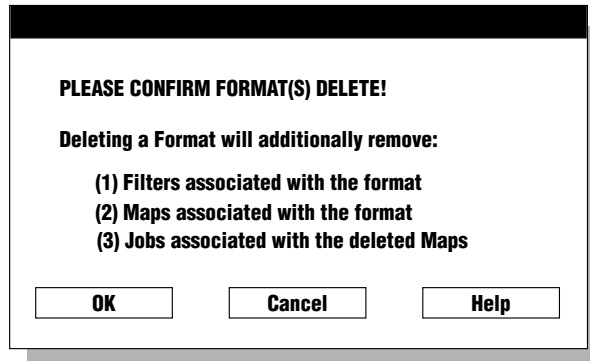


Figure 3-10
*The Confirm
Delete Window*

5. Click **OK** to confirm the deletion, or click **Cancel** to cancel it.
If you click **OK**, the format is deleted, as well as any associated maps, filters, and jobs.
6. Click **Cancel** to return to the HPL main window.

Notes Button

The **Notes** button lets you type descriptive text for an item. The text of the note is displayed in the **Notes** area in a window when you select the item. The **Notes** button is a useful tool for identifying **ipload** components, load jobs, unload jobs, and projects.

To create a note

1. Click the **Notes** button in a component-definition window.
The Notes window appears, as [Figure 3-11](#) shows.

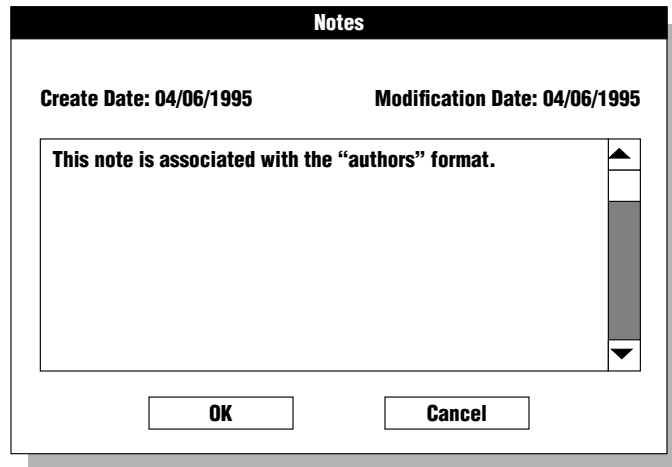


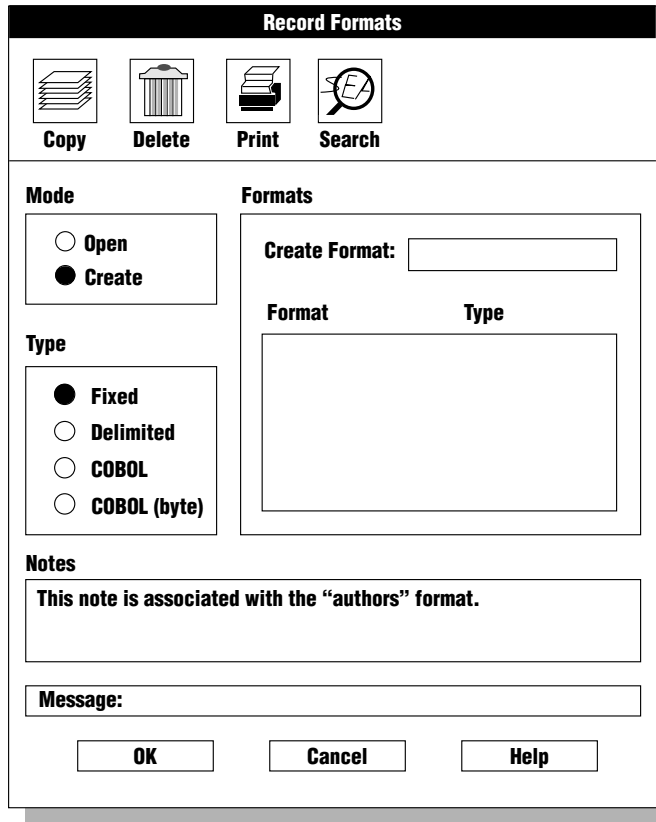
Figure 3-11
The Notes Window

2. Type the descriptive text in the **Notes** text box.
3. Click **OK** to store the note and return to the component-definition window.

When you select the component, the note text is displayed in the **Notes** area.

If you do not make any changes to a note, click **Cancel** instead of **OK**.

For example, the note created in [Figure 3-11](#) is associated with the **authors** format. The next time you go to the Record Formats page and select **authors**, **ipload** displays the note text, as [Figure 3-12](#) shows.



The screenshot shows a window titled "Record Formats". At the top is a toolbar with four icons: a stack of papers labeled "Copy", a trash can labeled "Delete", a printer labeled "Print", and a magnifying glass over a document labeled "Search". Below the toolbar, the window is divided into two main sections: "Mode" and "Formats".

The "Mode" section contains two radio buttons: "Open" (unselected) and "Create" (selected). Below this is a "Type" section with four radio buttons: "Fixed" (selected), "Delimited", "COBOL", and "COBOL (byte)".

The "Formats" section contains a "Create Format:" label followed by a text input field. Below this is a table with two columns: "Format" and "Type". The table is currently empty.

At the bottom of the window is a "Notes" section with a text area containing the text: "This note is associated with the 'authors' format." Below the notes section is a "Message:" label followed by a text input field. At the very bottom are three buttons: "OK", "Cancel", and "Help".

Figure 3-12
The Record Formats
Window with
Notes Text

The **ipload** utility stores the information that you type in the **note** table window of the **onpload** database. For a description of the **note** table, see ["Note Table" on page A-14](#).

Print Button

The **Print** button lets you print information that is associated with a component. Before you start **ipload**, you must set your workstation so that it can find a printer. For information about setting up a printer, see your operating-system manuals.

If you click the **Print** button in the map-definition window in [Figure 2-14 on page 2-26](#), the following printout results:

```
-----
LOAD MAP REPORT
-----

Project   : <default>
Name      : a_map

OPTIONS



      Database      Table      Format
      -----      -
      testdb        tab1       a_format

RECORD FORMAT MAP VIEW




      Format Field      Table Column      Option Data
      -----
      input1           col3
      input2           col2
      input3           col1
```

Icon Buttons



Icon buttons appear in the middle sections of the Load Job and Unload Job windows and views windows. The icon buttons represent various components. When you click it, each button opens another display. The following table shows and describes the icon buttons that are used in these windows.

Component	Description	Window	Action
 Device	The device or device array where the source files are located	Load Job Unload Job	<ul style="list-style-type: none">■ If the text box is empty, click the Device button to display the Device Array Selection window, where you can create or open a device type.■ If the text box has an entry, click the Device button to display the device-array definition window for that specific device or type the name of a different device in the text box.
 Filter	The filter that controls which records are selected from the data file for a database update (The use of a filter is optional.)	Load Job Filter Views	<ul style="list-style-type: none">■ If the text box is empty, click the Filter button to display the Filter Views window, where you select a filter and associated format. You can also create a filter from this window.■ If the text box has an entry, click the Filter button to display the filter-definition window for that specific filter or type the name of a different filter in the text box.■ In the Filter Views window, click the Filter button to display the filter-definition window for a specific filter.

(1 of 3)

Component	Description	Window	Action
 Format	The format of the source data used for this load or unload	Load Job Unload Job	<ul style="list-style-type: none">■ If the text box is empty, click the Format button to display the Format Views window, where you can select a format and associated map. You can also create a format from this window.■ If the text box has an entry, click the Format button to display the format-definition window for that specific format or type the name of a different format in the text box.■ In all views windows, click the Format button to display the format definition for a specific format. In these windows, the button shows only one of the three symbols (F, D, C) to indicate whether the type of format is fixed, delimited, or COBOL.
 Map	The map that correlates fields of the data source to database columns	Load Job Unload Job Map Views Format Views Database Views	<ul style="list-style-type: none">■ If the text box is empty, click the Map button to display the Map Views window, where you can select a map and associated table and format. You also can create a map from this window.■ If the text box has an entry, click the Map button to display the map-definition window for that specific map or type the name of a different map in the text box.■ In a views window, click the Map button to display the map-definition window for a specific map.
 Options	The options that let you specify characteristics of the load or unload	Load Job Unload Job	<ul style="list-style-type: none">■ Click the Options button to display the Load Options window. For a discussion of these options, see “Changing the Load Options” on page 12-14.

(2 of 3)

Component	Description	Window	Action
 Query	The query that selects data from the database table	Unload Job Database Views Map Views	<ul style="list-style-type: none"> ■ If the text box is empty, click the Query button to display the Database Views window from which you can select the table and associated map and format. ■ If the text box has an entry, click the Query button to display the Query-Definition window for that specific query or type the name of a different query in the text box. ■ In a views window, click the Query button to display the Query-Definition window for a specific query.
 Table	The database table into which the converted data will be loaded	Load Job Database Views Query Definition	<ul style="list-style-type: none"> ■ Click the Table button to display the Database Views window from which you can select the table and associated map and format. If an association is not apparent, click Create to create one. ■ Click the Table button in the Query-Definition window to choose a table and columns for the Select entry.

(3 of 3)

Buttons

The buttons across the bottom of the display let you indicate the next action. Most windows have one or more of the following buttons.

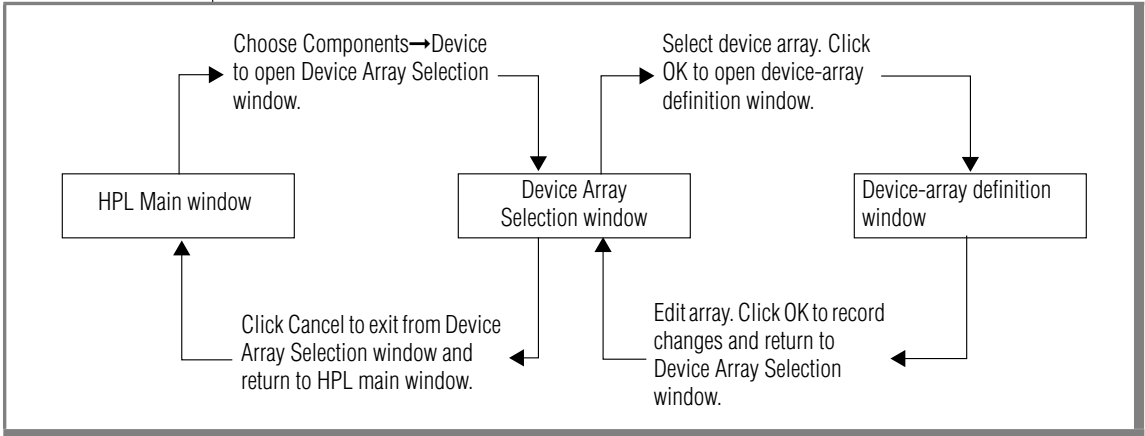
Button	Action
Apply	Save changes but do not exit.
Cancel	Do not save any changes. Exit to the previous display.
Create	Display the component-selection window.
Help	Display context-sensitive help in a separate window. For information about the online help, see “Using the Online Help” on page 3-31 .
OK	Save changes and exit to the previous display.

Use **OK** only when you have actually made a change on the display. If you are exiting from a series of displays, use **Cancel** to exit from the display.

[Figure 3-13](#) shows the use of **OK** and **Cancel**.

Figure 3-13

Using OK and Cancel from the HPL Main Window



Using the Online Help

The **Help** menu on the HPL main window has the following choices:

- Glossary
- Contents

The **Glossary** option opens a scrolling list of items. Select an item to see its definition. The **Contents** option takes you to the main contents page. This page directs you to discussions of various HPL topics.

If you click **Help** in any window other than the HPL main window, **Help** displays information that is related to the current window. After the **Help** window opens, you can click its **Help** button for more information about using the **Help** window.

Using UNIX Keyboard Commands to Move the Cursor

Instead of using the mouse to move from area to area in the HPL user interface, you can use keyboard commands to move the cursor. As you move around, the currently selected item is highlighted with a box.

The following table lists the cursor-moving keystrokes.

Keystroke	Result
TAB	Move from area to area. Sometimes used to move from tab stop to tab stop.
SHIFT-TAB	Back up; that is, move from area to area in reverse order.
CONTROL-TAB	Move from area to area when TAB is reserved to move from tab stop to tab stop.
Cursor keys	Move from item to item within a functional area.
SPACEBAR	Select the current item or action.

Most displays in the HPL user interface are divided into functional areas, such as toolbar buttons, selection group, component-name text box, component list box, and so on. Depending on the nature of the specific display, sometimes TAB moves from item to item (or even from tab stop to tab stop) within a major area. On other displays, TAB moves only between functional areas, and you must use SPACEBAR to move around within the functional area.

Defining Projects

In This Chapter	4-3
Project Organization	4-3
Projects Window	4-6
Creating a New Project	4-7
Selecting a Project	4-7

In This Chapter

The HPL lets you organize your work by specifying *projects*. A project is a collection of individual pieces that you use to load and unload data. A project can include load and unload jobs and the maps, formats, filters, and queries that you use to build the load and unload jobs. This chapter explains how to create a project and how projects are related. The individual components that you store in projects are described in later chapters.

Project Organization

The HPL uses only one database, **onpload**, to keep track of the preparation that you do for loading and unloading data. Using projects lets you organize your work into functional areas. For example, you might regularly transfer data to or from several unrelated databases. You could put all of the preparation for each database into a separate project.

When you first start **ipload**, **ipload** creates a project named **<default>**. If you prefer, you can select the **<default>** project and assign all of your work to that project. The HPL does not require that you create any additional projects. However, creating projects and putting separate tasks into distinct projects makes your work easier to maintain.

Figure 4-1 shows the relationships among projects, jobs, and components.

Figure 4-1
Illustration of Project Hierarchy

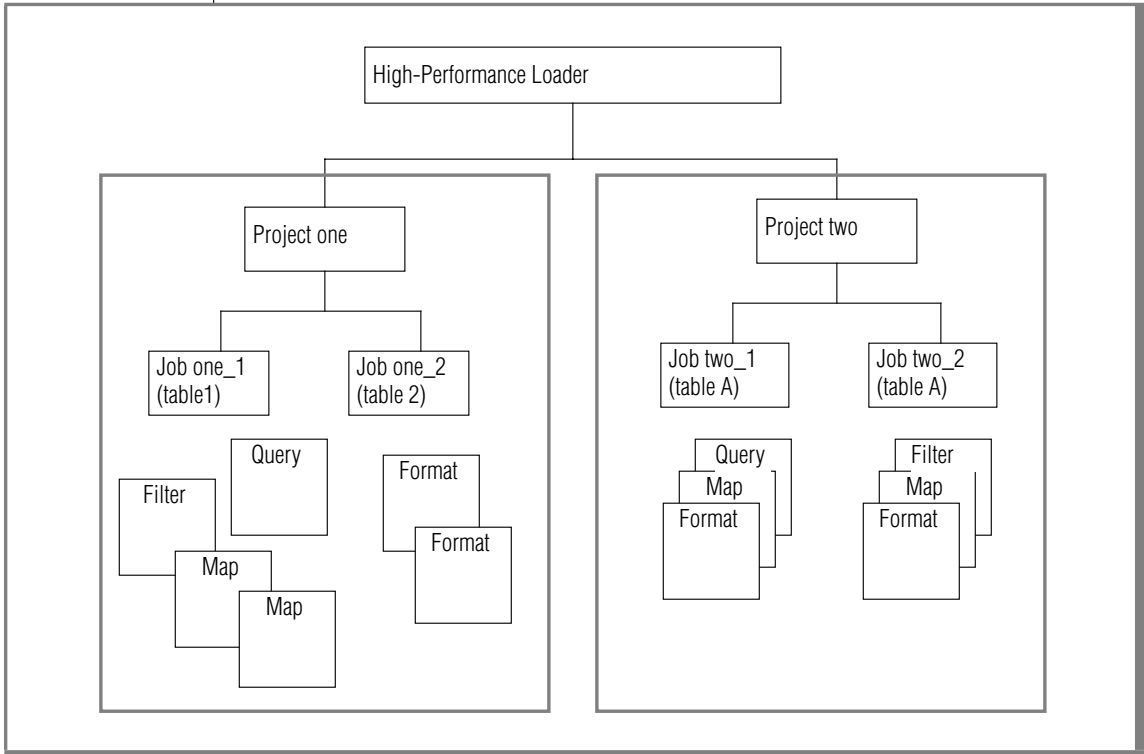


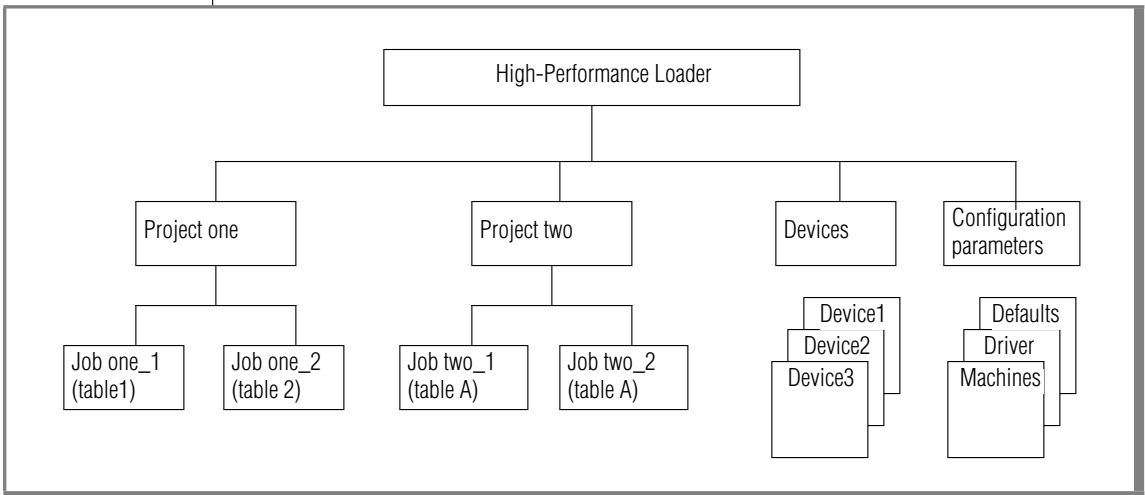
Figure 4-1 shows that jobs are linked directly to the projects. The format, map, filter, and query components belong to a project but are not directly linked to a job, as illustrated with Project one. In general, you create a format, map, and filter or query for each job, as shown with Project two. However, in some cases, you might use the same component for more than one job within a project.

For example, for reports about a medical study, you might want to create three reports: one about subjects under 50 years of age, one about subjects over 50, and one about all subjects. In that case, the description of how to find the information (the format and map) is the same for all three reports, but the selection of information (the query) is different for each report. (Formats, maps, and queries are described in detail in later chapters.)

All components (maps, formats, queries, filters, and load and unload jobs) that you create in a project are associated with that project in the **onpload** database. Components that are associated with a project are visible (usable) only when the project is selected. When you select a different project, a different set of components becomes available.

Device-array definitions and configuration parameters are not included in project definitions. [Figure 4-2](#) shows the components that the HPL uses. Each project is distinct, but the devices and configuration parameters apply to all projects.

Figure 4-2
Relationship of Projects, Devices, and Configuration Parameters



Projects Window

The Projects window (Figure 4-3) lets you select or create a project. After you select a project, you can copy the project, delete it, print the project parameters, or make a note that describes the project. The **ipload** utility stores project information in the **project** table of the **onpload** database. (The **project** table is described on [page A-15](#).)

Figure 4-3
The Projects Window

Projects

Copy Delete Print Notes

Projects

<default>
practice
stores_demo

Create Project:

Notes

Message: Enter project name to create, or select project to copy, delete, or print

OK Apply Cancel Help

Creating a New Project

To define a new project

1. Choose **Configure→Project** from the HPL main window.
The Projects window appears, as [Figure 4-3](#) shows.
2. Choose a name for the project and type it in the **Create Project** text box.
3. Click **Apply**.
The **ipload** utility creates the project but does not exit from the Projects window. You can create another new project, or you can use the toolbar buttons to manipulate the project.
4. Click **Cancel** to return to the HPL main window.

If you want to create one project and then exit, click **OK** instead of **Apply**.

Selecting a Project

The HPL provides two methods for selecting a project.

To select a project for a load or unload job or to edit components

1. Select the project name from the **Select Project** list box in the HPL main window.
2. Choose the action that you want to take from one of the menus on the HPL main window.

To select a project to edit

1. Choose **Configure→Project** from the HPL main window.
The Projects window appears, as [Figure 4-3](#) shows.
2. Select the project that you want to edit from the **Projects** list box.
3. Perform the desired edit actions (copy, delete, print, or describe with a note).
4. Click **Cancel** to return to the HPL main window.

Configuring the High-Performance Loader

In This Chapter	5-3
Configuring ipload	5-3
Selecting a Database Server	5-3
Using the Connect Server Window	5-4
Creating the onpload Database	5-5
Modifying the onpload Defaults	5-5
Defaults Window	5-5
Server Name	5-6
Machine Type	5-7
Data Code Set	5-7
Changing the onpload Defaults	5-7
Selecting a Driver	5-8
Drivers Window	5-9
Driver Class	5-10
Driver Name	5-10
Using the Drivers Window	5-10
Modifying the Machine Description	5-11
Machines Window.	5-12
Using the Machines Window	5-13

In This Chapter

This chapter describes how to configure the HPL. The configuration describes the type of computer, code sets, and other aspects of your database server environment. [“Performance” on page 15-10](#) describes how to modify the configuration to improve performance.

Configuring ipload

Configuration information is stored in the **onpload** database. The configuration tasks include:

- selecting a database server.
- modifying the **onpload** defaults.
- selecting or creating a driver.
- modifying the machine description.

Selecting a Database Server

The HPL needs to know the location of two databases: the **onpload** database and the *target database*. The target database is the Informix database into which you load data or from which you unload data. When you start **ipload**, **ipload** assumes that both the **onpload** database and the target database reside on the database server that the **INFORMIXSERVER** environment variable specifies. You can use the Connect Server window ([Figure 5-1 on page 5-4](#)) to specify different database servers.



The **sqlhosts** file or registry controls connectivity to database servers. The **ipload** utility scans the **sqlhosts** information to derive the lists of available database servers that the Connect Server window displays. For more information on how to configure connections, see the *Administrator's Guide*.

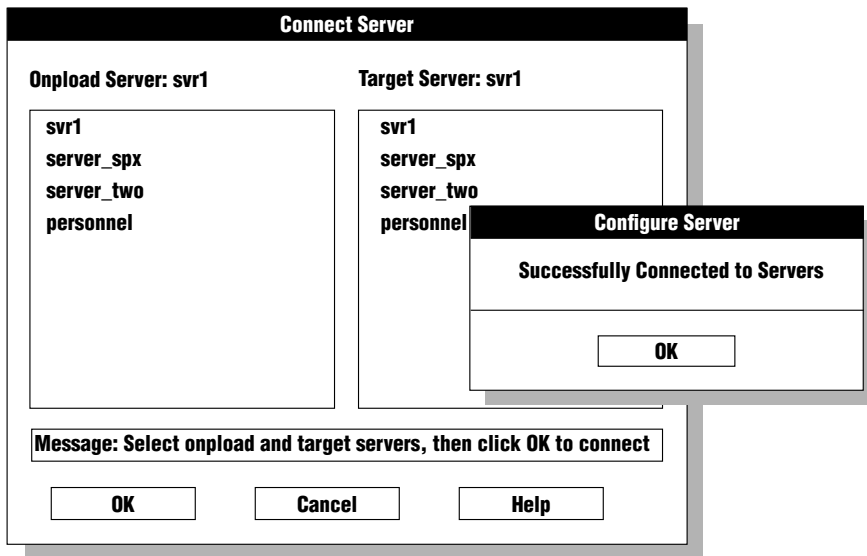
Important: You cannot use the alias **loghost** as a machine name in the **sqlhosts** file. The **loghost** alias is present on all computers; consequently, **onpload** cannot correctly identify computers based on this name.

Using the Connect Server Window

To select a database server

1. Choose **Configure→Server** from the HPL main window.
The Connect Server window appears, as [Figure 5-1](#) shows.

Figure 5-1
The Connect Server Window



2. Select the database server where the **onpload** database resides from the **Onpload Server** list box.
3. Select the database server that includes the database that you will load or unload from the **Target Server** list box.

4. Click **OK**.

The Configure Server confirmation window appears, as [Figure 5-1](#) shows.

5. Click **OK** in the Configure Server window to return to the HPL main window.

Creating the onpload Database

When you first start **ipload**, **ipload** creates an **onpload** database. If you use the Connect Server window to choose a different **onpload** database server, **ipload** creates another **onpload** database on that database server.

The default name of the database that the HPL uses is **onpload**. To give some other name to the HPL database, set the DBONPLOAD environment variable. For more information, see [“DBONPLOAD Environment Variable” on page 1-12](#).

Modifying the onpload Defaults

You must describe the computer environments of your database servers. This information applies to database servers. If you change the description of a database server, the changes apply to all jobs that you run on that database server. You can prepare a default computing environment that applies to all database servers that are not explicitly described.

Defaults Window

The Defaults window lets you change the following information:

- Server name
- Machine type
- Data code set

Figure 5-2 shows the Defaults window.

Figure 5-2
The Defaults Window

The **ipload** utility saves the information from the Defaults window in the **defaults** table of the **onpload** database. For more information about the **defaults** table, see [page A-2](#).



Tip: You can use DB-Access to examine the default values. The following tables in the **onpload** database contain default values: **defaults**, **delimiters**, **driver**, and **machines**.

Server Name

The **Server Name** text box specifies the database server with which the settings are associated. The information provided for the special server name **default** applies to all database servers for which no explicit information is provided. For example, if the majority of the database servers on your network (that will be using the HPL) are BrandX computers, **default** should describe the BrandX computers. To describe the computing environment of the other database servers on the network, specify the database server name.

The selection list that is associated with the **Server Name** text box lists the database servers that are in your **sqlhosts** file or registry. For information about the **sqlhosts** file or the registry, see the *Administrator's Guide*.

Machine Type

The **Machine Type** text box describes fixed-length, binary-format records. It defines the sizes and byte order of data in data files that the specified database server produces. The selection list that is associated with the **Machine Type** text box provides descriptions of several computers. You can use the **Machines** option on the **Configure** menu to add descriptions of other computers to this list (see [“Modifying the Machine Description” on page 5-11](#)).

Data Code Set

The **Data Code Set** text box specifies the character set of the data file. When you load data into a database, you can convert the character set of the data file into the character set of the database. For example, you can convert EBCDIC To ASCII, or any other character set that your system supports. Conversely, you can convert the data from a database into a selected character set when you unload data.

You can select a desired GLS code set from this selection list. The character set of the database is determined by the **DB_LOCALE** environment variable. For information about locales and code sets, see the *IBM Informix GLS User's Guide*.



GLS

Changing the onpload Defaults

To specify defaults for onpload

1. Choose **Configure→Defaults** from the HPL main window.
The Defaults window appears, as [Figure 5-2 on page 5-6](#) shows.
2. Update the values in each of the text boxes.
Click the down arrows to display selection lists that show possible values for each text box.
3. Click **Apply** to save the values and prepare the defaults for another database server.
4. Click **Cancel** to return to the HPL main window.

If you want to prepare the defaults for only one database server, click **OK** instead of **Apply**.

UNIX

Selecting a Driver

You can use pipe-device arrays to connect custom programs to the input or output of the **onpload** utility. Although pipe-device arrays provide the simplest way to customize the input or output of **onpload**, connecting the standard I/O of programs is less efficient than directly incorporating the functionality into **onpload**. ♦

The **ipload** utility identifies custom software by the driver name that you assign to the record-format definitions.

To incorporate custom file-handling software directly into the onpload program

1. Use the API described in [“Available API Support Functions” on page H-17](#) to write the driver.
2. Add the driver name to the **onpload** database.
For details, see [“Using the Drivers Window” on page 5-10](#).
3. Select the driver from the **Driver** selection list in the Format Options window.
For details, see [“Format Options” on page 7-28](#).


Drivers Window

The Drivers window lets you add information about custom drivers. After you add a custom driver name, you can assign the driver to the record-format definitions.

Figure 5-3 shows the Drivers window. The **Driver Name** list box displays the currently available drivers, their class, and type.

Figure 5-3
The Drivers Window

Drivers



Delete

Driver Name	Driver Class	Type
COBOL	COBOL	Standard
Delimited	Delimited	Standard
Fixed	Fixed	Standard

Create New Driver

Driver Class : Fixed

Driver Name :

Message: Select driver to delete or create new driver

OK

Apply

Cancel

Help

Driver Class

The **Driver Class** text box specifies the format type that the custom driver supports. The custom driver must produce data in a format that **onpload** supports. The **onpload** utility supports the following driver classes:

- Fixed
- Delimited
- COBOL

Driver Name

The **Driver Name** text box specifies the name of the custom-driver program. Before you can use the custom driver, you must add the program to your **onpload** shared library. For more information, see [“To build the shared-library file” on page H-6](#).

Using the Drivers Window

To add a custom-driver name

1. Choose **Configure→Driver** from the HPL main window.
The Drivers window appears, as [Figure 5-3 on page 5-9](#) shows.
2. Type the name of the driver that you are adding in the **Driver Name** text box.
3. Select the driver class.
4. Click **Apply** to save this driver and add another driver.
If you want to add only one driver, click **OK** instead of **Apply**.
5. When you finish, click **OK** to return to the HPL main window.

Modifying the Machine Description

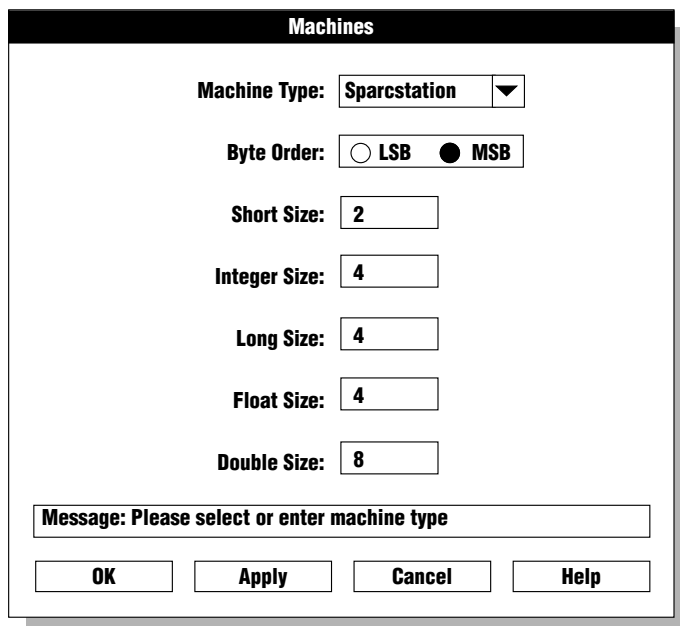
The information that the **Machines** option of the **Configure** menu stores describes the characteristics of a specific computer. The HPL uses these characteristics to control the conversion of binary data formats. Unless you are converting binary data, you do not need to be concerned about the machine description.

When you first start **ipload**, **ipload** stores the characteristics of several computers. You can select one of the existing computer types, modify an existing description, or create a new machine description.

You use the information from the Machines window when you prepare the defaults for the database servers on your network. (See [“Changing the onpload Defaults” on page 5-7](#).) The information from the Machines window is stored in the **machines** table of the **onpload** database. (For more information on the **machines** table, see [page A-10](#).) The default information for the HPL includes descriptions of the binary data sizes for several popular computers. If the default data does not include the computer from which you are reading data, you can create a description for that computer.

Machines Window

Figure 5-4 shows the Machines window. If you select Sparcstation from the **Machine Type** selection list, the following values appear in the Machines window.



The screenshot shows a window titled "Machines". Inside, there are several configuration fields:

- Machine Type:** A dropdown menu with "Sparcstation" selected.
- Byte Order:** Two radio buttons, "LSB" (unselected) and "MSB" (selected).
- Short Size:** A text box containing the value "2".
- Integer Size:** A text box containing the value "4".
- Long Size:** A text box containing the value "4".
- Float Size:** A text box containing the value "4".
- Double Size:** A text box containing the value "8".

Below these fields is a message box that says "Message: Please select or enter machine type". At the bottom of the window are four buttons: "OK", "Apply", "Cancel", and "Help".

Figure 5-4
The Machines Window

The Machines window displays the following information.

Item	Description
Machine Type	Name for the computer that this entry describes
Byte Order	Bit ordering of binary information for this computer. The two possible formats are LSB and MSB. In the LSB format, the least-significant bits of a value are at lower memory addresses. In the MSB format, the most-significant bits of a value are at lower memory addresses.
Short Size	Number of bytes required to hold a short integer value

(1 of 2)

Item	Description
Integer Size	Number of bytes required to hold an integer
Long Size	Number of bytes required to hold a long-integer value
Float Size	Number of bytes required to hold a floating-point value
Double Size	Number of bytes required to hold a double-sized floating-point value

(2 of 2)

Using the Machines Window

To edit the description of a computer

1. Choose **Configure→Machines** from the HPL main window.
The Machines window appears, as [Figure 5-4 on page 5-12](#) shows.
2. Click the down arrow and select a machine type from the selection list.
3. Modify the values as appropriate.
4. Click **Apply** to save the values and modify another machine description.
5. When you finish, click **Cancel** to return to the HPL main window.

If you want to modify the description of only one computer, click **OK** instead of **Apply**.

To add a computer type to the Machines list

1. Choose **Configure→Machines** from the HPL main window.
The Machines window appears, as [Figure 5-4 on page 5-12](#) shows.
2. Type the new name in the **Machine Type** text box.
3. Type an appropriate value in each text box.
4. Click **Apply** to save the values and add another computer type.
5. When you finish, click **Cancel** to return to the HPL main window.

If you want to add only one computer type, click **OK** instead of **Apply**.

Defining Device Arrays

In This Chapter	6-3
Device Arrays	6-3
Using Multiple Devices in a Device Array	6-3
Using the Device-Array Selection Window	6-4
Using the Device-Array Definition Window	6-6
Array Item Type Group	6-6
Device Text Box	6-7
Tape Parameters Group	6-8

In This Chapter

A *device array* groups several I/O devices so that the HPL can perform parallel processing of the input and output. When you specify multiple devices in a device array, **onpload** sets up separate, parallel streams of input or output when it performs a database load or unload.

Device Arrays

The HPL lets you use *device arrays* to group computer resources to perform parallel processing. Device arrays set up simultaneous access to one or more tape devices, files, or pipes (only on UNIX) so that the **onpload** utility can take advantage of parallel processing.

Device arrays are not project specific. You can use the same device array for a load or unload job on any of the projects that you define.

Using Multiple Devices in a Device Array

You can include files, pipes (only on UNIX), and tape devices in a single array. [“Devices for the Device Array” on page 15-13](#) discusses factors that you should take into account when you decide what devices to assign to an array. If your array includes pipe commands, **onpload** starts the pipe when it begins execution.

When the HPL unloads data, it assigns records to the devices of a device array in a round-robin fashion.

Using the Device-Array Selection Window

The Device-Array Selection window (Figure 6-1) lets you create a new device array or select an existing array.

If you select an existing array, you can edit that array or use one of the toolbar buttons to copy, delete, or print the array.

Device Array Selection

Copy **Delete** **Print**

Selection Type

☐ Open ☒ Create **Device Array:**

Current Arrays

largest
personnel
sales_sum_out

Notes

Message: Enter a device array name to create

OK **Cancel** **Help**

Figure 6-1
*The Device-Array
Selection Window*

To create a new device array

1. Choose **Components→Devices** from the HPL main window.
The Device-Array Selection window appears, as [Figure 6-1 on page 6-4](#) shows.
2. Click **Create** in the **Selection Type** group.
3. Select a name for the new device and type it in the **Device Array** text box.
4. Click **OK**.
The device-array definition window appears, as [Figure 6-2 on page 6-6](#) shows.

To open an existing device array

1. Choose **Components→Devices** from the HPL main window.
The Device-Array Selection window appears, as [Figure 6-1 on page 6-4](#) shows.
2. Click **Open** in the **Selection Type** group.
3. Select a device from the **Current Arrays** list box.
4. Click **OK**.
The device-array definition window appears, as [Figure 6-2 on page 6-6](#) shows.

Using the Device-Array Definition Window

Use the device-array definition window (Figure 6-2) to add, edit, or delete devices from an array.

Figure 6-2
A Partially Completed Device-Array Definition Window

new_array

Print **Notes**

Array Item Type

☐ Tape ☒ File ☐ Pipe

File Name:

Tape Parameters

Block Size: **Bytes** **Tape Size:** ☐ MB ☐ GB

Array Items

FILE /work/data/data_one

Perform

☒ Add ☐ Edit ☐ Delete

Message: Enter file name to load from or to create

OK **Cancel** **Help**

Array Item Type Group

The **Array Item Type** group lists the types of devices that you can include in a device array. You can mix different types of devices in a single array.

Device Text Box

Depending on the array item type that you selected from the **Array Item Type** group, the label for the text box where you type a device name is **Tape Device**, **File Name**, or **Pipe Command** (only on UNIX). Fill in this text box as follows.

Device Type	What to Type
Tape Device	The full pathname of the tape device (example: /dev/rmt/0)
File Name	The full pathname of the file (example: /work/mydata)
Pipe Command	The full pathname of the executable pipe command or shellscript (example: /tmp/g)

Tape Parameters Group

When you select Tape as the array item type, the **Tape Parameters** group becomes active (not gray), as [Figure 6-3](#) shows. You must type the block and tape size. The tape size must be greater than zero. The example shown in [Figure 6-3](#) is for UNIX. An example of a tape device name for Windows is: **11.1tape0**.

Figure 6-3
The Tape Parameters Group

The screenshot shows a window titled 'new_array'. At the top left are two icons: a printer labeled 'Print' and a notepad labeled 'Notes'. Below these is the 'Array Item Type' section with three radio buttons: 'Tape' (selected), 'File', and 'Pipe'. To the right of this is the 'Tape Device:' label followed by a text box containing '/dev/rmt/0o'. Below the 'Array Item Type' section is the 'Tape Parameters' group, which is active. It contains two text boxes: 'Block Size:' with '1638' and 'Bytes', and 'Tape Size:' with '10'. To the right of these are two radio buttons: 'MB' (selected) and 'GB'.

Windows

Windows does not support remote tape load and unload. ♦

To add devices to the device array

1. Click **Add** in the device-array definition window.
2. Click the device type in the **Array Item Type** group.
3. Type the full pathname of the device in the Device text box.
4. If you specified a tape device, the **Tape Parameters** group becomes active, as [Figure 6-3 on page 6-8](#) shows.
 - a. Type the block size in kilobytes.
 - b. Click **MB** (megabytes) or **GB** (gigabytes) to specify the units to use for the tape size.
 - c. Type the tape size.

5. When you have included all of the information for the device, click **Perform**.

The device that you added appears in the **Array Items** list box.

6. Repeat steps 2 through 5 to add other items to the device array.
7. When you have added all of the devices to the array, click **OK** to return to the Device-Array Selection window.

Your new array appears in the **Current Arrays** list box.

8. Click **Cancel** to return to the HPL main window.

To edit a device in the device array

1. Click **Edit** in the device-array definition window.
2. Select a device from the **Array Items** list box.
The selected item appears in the Device text box.
3. Edit the pathname and tape parameters, as appropriate.
4. Click **Perform**.
5. Click **OK** to return to the Device-Array Selection window.
6. Click **Cancel** to return to the HPL main window.



Tip: When you edit a device, you can change the pathname and the tape parameters, but you cannot change the array-item type (tape, file, pipe). If you need to change the device type, you must delete the item and then add a new item.

To delete a device from the device array

1. Click **Delete** in the device-array definition window.
2. Select a device from the **Array Items** list box.
3. Click **Perform**.
4. Click **OK** to return to the Device-Array Selection window.
5. Click **Cancel** to return to the HPL main window.

Defining Formats

In This Chapter	7-3
Formats.	7-3
Fixed-Length Records	7-4
Creating a Fixed Format.	7-5
Data Types Allowed in a Fixed Format	7-8
Bytes	7-10
Decimals.	7-10
Editing a Format	7-10
Creating a Fixed Format That Uses Carriage Returns	7-12
Creating a Fixed Format That Includes BYTE or TEXT Data	7-13
Inline Data	7-13
Data in a Separate File	7-15
Creating a Fixed Format That Includes Ext Type or Simple LO Data	7-16
Fixed-Length Data	7-16
Inline Data	7-16
Simple LO Data in a Separate File	7-18
Delimited Records	7-19
Creating a Delimited Format	7-19
Data Types Allowed in a Delimited Format.	7-20
Creating a Delimited Format That Includes BYTE or TEXT Data	7-21
Creating a Delimited Format with Extended Data Types	7-22
COBOL Records	7-24
Creating a COBOL Format	7-25
Picture and Usage Descriptions	7-26
Picture Description	7-26
Usage Description	7-26
Packed-Decimal Conversions	7-26

Other Formats	7-27
Fast Format	7-27
Fast Job	7-27
Format Options	7-28
Modifying Fixed and COBOL Formats	7-28
Modifying Delimited-Format Options	7-30
Format Views Window	7-32

In This Chapter

A *format* describes the structure of the data in a data file. Before you can import records from a data file into an Informix database or export records from a database to a data file, you must define a format that describes the data file. You do not need to define a format for the database because **ipload** already knows the schema (the organization) of the database table.

This manual uses the word *format* in two ways:

- To refer to the arrangement of data fields in a record of a data file
- To refer to the HPL component that documents the arrangement of the data fields

This chapter describes the formats that the HPL provides and shows how to prepare and edit the format component.

After you familiarize yourself with the concepts in this chapter, you might save yourself some work by using one of the Generate options to create formats automatically. For a description of these options, see [Chapter 13, “Generate Options.”](#)

Formats

Data files can be structured in a variety of ways. The HPL supports data-file records of the following formats:

- Fixed-length
- Delimited
- COBOL
- Other formats

You can define new format components at any time. Also, you can test your format before you actually load or unload data. For information about testing a format, see [“Previewing Data-File Records” on page 14-3](#).

The **ipload** utility includes options that let you modify the data before it is inserted into the database. For information about how to modify data, see [“Format Options” on page 7-28](#).

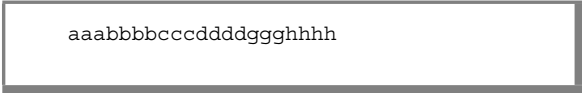
The **ipload** utility stores information about formats in the **formatitem** and **format** tables of the **onpload** database. For more information about the **formatitem** and **format** tables, see [page A-6](#) and [page A-8](#), respectively.



Important: To prepare the format component, you must know the format of the records in the data file. If you do not know the data-file format, you must get it from the person who provided the data file.

Fixed-Length Records

In *fixed-length* or *fixed-format* records, each field starts and ends at the same place in every record. A data file that contains data records of equal and constant length might be organized as [Figure 7-1](#) shows.



```
aaabbbbccccdddgghhhh
```

Figure 7-1
*Sample File with Fixed-Length
Records*

The data file illustrated in [Figure 7-1](#) has three records. Each record has a field of three characters followed by a field of four characters, so the total record length is seven characters. The file does not contain any separation between records; delimiters are unnecessary because all fields have the same length. VARCHAR data types are therefore always the fixed maximum size of the field.

When you define a fixed-length format, you specify the length of each field. The **ipload** utility calculates the offset for each field and the total length of the record from the field lengths that you supply.

Creating a Fixed Format

The Record Formats window and the Fixed Format definition windows let you create and define formats for fixed-length records.

To create a format for fixed-length records

1. Choose **Components→Formats** from the HPL main window.
The Record Formats window appears, as [Figure 7-2](#) shows.

Record Formats

Copy Delete Print Search

Mode

☐ Open
☒ Create

Type

☒ Fixed
☐ Delimited
☐ COBOL
☐ COBOL (byte)

Format

Format	Type

Notes

Message:

OK Cancel Help

Figure 7-2
The
Record
Formats
Window

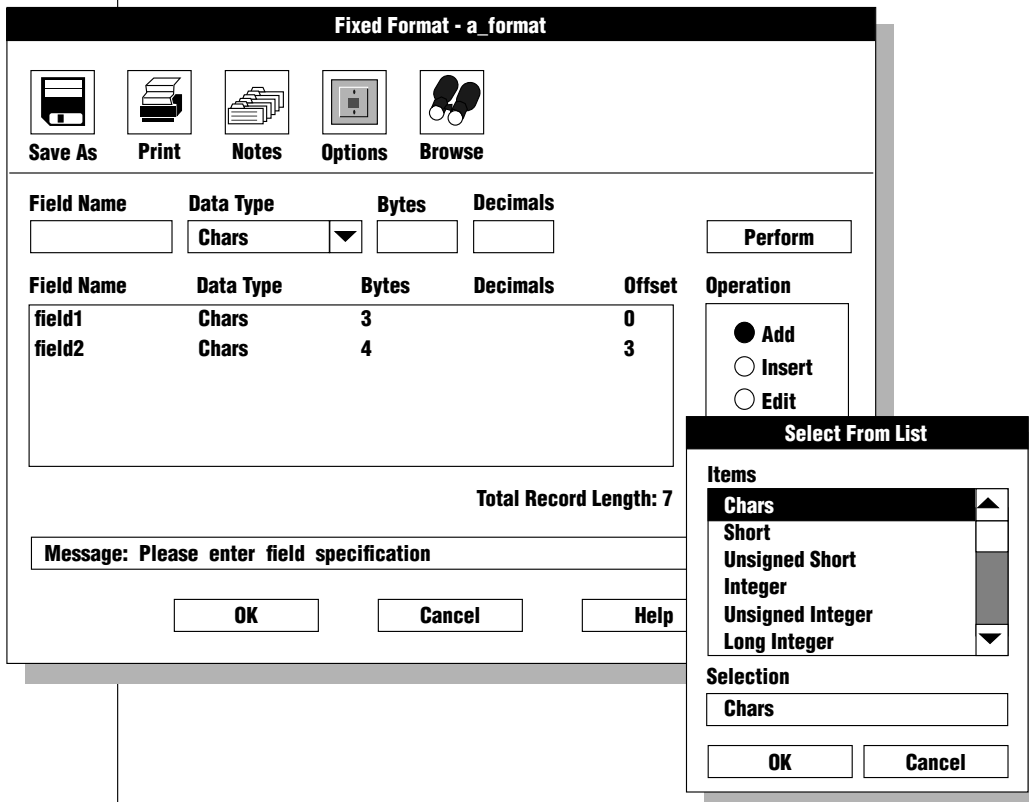
2. Click **Create** in the **Mode** group.
3. Click **Fixed** in the **Type** group.
4. Choose a name for the format and type it in the **Create Format** text box.

5. Click OK.

The Fixed Format definition window appears. The title bar includes the name that you chose for the format. [Figure 7-3](#) shows the Fixed Format definition window as it might appear after you prepare the format for the file that [Figure 7-1 on page 7-4](#) shows.

Figure 7-3

A Completed Fixed-Format Definition Window with an Open Selection List



6. Click **Add** in the **Operation** group.
7. Choose a name for the field and type the name in the **Field Name** text box.
8. Type the data type in the **Data Type** text box. For columns that contain user-defined types (UDTs) columns, you must choose an Ext Type format. For information about user-defined types, refer to *IBM Informix Guide to SQL: Reference*.

The down arrow next to the **Data Type** text box displays the selection list that appears at the right in [Figure 7-3. “Data Types Allowed in a Fixed Format” on page 7-8](#) describes the data types that appear in the selection list.

9. Type the appropriate value in the **Bytes** text box (or in the **Decimals** text box, if appropriate). If you chose the Ext Type String in the Data Type box, you must specify a size, or you will get an error. Field size depends on the type of data and its representation in the data file.
10. Click **Perform**.
After you click **Perform**, **ipload** calculates the proper offset for this field in the record and displays the value under the Offset heading, as [Figure 7-3](#) shows.
11. Repeat steps 7 through 10 for each field in your data file.
12. Click **OK** to save the format and return to the Record Formats window.
13. Click **Cancel** to return to the HPL main window.

Tip: Use the field name to map the data file to the database. You can type any name that you choose. You might find it easier to remember the names if you use the same name as the corresponding column of the database.



Data Types Allowed in a Fixed Format

You can use the following data types when you are preparing a fixed format.

Data Type	Description
Chars	ASCII format data
Short Unsigned Short Integer Unsigned Integer Long Integer Unsigned Long Float Double	The Machines description specifies the number of bytes required in fixed format for integers and floating-point values (see “Machines Window” on page 5-12.) When you select one of these data types, ipload sets the number of bytes.
Date	Date string
UNIX Date	A long integer interpreted as the system date from a UNIX system
Blob Length	The number of bytes of BYTE and TEXT (binary large object) information that follow this record
Blob File	A file that contains BYTE and TEXT data
BLOB and CLOB	A file that contains BLOB and CLOB data
Simple LO Length	The number of bytes of simple large object data (BYTE and TEXT data) that follow this record
Simple LO File	The name of a file that contains a Simple LO data (BYTE and TEXT data)
Int8	An eight-byte integer type
Serial8	An eight-byte serial column
Ext Type String	The ASCII representation of an extended data type (Ext Type) value

(1 of 2)

Data Type	Description
Ext Type String Length	<p>The length of an ASCII Ext Type value</p> <p>The Ext Type value follows at the end of the input/output record.</p> <p>Use Ext Type String Length data type if you have data that contains null UDT values.</p>
Ext Type Binary	<p>The binary representation of an Ext Type value</p>
Ext Type Binary Length	<p>The length of the binary representation of an Ext Type value.</p> <p>The binary value follows at the end of the input/output record.</p> <p>Use Ext Type Binary Length data type if you have data that contains null UDT values.</p>

(2 of 2)

The HPL supports several data types under the Ext Type mechanism. As a result, the specific names of these data types do not appear in the data-type selection list. For the following data types, choose the appropriate Ext Type data type:

- INT8
- LVARCHAR
- SERIAL8
- BLOB
- BOOLEAN
- CLOB
- Collection data types
- Distinct data type
- Opaque data types
- Row types

Bytes

In [Figure 7-3 on page 7-6](#), the **Bytes** text box specifies the number of characters that the field occupies in the record. In the **Bytes** text box, you must set the number of bytes for your data types. Although **ipload** uses default information to calculate an offset if you create a new format that has a new length, it does not readjust the lengths for existing formats. To change the default information, see [“Machines Window” on page 5-12](#). The **ipload** utility automatically calculates the total length of the data file as you add each field description.

Decimals

In [Figure 7-3 on page 7-6](#), the **Decimals** text box specifies the number of decimal places that are displayed when you convert floating-point types to ASCII. You can set the number of decimals only for the **Float** and **Double** data types.

Editing a Format

After you create and save a format, you might need to add a new field, insert a new field, edit a field, or delete a field. The process for editing an existing format is essentially the same, regardless of the file type. The following example uses a fixed-format file, but the same procedure applies to COBOL AND delimited files also.

To add a new field description to the format

1. Open the Fixed Format definition window.
For more information, see [“Creating a Fixed Format” on page 7-5](#).
2. Click **Add** in the **Operation** group.
3. Type the field specifications in the text boxes at the top of the window.
4. Click **Perform** to add the new field at the end of the list.
5. Click **OK** to save your changes and return to the Record Formats window.
6. Click **Cancel** to return to the HPL main window.

To insert a new field into the format

1. Open the Fixed Format definition window.
For more information, see [“Creating a Fixed Format” on page 7-5](#).
2. Click **Insert** in the **Operation** group.
3. Select the field *before which* you want to insert the new field.
4. Type the field specifications in the text boxes at the top of the window.
5. Click **Perform** to insert the new field before the selected field.
6. Click **OK** to save your changes and return to the Record Formats window.
7. Click **Cancel** to return to the HPL main window.

To edit the description of a field

1. Open the Fixed Format definition window.
For more information, see [“Creating a Fixed Format” on page 7-5](#).
2. Click **Edit** in the **Operation** group.
3. Select the field from the list of fields.
4. Change the desired information.
5. Click **Perform**.
6. Click **OK** to save your changes and return to the Record Formats window.
7. Click **Cancel** to return to the HPL main window.

To delete a field description from the format

1. Open the Fixed Format definition window.
For more information, see [“Creating a Fixed Format” on page 7-5](#).
2. Click **Delete** in the **Operation** group.
3. Select the field that you want to delete.
4. Click **Perform** to delete the field.
5. Click **OK** to save your changes and return to the Record Formats window.
6. Click **Cancel** to return to the HPL main window.

Creating a Fixed Format That Uses Carriage Returns






A fixed-format data file often includes a carriage return (new line) at the end of each record, such as the data file in [Figure 7-4](#).

20 chars	20 chars	15 chars	2 chars
John Brown	100 Main St.	Citadel	LA
Mary Smith	3141 Temple Way	Chesapeake	AZ
Larry Little	44 Elm Rd. #6	Boston	MA

Figure 7-4
Fixed-Format File
with Carriage
Returns

When you prepare the format for this data file, you must include a dummy field for the carriage return. When you create the load map for this format (see “[Creating a Load Map](#)” on page 9-6), do not link the dummy field to a database column. [Figure 7-5](#) shows the format for the data file illustrated in [Figure 7-4](#).

Fixed Format - a_format

 Save As
  Print
  Notes
  Options
  Browse

Field Name	Data Type	Bytes	Decimals
	Chars		

Perform

Field Name	Data Type	Bytes	Decimal	Offset	Operation
name	Chars	20		0	<input checked="" type="radio"/> Add
street	Chars	20		20	<input type="radio"/> Insert
city	Chars	15		40	<input type="radio"/> Edit
state	Chars	2		55	<input type="radio"/> Delete
dummyCR	Chars	1		57	

Total Record Length: 58

Message: Please enter field specification

OK

Cancel

Help

Figure 7-5
Fixed Format with
Dummy Entry for
Carriage Return

Creating a Fixed Format That Includes BYTE or TEXT Data

You can organize the BYTE or TEXT data in a fixed-format data file in the following ways:

- Inline data
- Data in a separate file

Inline Data

BYTE or TEXT data that is included as part of a fixed-format data file is called *inline* data. When BYTE or TEXT data is inline, the data-file record has two parts: a fixed-length part and a variable-length part. For example, a record with two fields and BYTE or TEXT data might be organized as [Figure 7-6](#) shows.

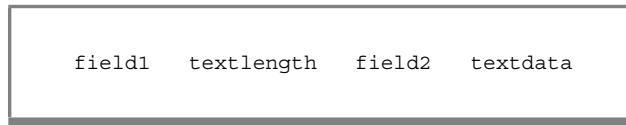


Figure 7-6
*Organization of a
Record that Includes
Inline TEXT Data*

The *length* of the TEXT data is included in the fixed-length part of the record. The actual TEXT data is inserted at the end of the fixed-length part of the record. The HPL reads the TEXT length from the fixed-length part of the record and uses that length to read the actual TEXT data. The HPL also uses the TEXT length to calculate the offset to the beginning of the next record.

Figure 7-7 shows the format definition of a record with inline BYTE and TEXT data. The arrows show how the HPL puts the record into the database. The arrows from **field 1** and **field 2** indicate entries in fixed-length format. The split arrow shows that the HPL uses the **TEXT length** information to find the **TEXT data** and insert it into the table. The HPL does not insert the **TEXT length** into the database.

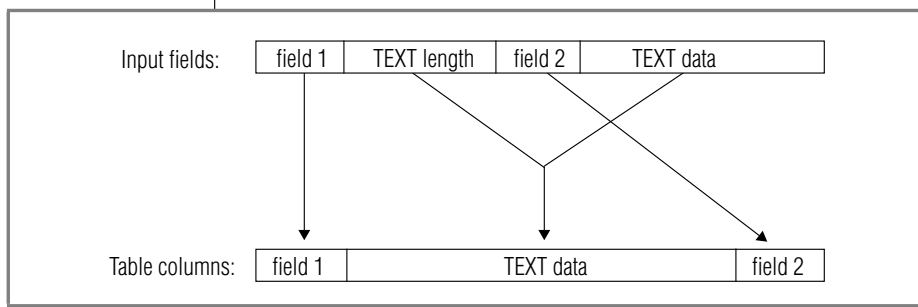


Figure 7-7
Inline TEXT Data

When you define the format in the format-definition window, select **Blob Length** as the data type for the **textlength** field. Figure 7-8 shows the format for the example in Figure 7-6. The format does not include an entry for TEXT data.

Fixed Format - b_format

Save As Print Notes Options Browse

Field Name	Data Type	Bytes	Decimals
	Chars		

Perform

Field Name	Data Type	Bytes	Decimal	Offset	Operation
field1	Chars	10		0	<input checked="" type="radio"/> Add
TEXTlength	Blob Length	4		10	<input type="radio"/> Insert
field2	Chars	30		14	<input type="radio"/> Edit
					<input type="radio"/> Delete

Figure 7-8
Fixed Format That
Includes TEXT Data

When you create a map to link the input fields that are defined by the format to the columns of a database table (see [Chapter 9](#)), connect the **textlength** input field to the table column that contains the TEXT data.

Data in a Separate File

You can also store BYTE and TEXT data in separate files. During a load, BYTE and TEXT data files are read and inserted into the database. During an unload, the file is created and BYTE and TEXT data is written to the file. When the fixed-format input contains the pathname of a data file, the HPL uses that pathname to insert data into a column of the database table, as [Figure 7-9](#) shows. When you prepare the format, select **Blob File** for the data type.

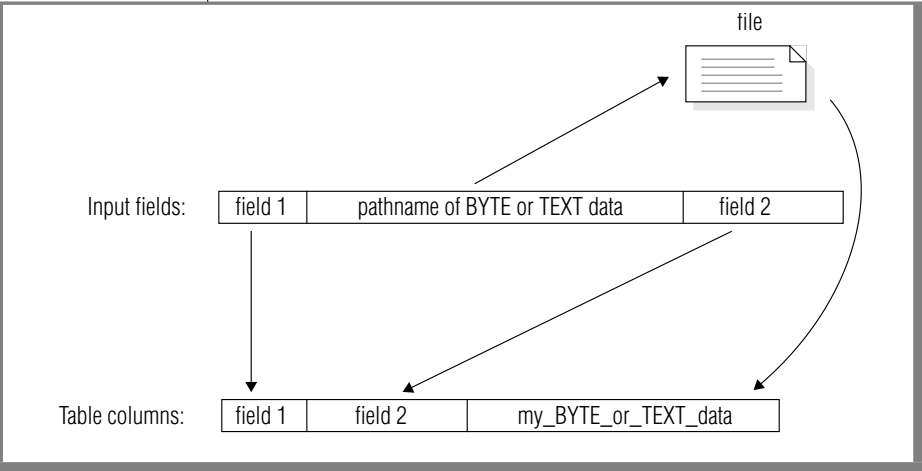


Figure 7-9
*BYTE or TEXT
Data in a File*

When you create a map to link the fields of the input record to the columns of a database table (see [Chapter 9](#)), link the name of the BYTE or TEXT file with the BYTE or TEXT column. The arrows in [Figure 7-9](#) illustrate how the HPL inserts the BYTE or TEXT data into the column.

Creating a Fixed Format That Includes Ext Type or Simple LO Data

You can organize the Ext Type or Simple LO data in a fixed-format data file in the following ways:

- Fixed-length data
- Inline data

Fixed-Length Data

When you designate a field as an Ext Type String or Ext Type Binary data type, you specify that the data will occupy a fixed amount of space, similar to the behavior of a fixed-length Chars data type. With fixed-length data format, you must specify the number of bytes that the field occupies in the record.

Inline Data

Simple LO data or varying-sized Ext Type data that is included as part of a fixed-format data file is called *inline* data. When Ext Type or Simple LO data is inline, the data-file record has two parts: a fixed-length part and a variable-length part. For example, a record with two fields and a Simple LO might be organized as [Figure 7-10](#) shows.

field1	Simple LO length	field1	Simple LO data
--------	------------------	--------	----------------

Figure 7-10
Organization of a Record that Includes Inline TEXT Data

The *data-type length* of the Ext Type or Simple LO data is included in the fixed-length part of the record. The actual TEXT data is inserted at the end of the fixed-length part of the record. The HPL reads the Ext Type or Simple LO length from the fixed-length part of the record and uses that length to read the actual Ext Type or Simple LO data. The HPL also uses the Ext Type or Simple LO length to calculate the offset to the beginning of the next record.

Figure 7-11 on page 7-17 shows the format definition of a record with inline Ext Type or Simple LO data. The arrows show how the HPL puts the record into the database. The arrows from **field 1** and **field 2** indicate entries in fixed-length format. The split arrow shows that the HPL uses the **Simple LO length** information to find the Simple LO data and insert it into the table. The HPL does not insert the Simple LO length into the database.

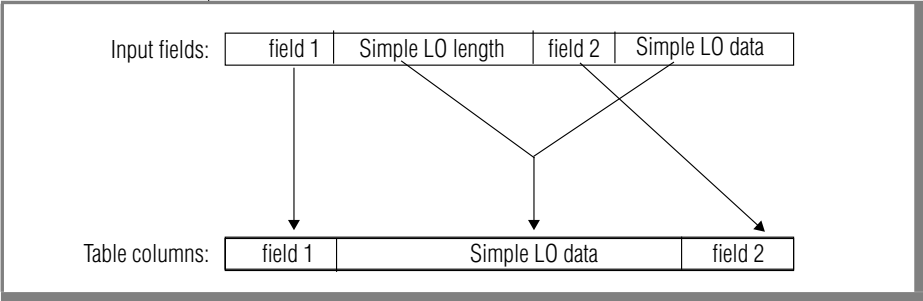


Figure 7-11
Inline TEXT Data

When you define the format in the format-definition window, select the appropriate data-type length data type (**Ext Type String Length**, **Ext Type Binary Length**, or **Simple LO Length**) for the data-type length field. Figure 7-12 shows the format for the example in Figure 7-10. The format does not include an entry for Simple LO data.

Fixed Format - b_format

Save As Print Notes Options Browse

Field Name	Data Type	Bytes	Decimals
<input type="text"/>	Chars <input type="button" value="v"/>	<input type="text"/>	<input type="text"/>

Perform

Field Name	Data Type	Bytes	Decimal	Offset	Operation
field1	Chars	10		0	<input checked="" type="radio"/> Add
Simple LO Length	Simple LO Length	4		10	<input type="radio"/> Insert
field2	Chars	30		14	<input type="radio"/> Edit
					<input type="radio"/> Delete

Figure 7-12
Fixed Format That Includes a Simple LO



Important: *Ext Type binary-length format is not supported for complex types.*

When you create a map to link the input fields that are defined by the format to the columns of a database table (see [Chapter 9](#)), connect the data type length input field to the table column that contains that particular data. In this case, connect the **Simple LO length** input field to the table column that contains the Simple LO data.

Simple LO Data in a Separate File

You can also store Simple LO data in separate files. During a load, Simple LO data files are read and inserted into the database. During an unload, the file is created and BYTE and TEXT data is written to the file. When the fixed-format input contains the pathname of a data file, the HPL uses that pathname to insert data into a column of the database table, as [Figure 7-13](#) shows. When you prepare the format, select **Blob File** for the data type.

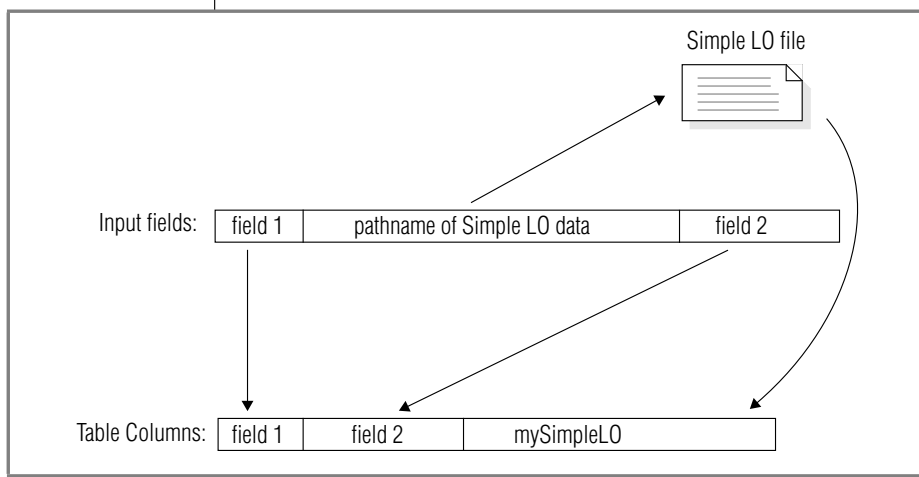


Figure 7-13
*Simple LO Data
in a File*

When you create a map to link the fields of the input record to the columns of a database table (see [Chapter 9](#)), link the name of the Simple LO file with the Simple LO column. The arrows in [Figure 7-13](#) illustrate how the HPL inserts the Simple LO data into the column.

Delimited Records

Delimited records are records whose fields can vary in length. In a data file that contains delimited records, the records and fields are separated by a *delimiter*. The following data file uses a vertical bar (|) as the field delimiter and a carriage return as the record delimiter:

```
John Brown|100 Main St.|Citadel|LA|215/887-1931
Mary Smith|3141 Temple Way|Chesapeake|AZ|415/812-9919
Larry Little|44 Elm Rd. # 6|Boston|MA|617/184-1231
```

The **ipload** utility uses the vertical bar and carriage return as the default field and record delimiters. For instructions on how to choose a different delimiter, see [“Modifying Delimited-Format Options” on page 7-30](#).

Creating a Delimited Format

To create a format for delimited records, follow the same steps as in [“Creating a Fixed Format” on page 7-5](#), with the following modifications:

- In step 3, click **Delimited** in the **Type** group.
- Omit step 9. When you use delimited records, **ipload** does not need byte or decimal information.

Data Types Allowed in a Delimited Format

You can use the following data types when you prepare a delimited format.

Data Type	Description
Chars	Normal ASCII data
BYTE or TEXT File	File that contains BYTE or TEXT data
TEXT Data	TEXT data is formatted as ASCII text. If the text includes carriage returns (new lines) or delimiters, a backslash (\) must precede those characters.
BYTE or TEXT HexASCII	BYTE or TEXT data that is formatted in ASCII hexadecimal. onpload translates the data into binary before it loads the data into the database.
BLOB or CLOB	File that contains BLOB or CLOB data
Simple LO File	The name of a file that contains Simple LO data (BYTE and TEXT data)
Simple LO Text	Simple LO data that is formatted as ASCII text. If the text includes carriage returns, newline characters, or delimiters, a backslash (\) must precede those characters
Simple LO HexASCII	Simple LO data that is formatted in ASCII hexadecimal. onpload translates the data into binary before it loads the data into the database.
Ext Type String	The ASCII representation of a Data Type (Ext Type) value
Ext Type HexASCII	The HexASCII representation of an Ext Type value

Creating a Delimited Format That Includes BYTE or TEXT Data






In a delimited format, BYTE or TEXT data can be characters, hexadecimal data, or in a separate file. [Figure 7-14](#) shows a data record that has two fields of character data followed by a field of character BYTE or TEXT data, a field of hexadecimal BYTE or TEXT data, and the pathname of a file that contains BYTE or TEXT data.

```
field1|field2|TEXT data|BEEEF6699|/bbs/kaths/data2jn95
```

Figure 7-14
Sample Data-File
Record that Includes
BYTE or TEXT Data

[Figure 7-15](#) shows a format for the file in [Figure 7-14](#).

Delimited Format- c_format

 **Save As**  **Print**  **Notes**  **Options**  **Browse**

Field Name	Field Type
x_field	Chars
y_field	Chars
a_txt_data	Blob Text
a_hex_data	Blob HexAXCII
a_txt_data_path	Blob File

Operation

☒ Add
☐ Insert
☐ Edit
☐ Delete

Message: Please enter field specification

OK **Cancel** **Help**

Figure 7-15
Delimited Format
with BYTE or
TEXT Entries

Creating a Delimited Format with Extended Data Types

Extended data types include the following data types:

- User defined (distinct and opaque)
- Collection
- CLOB and BLOB
- Row

In a delimited format, CLOB and BLOB data are always written to a file. CLOB data can be ASCII or hexadecimal data. BLOB data can be binary. The pathname of the file, the file size, and the offset are embedded in the data file during unload. However, when you perform a load, you only need to specify the pathname.

```
field1|ROW('abcd', NULL|SET{1}|10|/work/photo.jpg|20|  
/work/text.txt
```

Figure 7-16
*Sample Data-File
Record that
Includes Extended
Data Types*

Figure 7-16 shows a data record that has a field of character data (`field1`), a row-type field (`ROW('abcd', NULL)`), a collection-type field (`SET{1}`), an integer field (`10`), a BLOB-type field (`/work/data/photo.jpg`), an integer field (`20`), and a CLOB -type field (`work/data/text.txt`).

Figure 7-17 shows a sample format for the file in Figure 7-16.

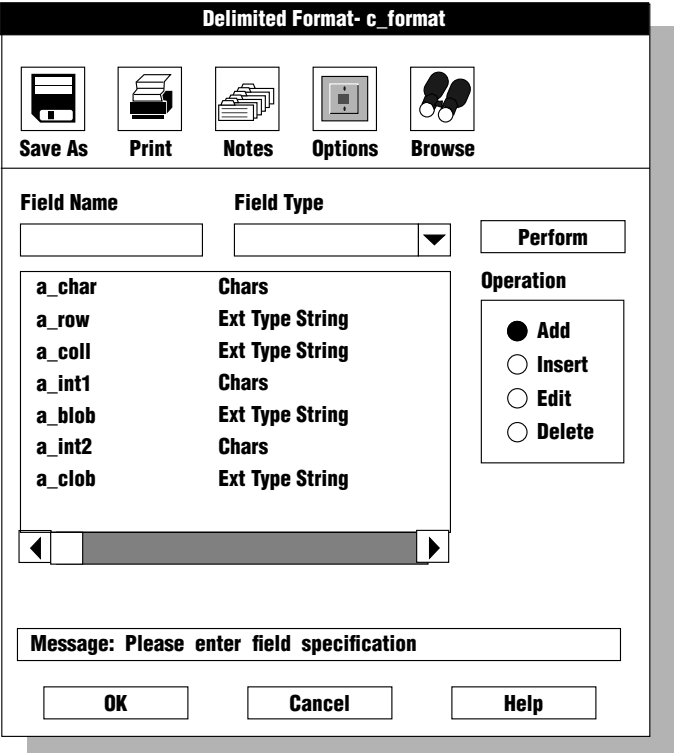


Figure 7-17
*Delimited Format
with Extended Data
Type Entries*

If you unload using a format that is similar to Figure 7-17, the unloaded data might resemble Figure 7-18. You can use this data to load the same file again, instead of using the pathname.

```
field1|ROW('abcd', NULL|SET{1 }|10|0,51f4,blob67e9.8ac|20|
0,c 692,clob67e9.9ad
```

Figure 7-18
*Sample Data-File
Record that Includes
Extended Data Types*

By default, the **clob67e9.8ad** and **blob67e9.8ac** files in Figure 7-18 will be written to **/tmp**. To change the default, modify the path in the **PLOAD_LO_PATH** environment variable.

COBOL Records

The HPL supports COBOL sequential data files that do not contain internal indexing. [Figure 7-19](#) shows the COBOL-Format definition window for preparing a COBOL format.

COBOL Format - see_cobol

Save As Print Notes Options Browse

Field Name Picture Usage

Chars Perform

Name Picture Usage Operation

Add Insert Edit Delete

Record Length: 0

Message: Please enter field specification

OK Cancel Help

Figure 7-19
Fixed-Format
Definition Window
for a COBOL Format

Creating a COBOL Format

To create a format for COBOL records, follow the same steps as in [“Creating a Fixed Format” on page 7-5](#), with the following modifications:

- In step 3, click **COBOL** in the **Type** group in the Record Formats window.
- In step 8, type the COBOL picture description in the **Picture** text box.
- In step 9, type the data type in the **Usage** text box.
- The arrow displays the selection list of available data types for the **Usage** text box, as [Figure 7-20](#) shows.

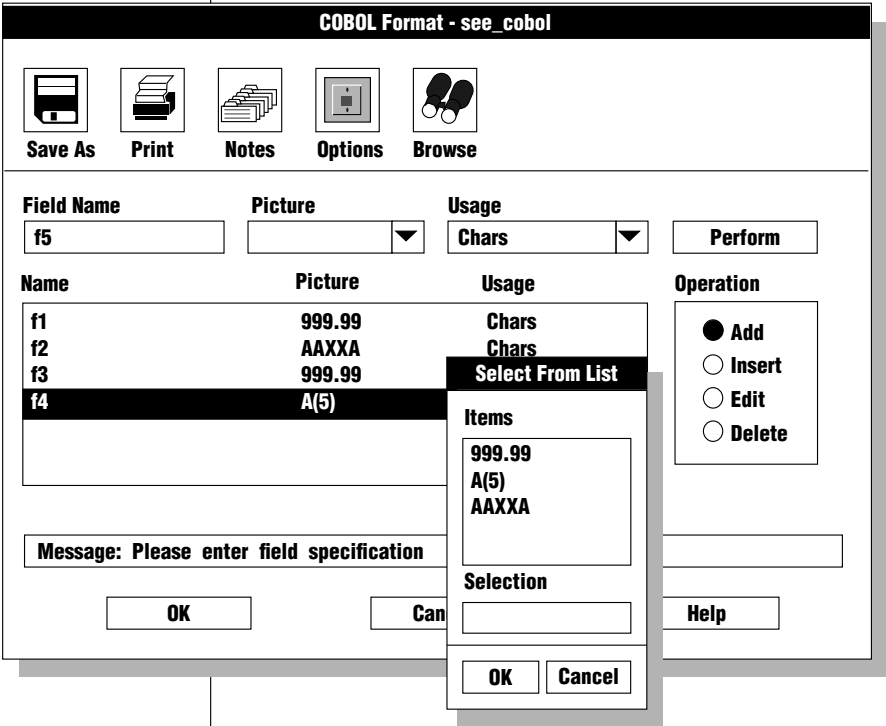


Figure 7-20
*Fixed-Format
Definition Window
for a COBOL Format*



Important: Ext Types are not supported in COBOL format.

Picture and Usage Descriptions

The picture and usage description must conform to ANSI-COBOL 85 specifications. For information about COBOL picture strings, see the documentation for the COBOL compiler.

Picture Description

The picture description must match the record file descriptor (FD) from the COBOL program that generates or will use the data. For information on COBOL formats, see your COBOL programmer's manual.

Usage Description

The usage description must match the data-field type described in the FD descriptor of the COBOL program. If the COBOL program does not include a usage clause, select the Chars (character) option for the usage.

Packed-Decimal Conversions

When values are converted to packed-decimal formats, supply a picture clause that matches the picture clauses in the COBOL programs that use the data. Otherwise, the COBOL interpretation of the values will be wrong.

The following table lists some examples of appropriate picture clauses.

Picture	Input Data	Output Data	COBOL value =
9999999	123	0000123C	123
9999V99	123	0000123C	1.23
9999.99	123	0000123C	1.23
9999V99	-123.22	0012322D	-123.22

Other Formats

In addition to delimited, fixed, and COBOL formats, the HPL provides two other formats for loading and unloading data: *fast format* and *fast job*. These formats are not included on the Record Formats window because the format specifications are predefined; you do not need to make any choices.

Fast format and fast job are the most efficient ways to load and unload data because their formats are predefined.

Fast Format

Fast format loads or unloads data in which each individual column uses Informix internal format. You can reorder, add, or delete columns, but you cannot do any kind of conversion on the column itself.

Select **Fixed internal** in the Generate window to get this type of load. For information about the Generate window, see [Chapter 13, “Generate Options.”](#)

Fast Job

A fast job loads or unloads an entire row of an Informix database table in Informix internal format. A fast job is also called a *raw load* or a *no conversion job*. For more information, see [“Format Type Group” on page 13-11.](#)

The **-fn** flag of the **onpload** command-line utility specifies a fast job. For information about the **onpload** utility, see [Chapter 16, “The onpload Utility.”](#)

Format Options

The format options let you change the default driver, the character set, the default computer type, and the delimiters. Information about the format options is stored in the **formats** table of the **onpload** database. For more information about the **formats** table, see [page A-8](#).

Modifying Fixed and COBOL Formats

You can modify the following options for fixed and COBOL formats.

Option	Description
Character set	The code set that is used to translate the data in the data table
Driver	The driver that is used with the delimited format. For more information, see “Selecting a Driver” on page 5-8 .
Machine	The machine type that produced the data files. For more information, see “Modifying the Machine Description” on page 5-11 .

For a fixed format, you can select a desired GLS code set from the **Character Set** selection list. For information about locales and code sets, see the *IBM Informix GLS User’s Guide*. ♦

GLS

To modify the options for fixed and COBOL formats

1. Display the format-definition window for the desired format.
To do this, follow the steps in [“Creating a Fixed Format” on page 7-5](#).
2. Click **Options**.
The Options window (in this example, the Fixed Format Options window) appears, as [Figure 7-21](#) shows.

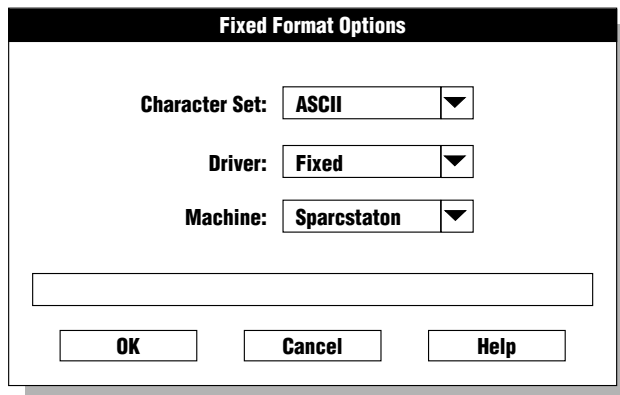


Figure 7-21
*The Fixed
Format
Options
Window*

3. Modify the options as appropriate.
4. Click **OK** to save your options and return to the format-definition window.

Modifying Delimited-Format Options

The Delimiter Options window ([Figure 7-22](#)) lets you modify the following options of the delimited format.

Option	Description
Character set	The code set used to translate the data in the data table.
Driver	The driver that is used with the delimited format. For more information, see “Selecting a Driver” on page 5-8 .
Delimiting characters	<p>The <i>delimiting characters</i>, which are sometimes called <i>record separators</i> and <i>field separators</i>, indicate the beginning and end of records and fields.</p> <p>You can specify the delimiting characters in ASCII, HEX, OCTAL, or DECIMAL format.</p>

GLS

You can select a desired GLS code set from the **Character Set** selection list. For information about locales and code sets, see the *IBM Informix GLS User's Guide*.



To modify the options for delimited formats

1. Display the Delimited Format definition window.
To do this, complete the steps in “Creating a Fixed Format” on page 7-5 with the following modification: in step 3, click **Delimited**.
2. Click the **Options** button in the Delimited Format definition window.
The Delimiter Options window appears, as Figure 7-22 shows.

Figure 7-22
The Delimiter Options Window

HEX	OCTAL	CONTROL	ASCII
00	000		Null
01	001	Ctrl-A	sch
02	002	Ctrl-B	stx
03	003	Ctrl-C	etx

3. Modify the options that you want to change.
4. Click **OK** to save your changes and return to the Delimited Format definition window.



Tip: You can use the **DBDELIMITER** environment variable to set the field delimiter for the **dbexport** utility and the **LOAD** and **UNLOAD** statements. However, do not use **DBDELIMITER** with the **HPL** because the **onpload** utility does not use this environment variable.

For more information on environment variables, see the *IBM Informix Guide to SQL: Reference*.

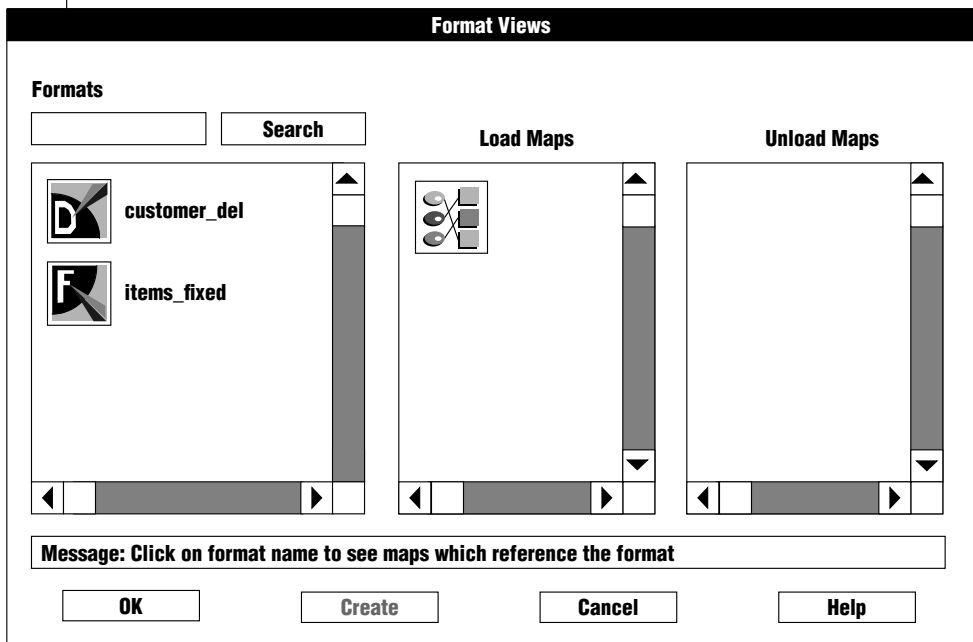
Format Views Window

The Format Views window lets you display a list of the formats and load and unload maps that are associated with a project. The Format Views window also lets you create or edit a format. The Format Views window appears in the following situations:

- When you click **Format** in the Unload Job window and no format name is in the **Formats** text box
- When you click **Search** in the Query window

Figure 7-23 shows a Format Views window. “Views Windows” on page 3-14 discusses how to use Views windows.

Figure 7-23
The Format-Views Window



Defining Queries

In This Chapter	8-3
Queries	8-4
Creating a Query	8-4
Using the Table Button	8-7
Editing the WHERE Clause	8-11
Editing a Query	8-12
Exporting and Importing Queries.	8-13
Importing a Query	8-13
Exporting a Query.	8-16
Database Views Window	8-17

In This Chapter

This chapter describes how to create, edit, and use queries. During the unload process, the HPL uses a *query* to select data from a database table (or tables), as [Figure 8-1](#) shows. The HPL can process any valid SQL statement.

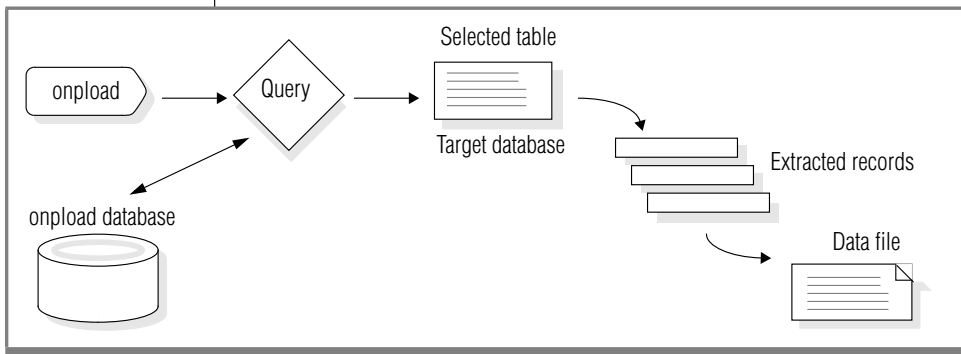


Figure 8-1
*Extracting
Data from a
Database Table*

Queries

The **ipload** query component lets you build an SQL statement. This manual uses the word *query* in two ways:

- To refer to the SQL statement that selects information from the database
- To refer to the HPL component that lets you build and store the SQL statement

The **ipload** utility stores query information in the **query** table of the **onpload** database. (For more information about the **query** table, see [page A-16](#).) The SQL statement is stored as TEXT data.

Creating a Query

Use the Query window to create a new query.

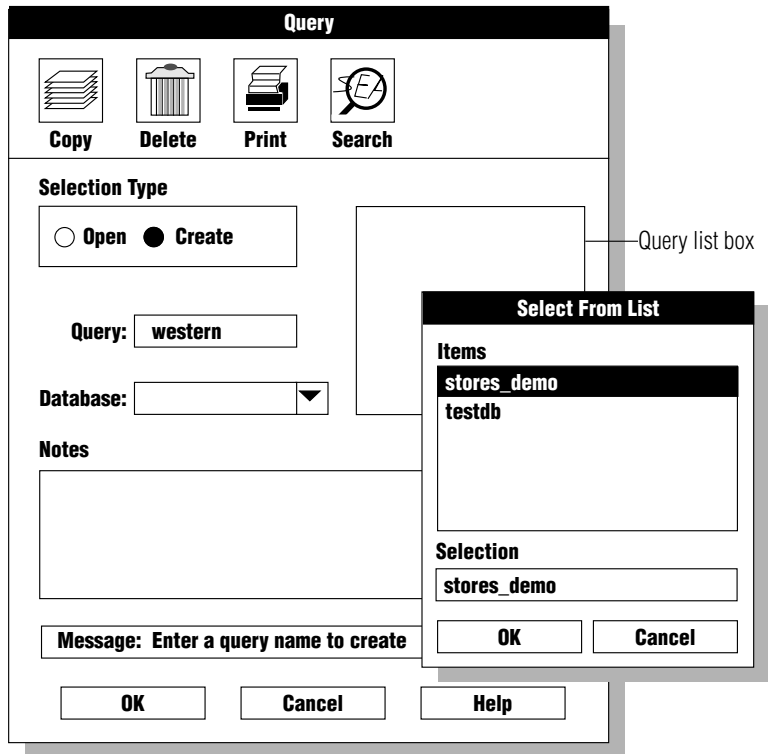
To create a query

1. Choose **Components→Query** from the HPL main window.
The Query window appears, as [Figure 8-2](#) shows.
2. Click **Create** in the **Selection Type** group.
3. Choose a name for your query and type it in the **Query** text box.

4. In the **Database** text box, type the name of the database that contains the table(s) from which you want to extract data, or click the down arrow to select from a database selection list.

Figure 8-2 shows the Query window with the **Query** text box completed and **stores_demo** selected from the selection list.

Figure 8-2
The Query Window



5. Click OK.

The query-definition window appears, as Figure 8-3 on page 8-6 shows. The name that you chose for your query appears in the title bar.

6. Type your query in the **Select**, **From**, and **Where** text boxes.

Figure 8-3 shows the following simple query of the **customer** table of the **stores_demo** database:

```
SELECT customer.fname, customer.lname,  
       customer.zipcode  
FROM customer  
WHERE zipcode > 50000
```

If you prefer, you can type the entire query into the **Select** text box. If you later edit the query, **ipload** divides the query into SELECT, FROM, and WHERE clauses.

The screenshot shows a window titled "western" which is the Query-Definition Window. At the top, there are five icons with labels: "Save As", "Notes", "Print", "File", and "Table". Below these icons, the text "Database: stores_demo" is displayed. The window is divided into three main sections: "Select", "From", and "Where". The "Select" section contains the text "customer.fname, customer.lname, customer.zipcode". The "From" section contains the text "customer". The "Where" section contains the text "zipcode > 50000". At the bottom of the window, there is a message box that says "Message: Enter select, from, and where part of select query in appropriate window". Below the message box are three buttons: "OK", "Cancel", and "Help".

Figure 8-3
The Query-
Definition
Window

7. Click **OK** to save the query and return to the Query window.

The query that you just created now appears in the **Query** list box at the right-hand side of the Query window.

8. Click **Cancel** to return to the HPL main window.

Using the Table Button

The **Table** button displays the Column Selection window. You can use the Column Selection window to build queries by selecting tables and columns. The **ipload** utility inserts the selected columns and tables into the appropriate text boxes of the query-definition window.

To use the Column Selection window

1. Follow the steps in [“Creating a Query” on page 8-4](#) to display the query-definition window ([Figure 8-3](#)).

2. Click **Table**.

The Column Selection window appears, as [Figure 8-4 on page 8-8](#) shows. The **Tables** list box includes synonyms and views that are valid for the local database server.

3. Select a table.

After you select a table, the right-hand pane displays a list of the columns in that table. [Figure 8-4](#) shows the Column Selection window with the **customer** table selected.

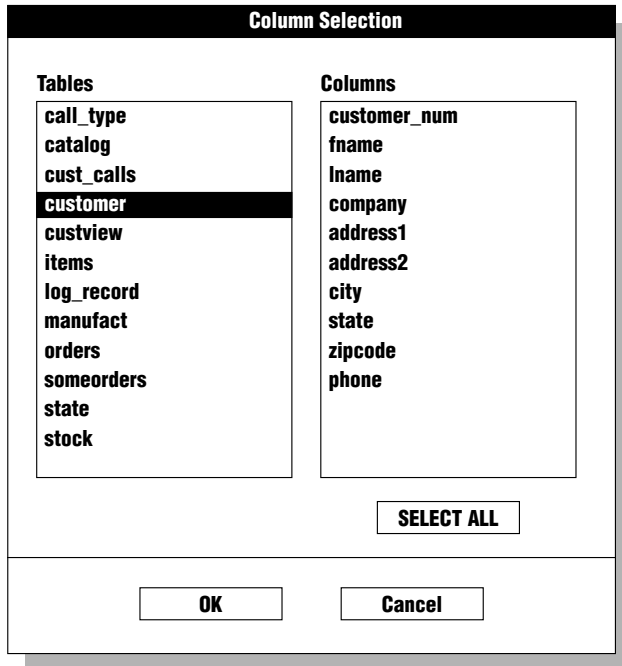


Figure 8-4
*The Column
Selection
Window After
Selecting a
Table*

4. Select one or more columns to use in the query.
Use the following actions to select columns.

To Select	Perform This Action
A single column	Select that column.
All columns	Click Select All .
Consecutive columns	Select the first column. Move to the final column and hold down SHIFT while you select that column.
Several nonconsecutive columns	Select a column. Hold down CONTROL while you select additional items.

Figure 8-5 shows the Column Selection window with several columns selected.

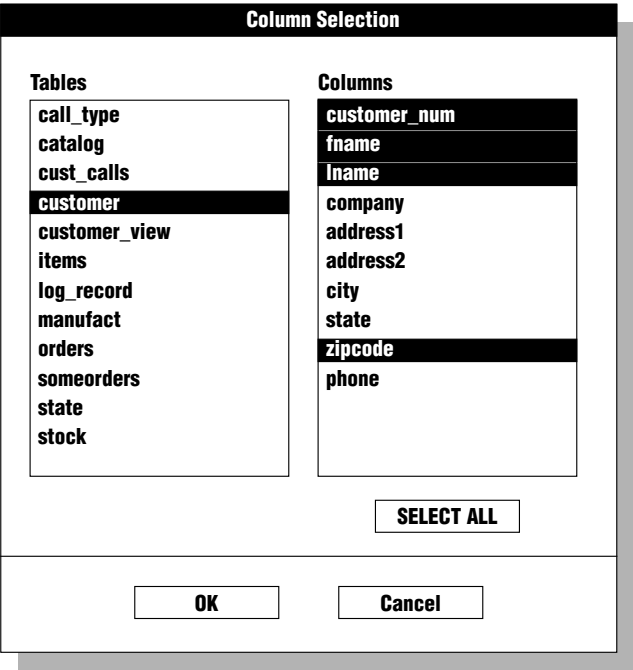


Figure 8-5
*Columns
Selected
from a Table*

5. When you finish selecting columns, click **OK** to return to the query-definition window.

When the query-definition window reappears, the mouse cursor changes to a pointing hand and the message line reads:

Position Cursor Where Column Data to be Inserted

6. Select the **Select** text box or the **Where** text box.

[Figure 8-6](#) shows columns inserted into the **Select** text box. The **ipload** utility also inserts the table name into the **From** text box.

The screenshot shows a window titled "test_query". At the top, there is a toolbar with icons for "Save As", "Notes", "Print", "File", and "Table". Below the toolbar, the window is divided into three main sections: "Select", "From", and "Where". The "Select" section contains the text "customer.customer_num, customer.fname, customer.lname, customer.zipcode". The "From" section contains the text "customer". The "Where" section is empty. To the right of each text box is a vertical scroll bar. At the top right of the window, it says "Database: stores_demo". At the bottom, there is a message box that says "Message: Enter select, from, and where part of select query in appropriate window". Below the message box are three buttons: "OK", "Cancel", and "Help".

Figure 8-6
*The Query-
Definition Window
After Using the
Table Button*

7. Repeat steps 2 through 6 to add columns from other tables.
8. Modify the text in the **Where** text box so that it is a valid WHERE clause.

See [“Editing the WHERE Clause” on page 8-11](#).

9. Click **OK** to save the query and return to the Query window.
10. Click **Cancel** to return to the HPL window.

Editing the WHERE Clause

When you use the Column Selection window (Figure 8-4 on page 8-8) to select a column or columns for the WHERE clause, the selected columns appear in the **Where** text box. The `=?` symbols indicate where you must provide match conditions. Figure 8-7 shows the result when you choose `zipcode` and `customer_num` from the `customer` table.

The screenshot shows a window titled "test_query" with a toolbar at the top containing icons for "Save As", "Notes", "Print", "File", and "Table". Below the toolbar, the "Database: stores_demo" is indicated. The window is divided into three main sections: "Select", "From", and "Where". The "Select" section contains the text "customer.customer_num, customer.fname, customer.lname, customer.zipcode". The "From" section contains the text "customer". The "Where" section contains the text "customer.zipcode=? and customer.customer_num=?". At the bottom of the window, there is a message box that says "Message: Enter select, from, and where part of select query in appropriate window" and three buttons: "OK", "Cancel", and "Help".

Figure 8-7
The Where Text Box
Entry After You Use
the Table Button

To edit the WHERE clause

1. Select `=?` and change it to the desired match condition.

For example, the **Where** text box in [Figure 8-7 on page 8-11](#) contains the following text:

```
customer.zipcode =? and customer.customer_num =?
```

You must change both occurrences of `=?` to valid match conditions. You might change the text as follows:

```
customer.zipcode > 50000 and  
customer.customer_num > 150
```

For a full description of match conditions, see [Appendix D, “Match Condition Operators and Characters.”](#)

2. Check the comparison operators.

When you select multiple columns from the Column Selection window, **ipload** inserts `and` into the expression between each column. You might need to change `and` to `or`.

Editing a Query

To edit a query, follow the same steps as for creating a query, but open an already existing query in the Query window.

To edit a query

1. Choose **Components→Query** from the HPL main window.
The Query window appears, as [Figure 8-2 on page 8-5](#) shows.
2. Click **Open**.
3. Select a query from the list of queries.
4. Click **OK**.

The query-definition window appears, as [Figure 8-3 on page 8-6](#) shows. The name of the query that you are editing appears in the title bar.

5. Modify the **Select**, **From**, and **Where** text boxes.
6. Click **OK** to save the modified query.
7. Click **Cancel** to return to the HPL main window.

Exporting and Importing Queries

You can use the **File** button on the query-definition window to export a query to a file or to import a query that you prepared with some other tool. For example, you might use DB-Access to prepare and test a query and to save the query to a file. You can then import that query into the HPL.

Importing a Query

You can use the Import/Export File Selection window to import a query that you prepared outside of the HPL.

To import a query

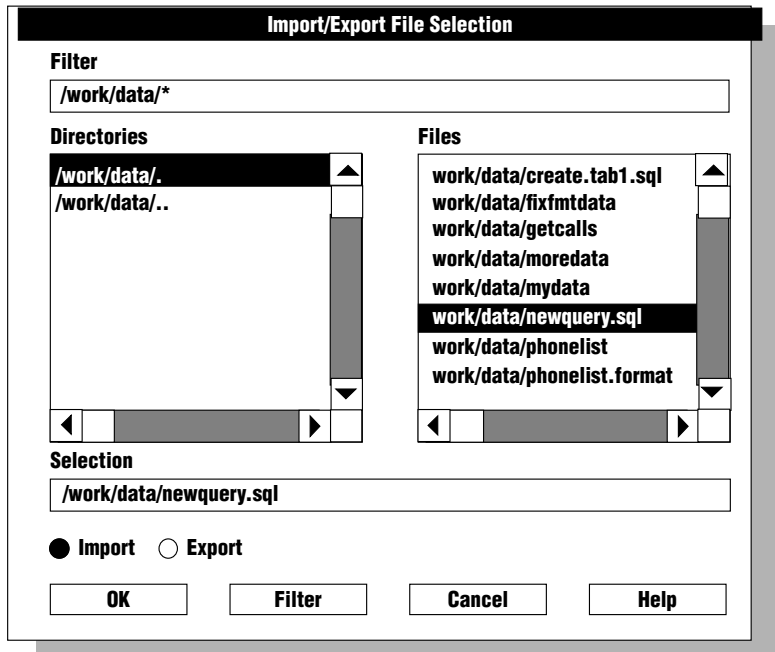
1. Display the query-definition window ([Figure 8-3 on page 8-6](#)) by following the steps in [“Creating a Query” on page 8-4](#).

2. Click **File**.

The Import/Export File Selection window appears, as [Figure 8-8 on page 8-14](#) shows.

Figure 8-8

The Import/Export File Selection Window



3. Click **Import**.

4. Specify the file that you want to import.

You can do this in either of the following ways:

- Type a pathname and appropriate wildcard(s) in the **Filter** text box and click **Filter**. Use an asterisk (*) to list all of the files in the directory. Then select a file and click **OK** or double-click a filename.
- Type the full pathname in the **Selection** text box and then click **OK**.

The text from the imported file appears in the query-definition window.

If **ipload** can interpret the SQL statement, the SQL statement is inserted into the appropriate **Select**, **From**, and **Where** text boxes.

If **ipload** cannot interpret the SQL statement, the entire content of the imported file appears in the **Select** text box.

5. Edit the query so that it meets your needs.
6. Click **OK**.

If the query is a valid SQL query, the display returns to the Query window.

If the query is not a valid SQL query, **ipload** highlights the portion of the query that it cannot interpret and provides an error message.

7. From the Query window, click **Cancel** to return to the HPL main window.

Exporting a Query

The **File** button also allows you to export the query as an SQL statement. You can prepare a query for export in the following ways:

- Create a new query. (See [“Creating a Query” on page 8-4.](#))
- Open an existing query.
- Import an already prepared query and modify it. (See [“Importing a Query” on page 8-13.](#))

To export a file

1. Follow the steps in [“Creating a Query” on page 8-4](#) to prepare a query in the query-definition window (see [Figure 8-3 on page 8-6.](#))

2. Click **File**.

The Import/Export File Selection window appears (see [Figure 8-8 on page 8-14.](#))

3. Click **Export**.

4. Select the directory and file where the query should be stored.

You can do this in either of the following ways:

- Add the name of a new file to a pathname in the **Selection** text box and click **OK**.
- Type a pathname and appropriate wildcard in the **Filter** text box and click **Filter**. Then select a filename.

5. Click **OK**.

If the file that you specified already exists, **ipload** asks if you want to overwrite the existing file, as [Figure 8-9](#) shows.

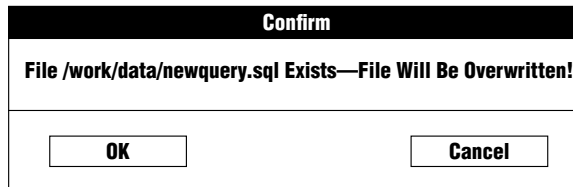


Figure 8-9
The Confirm File Overwrite Window

6. You now have two choices:

- Click **OK** to overwrite. The display returns to the Query window.
- Click **Cancel** to choose a different filename.

The **ipload** utility writes the text from the **Select**, **From**, and **Where** text boxes into the specified file as a single SQL statement.

7. Click **OK**.

The display returns to the Query window.

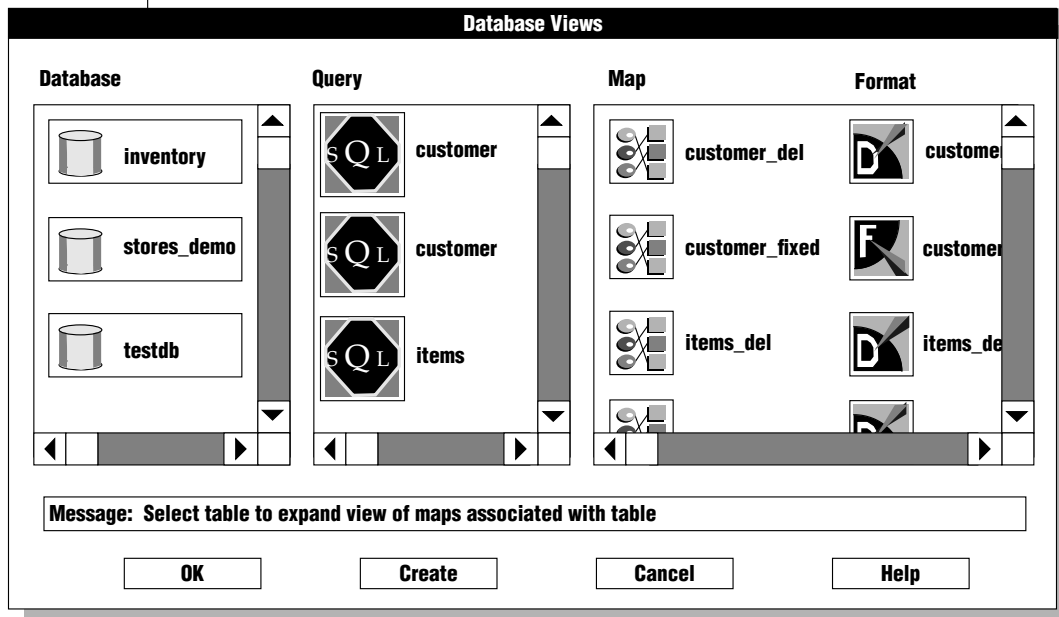
Database Views Window

The Database Views window lets you display a list of the queries, maps, and formats that are associated with a project. The Database Views window also lets you create or edit a query. The Database Views window appears in the following situations:

- When you click **Query** in the Unload Job window and no query name is in the **Query** text box
- When you click **Search** in the Query window

Figure 8-10 shows the Database Views window. “Views Windows” on page 3-14 discusses how to use views windows.

Figure 8-10
The Database Views Window



Defining Maps

In This Chapter	9-3
Load and Unload Maps	9-3
Using the Map-Definition Window	9-4
Using the Table and Format Panes	9-5
Using Unassigned or Multiple-Assigned Fields and Columns	9-5
Using Identical Field Names and Column Names	9-6
Creating a Load Map	9-6
Unload Maps	9-10
Creating an Unload Map	9-10
Unloading Data Using Functions	9-13
Mapping Options	9-14
Using Mapping Options.	9-15
Setting the Mapping Options	9-16
Justification.	9-16
Case Convert	9-16
Default Value	9-17
Transfer Bytes	9-17
Column Offset.	9-17
Field Offset	9-17
Field Minimum and Field Maximum	9-17
Fill Character	9-17
Picture	9-18
Function	9-18

Editing Options	9-18
Using the Delete Button	9-18
Using the Find Button	9-19
Using the Specs Button	9-21
Map Views Window	9-22

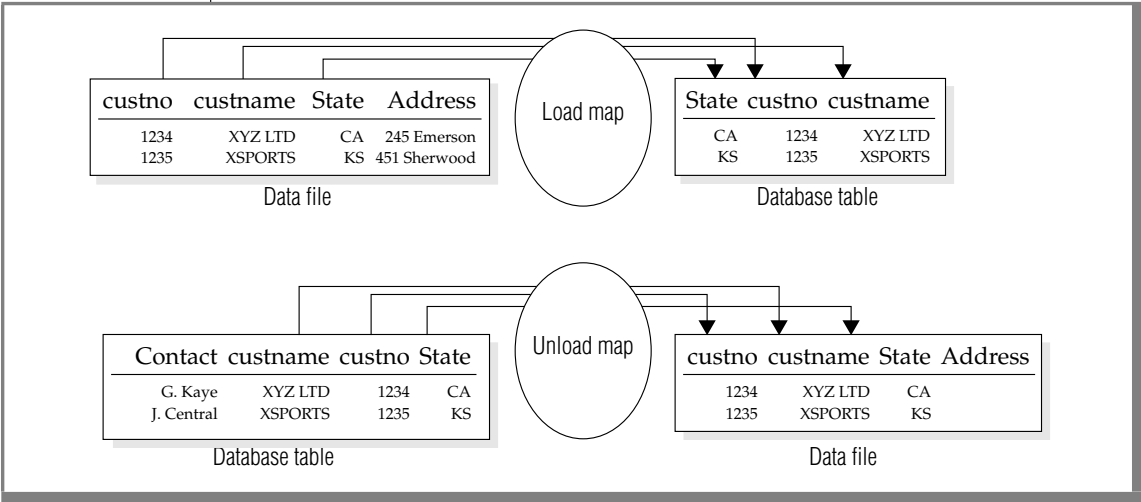
In This Chapter

This chapter describes how to use **ipload** to build a map. A *map* specifies the relationship between the fields of a data file and the columns of a database. To load data into a database, you define a *load map*, which associates fields from records in a data file to columns in a database table. To unload data, you define an *unload map*. The unload map associates the columns that a query retrieves from one or more tables to the fields in a data file.

Load and Unload Maps

Figure 9-1 shows the relationship between load and unload maps.

Figure 9-1
Using a Map



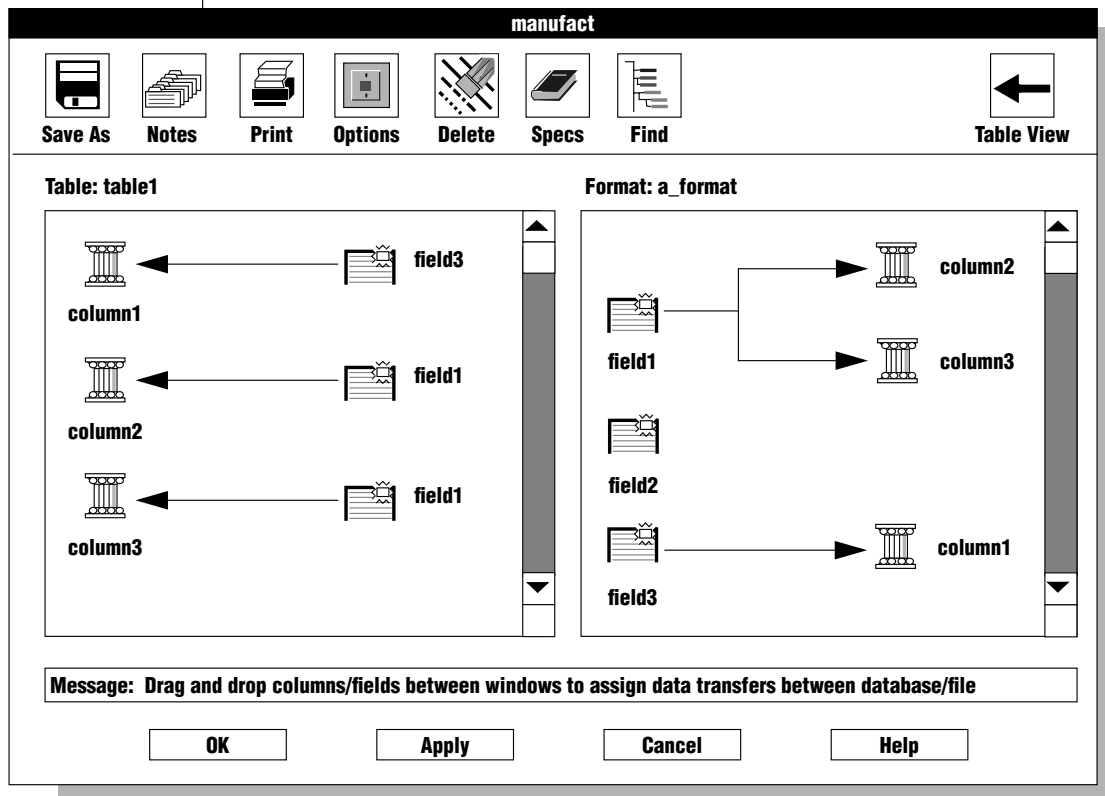
The **ipload** utility stores information about maps in the **maps**, **mapitem**, **mapoption**, and **mapreplace** tables of the **onpload** database. [Appendix A, “onpload Database.”](#) describes these tables.

You can define a map at any time. After you define a map, you use it with the Load Job window or the **onpload** utility.

Using the Map-Definition Window

The map-definition window lets you associate an input item with a table column. [Figure 9-2](#) shows a map-definition window for a load map.

Figure 9-2
The Map-Definition Window



The map specifies which fields of the data file are loaded into database columns. The data moves from the fields of a data file into the columns of a database.

Using the Table and Format Panes

The map-definition window contains two panes: the **Table** pane and the **Format** pane. The window has two panes so that you can take the following actions:

- Scroll the panes to see all of the columns or fields of a long data file or database table
- Connect an input field to more than one column

The left column of icons in each pane represents the active elements of the display. These left columns do not change. In a load map, the columns in the **Table** pane *receive* the input. In the **Format** pane, data from the fields *moves into* the columns of the database table.

The right column of icons in each pane represents the associations that you make. These columns change as you build the map. A field might be listed more than once in the right column of the **Table** pane because you can store a field from the data file in more than one database column. This field is mapped (with a split arrow) to two columns in the **Format** pane. A column never appears more than once in the right-hand list of the **Format** pane because a column can only receive input from one database field.

By scanning the left pane, you can easily see which columns are receiving data from the data file. By scanning the right pane, you can see which fields of the data file are providing data and which fields are not being used.

Using Unassigned or Multiple-Assigned Fields and Columns

The HPL does not require a one-to-one connection between the fields and columns. You can map a field to multiple columns. [Figure 9-6 on page 9-12](#) shows a map where the data from one field is placed into two columns.

You can also have a column that has no mapping association. Field 1 in the **Format** pane in [Figure 9-4 on page 9-9](#) does not have an association. If a column does not receive input, **onpload** sets the column to null.

Using Identical Field Names and Column Names

When you create a format, you can assign arbitrary names to the fields of the data file. You might find it convenient to assign names that correspond to the names of the columns in the database. When you create a map, **ipload** automatically links columns and fields that have the same name and type, thus saving you work.

Creating a Load Map

You can create a load map from the Load Job window or from the **Components** menu of the HPL main window. Before you can create a load map, you must create a format that describes the data file that you plan to load. For information about how to create a format, see [Chapter 7, “Defining Formats.”](#)



Important: *The HPL does not support conversion from extended type data and smart-large-object data (Ext Type data types) to non-Ext Type data types. A field that is defined as an Ext Type data type can be mapped only to an Ext Type column. For more information on Ext Type data types, see “Data Types Allowed in a Fixed Format” on page 7-8 or “Data Types Allowed in a Delimited Format” on page 7-20.*


To create a load map


1. Choose **Components→Maps→Load Map** from the HPL main window.


The Record Maps window appears, as [Figure 9-3](#) shows.


Figure 9-3
The Load Record Maps Window

Load Record Maps

Copy

Delete

Print

Search

Selection Type

☐ Open ☒ Create

Map Data

Map Name:

Database: ▼

Table: ▼

Format: ▼

Current Maps

Notes

Message: Enter a map name to create

OK

Cancel

Help

2. Click **Create** in the **Selection Type** group.
3. Choose a name for the map, and type it in the Map Name text box.
4. Type the names of the database and table where the data will be loaded in their corresponding text boxes.

You can also click the down arrow to choose the names from a selection list. The Tables selection list includes synonyms that are valid for the local database server.
5. Type the format that describes the data file in the **Format** text box.

You can also click the down arrow to choose the format from a selection list.
6. Click **OK** to open the map-definition window.

A map-definition window similar to [Figure 9-4](#) appears.
7. Click a column icon in the left-hand column in the **Table** pane and *hold the mouse button down*. A box appears around the icon and its name.

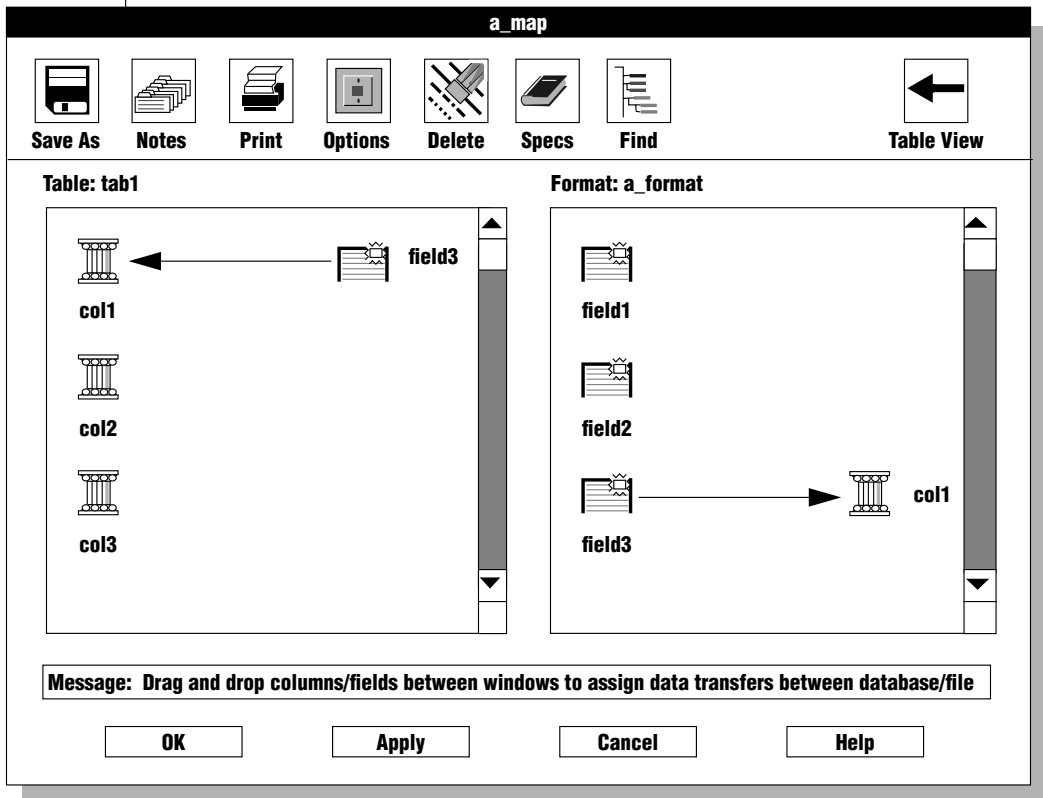
8. Drag the box to a field icon in the **Format** pane.

When you connect columns to fields, it does not matter whether you drag a column to a field or drag a field to a column, but you must always connect items from the left-hand column of each pane.

Figure 9-4 shows a map-definition window with this step completed.

Figure 9-4

Map-Definition Window, One Association Completed



9. Repeat steps 7 and 8 for each field that you want to transfer into the database.
10. Add desired options, if any.
For instructions, see [“Using Mapping Options” on page 9-15](#).
11. Click **OK** to return to the Load Record Maps window.

Unload Maps

An unload map associates columns extracted from a database by a query with the fields in a data-file record. You can create an unload map from the Load Job window or from the **Components** menu of the HPL main window. After you define an unload map, you use it with the Unload Job window or the **onpload** utility.

Creating an Unload Map

Before you can create an unload map, you must define a query on the table that will be unloaded. For instructions on how to define a query, see [“Queries” on page 8-4](#).



Important: The HPL does not support conversion from extended type data and smart-large-object data (Ext Type data types) to non-Ext Type data types. An Ext Type column can be mapped only to an Ext Type field. For more information on Ext Type data types, see [“Data Types Allowed in a Fixed Format” on page 7-8](#) or [“Data Types Allowed in a Delimited Format” on page 7-20](#).

To create an unload map

1. Choose **Components→Maps→Unload Map** from the HPL main window.

The Unload Record Maps window appears, as [Figure 9-5](#) shows.

Figure 9-5

The Unload Record Maps Window

Unload Record Maps

Copy Delete Print Search

Selection Type

☐ Open ☒ Create

Map Data

Map Name:

Database:

Query:

Format:

Current Maps

Notes

Message: Enter a map name to create

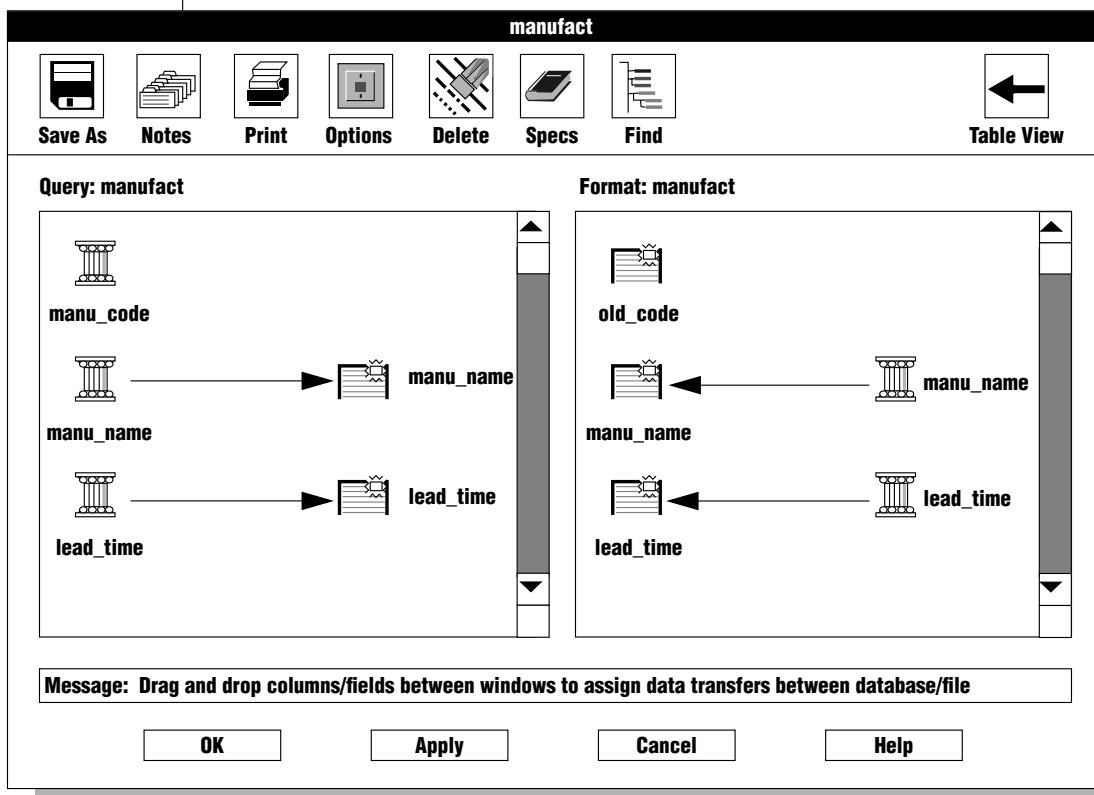
OK Cancel Help

2. Click **Create** in the **Selection Type** group.
3. Choose a name for the map and type the name in the **Map Name** text box.
4. Type the name of the database in the **Database** text box.
You can click the down arrow to choose a database from a selection list of databases.

5. Type the name of a query in the **Query** text box.
You can click the down arrow to choose a query from a selection list of queries.
6. Type the format name in the **Format** text box.
You can click the down arrow to choose a format from a selection list of formats.
7. Click **OK**.

A map-definition window similar to [Figure 9-6](#) appears. In this figure, some of the field names match column names. The **ipload** utility automatically maps columns to fields of the same name. The direction of the arrows indicates the flow of data, as shown in [Figure 9-6](#).

Figure 9-6
The Map-Definition Window



8. To map a database column to a data-file field, click the database-column icon. Drag the column to the desired data-file field icon.
An arrow links the column icon to the field icon.
9. Repeat step 8 until you have mapped all desired columns to fields.
10. Define any mapping options as appropriate.
For information about mapping options, see [“Mapping Options.”](#)
11. Click **OK** to save the map and return to the Unload Record Maps window.
12. Click **Cancel** to return to the HPL main window.

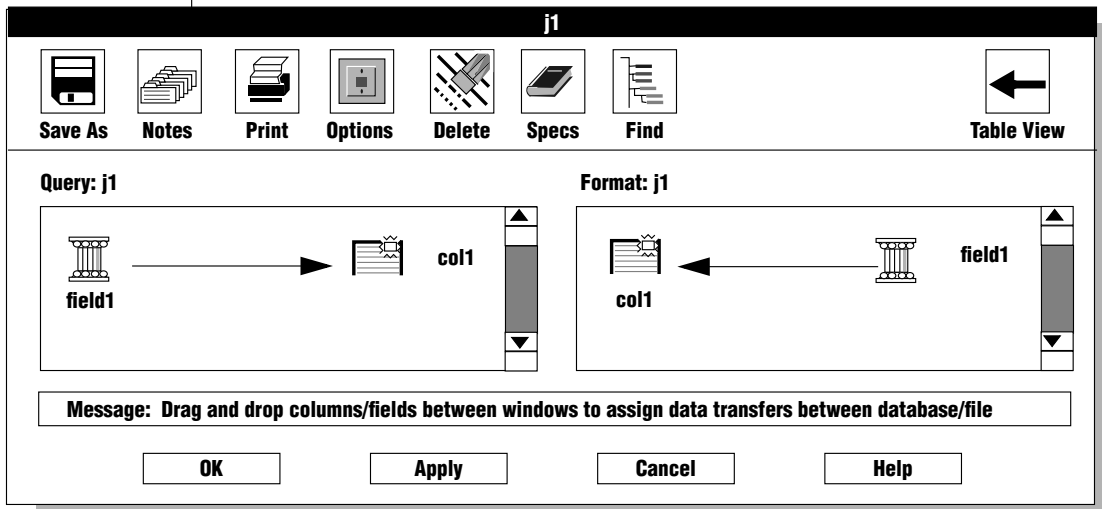
Unloading Data Using Functions

If you use a function in a query to unload data, you must associate a name with the result of that function. In the following example, the returned value of the function TRIM is assigned the name field1.

```
SELECT TRIM(col1) field1 FROM tab1
```

After submitting the query, you must attach field1 to col1 of the unload file manually, as [Figure 9-7](#) shows.

Figure 9-7
The Map-Definition Window



Mapping Options

The mapping options define conversions that **onpload** applies to the data before it inserts the data into the database (for a load job) or into the data file (for an unload job). These conversions can include case conversion, text justification, data masking through picture strings, default values, and fill characters. The mapping options also allow you to replace imported data with data from other database tables.

The information from the Mapping Options window is stored in the **mapoption** table of the **onpload** database. (For more information about the **mapoption** table, see page A-11.)

Using Mapping Options

This procedure describes how to specify mapping options.

To define mapping options

1. Display the map-definition window by following the steps for [“Creating a Load Map” on page 9-6](#), or [“Creating an Unload Map” on page 9-10](#).
2. Select the field or column (in the right-hand column of a pane) that you want to modify.
3. Click **Options**.

The Mapping Options window appears, as [Figure 9-8](#) shows.

Figure 9-8
The Mapping Options Window

Mapping Options	
Field Name: old_code	Column Name: manu_code
Justification: None ▼	Default Value: <input type="text"/>
Case Convert: None ▼	Transfer Bytes: <input type="text"/>
	Column Offset: <input type="text" value="0"/>
	Field Offset: <input type="text"/>
	Field Minimum: <input type="text"/>
	Field Maximum: <input type="text"/>
	Fill Character: <input type="text"/>
	Picture: <input type="text"/> ▼
	Function: <input type="text"/>
Message: All parameters displayed here are optional	
<input type="button" value="OK"/>	<input type="button" value="Cancel"/> <input type="button" value="Help"/>

4. Change the desired options.
5. When you have set all the desired options, click **OK** to return to the map-definition window.

When you return to the map window, an *options symbol* (a small box) appears between the field and the column, as [Figure 9-9](#) shows. The options symbol indicates that mapping options are in effect.

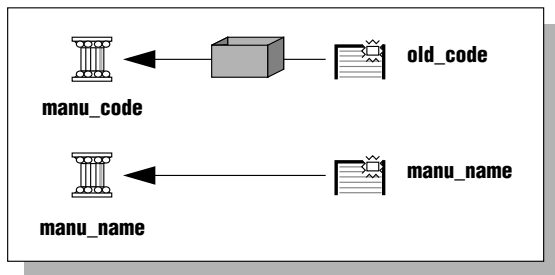


Figure 9-9
Fragment of the Map-Definition Window Showing an Options Symbol

Setting the Mapping Options

You can set as many of the choices on the Mapping Options window as you need.

Justification

The **Justification** option positions text within a record. You can justify the text to the left or right, or you can center it.

Case Convert

The **Case Convert** option converts the case of the data to the selected case. The HPL supports upper, lower, and proper-name conversions. For example, you can make the following conversions.

Input	Conversion Type	Result
JOHN LEE SMITH	Proper Name	John Lee Smith
john lee smith	Proper Name	John Lee Smith
john lee smith	Upper	JOHN LEE SMITH
JOHN LEE SMITH	Lower	john lee smith

Default Value

The **Default Value** option specifies the value that is inserted into the column when no field is mapped into that column.

Transfer Bytes

The **Transfer Bytes** option specifies the number of bytes in the record field to transfer to the database column.

For variable-length format records, this number reflects the maximum size of the field. The actual number of bytes to transfer is determined by the record or field delimiters.

Column Offset

The **Column Offset** option specifies the offset from the beginning of a column field at which to start transferring the data from the field of the data record. Offsets are zero based.

Field Offset

The **Field Offset** option specifies the offset from the beginning of a record field at which to start transferring data to the column. Offsets are zero based.

Field Minimum and Field Maximum

The **Field Minimum** and **Field Maximum** options specify the smallest and largest acceptable values for a numeric column. If the data in the field is outside that range, the HPL rejects the record. This option is available only for fields with numeric formats, such as integer, short, or float.

Fill Character

The **Fill Character** option lets you specify a character that you use to pad the contents of a field. The fill character can be any character that you can type on the keyboard. You can specify a fill character for fixed ASCII and COBOL loads or unloads. The fill character is filled in as a trailing character.

Picture

The **Picture** option lets you reformat and/or mask data from the field of a record before the data is transferred to the database. [Appendix C, “Picture Strings,”](#) explains picture strings.

Function

The **Function** option specifies a user-defined function that is called for every record that is processed. You must add the function to the dynamically linked library. For information on using custom functions, see the API interface documentation in [Appendix E](#).

Editing Options

This section discusses specialized options in the map-definition window.

Using the Delete Button

The **Delete** button lets you break the association between a column and a field.

To use the Delete button

1. Click an icon in the right column of either of the panes in the map-definition window.
2. Click **Delete** to remove the arrow that connects the item to another item.

Using the Find Button

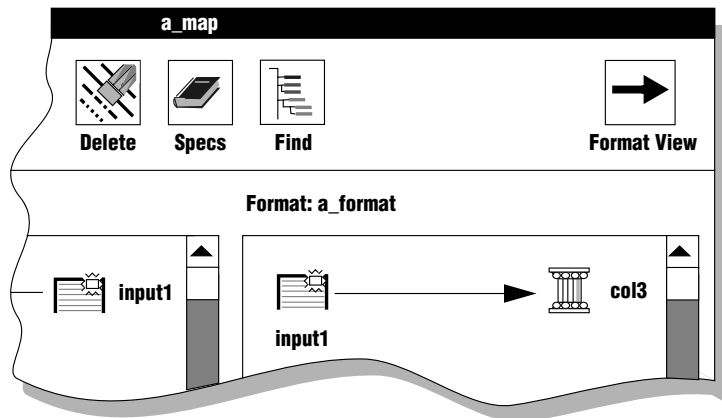
The **Find** button lets you find a column or field in a pane. The **ipload** utility scrolls the selected item into view and puts a box around it. This option is useful when the list of columns or fields is so long that the pane cannot display all of the items.

To use the Find button

1. In the map-definition window, select either the **Table** pane or the **Format** pane.

When you select a pane, the view indicator in the upper-right corner of the window changes to show which pane you selected. [Figure 9-10 on page 9-19](#) shows the upper portion of the map-definition window after you select the **Format** pane.

Figure 9-10
The View Indicator



2. Click **Find**.

The Find Node window appears, as [Figure 9-11](#) shows.

Because the view indicator shows Format View, the Find Node window lists the fields of the data file. To see the columns of the database table, make sure that the view indicator shows Table View.

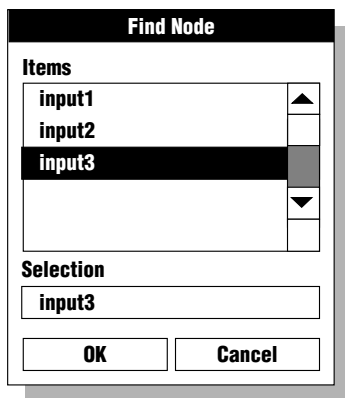


Figure 9-11
The Find Node Window

3. To select the item to find, you can use either of these methods:
- Scroll through the list box to locate the item that you want to find and then select the item.
 - Type the name of the item that you want to find in the **Selection** text box.
4. Click **OK**.

The map-definition window appears again. The selected field or column is highlighted with a box.

Using the Specs Button

The **Specs** button lets you display the Specifications window, which lets you examine the characteristics of the columns and fields in your map.

[Figure 9-12](#) shows a sample Specifications window.

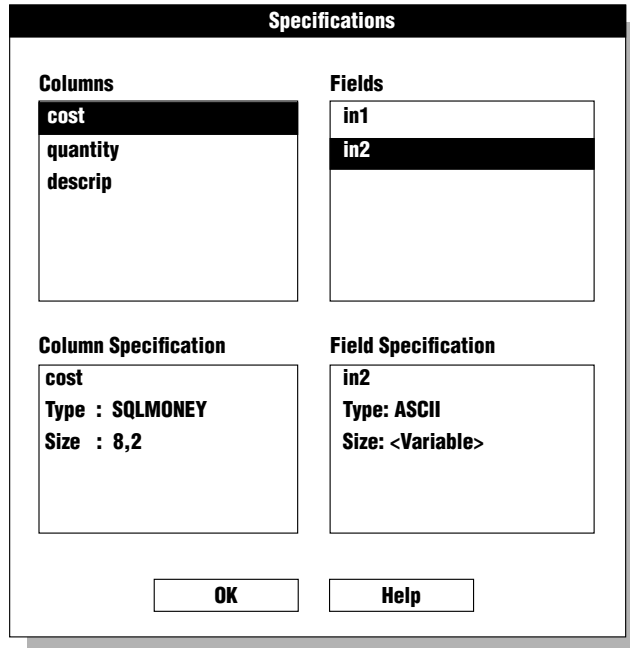


Figure 9-12
The Specifications Window

To use the Specifications window

1. Click **Specs** in the map-definition window.
The Specifications window appears as [Figure 9-12 on page 9-21](#) shows.
2. Select a column from the **Columns** list box or a field from the **Fields** list box or both.
The specification boxes in the lower part of the screen display the characteristics of the selected items.
3. When you finish examining the specifications, click **OK** to return to the map-definition window.

The Specifications window displays the attributes of columns and fields. The Specifications window does not allow you to edit the attributes it displays. To change the attributes of a field, you must modify the format of the data file. (See [“Format Options” on page 7-28.](#)) To change the attributes of a column, you must use appropriate SQL statements to modify the database table.

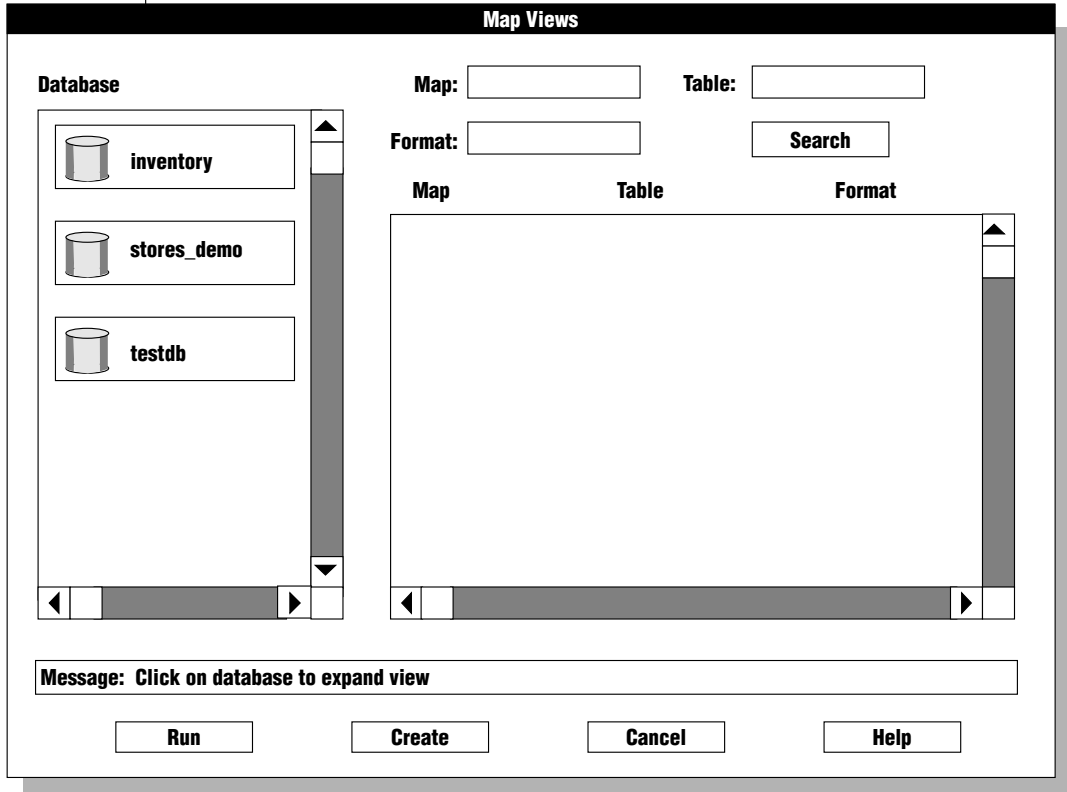
Map Views Window

The Map Views window lets you display a list of the components that are associated with a database in a specific project. The Map Views window also lets you create or edit a map. The Map Views window appears in the following situations:

- If you click **Map** in the Load Job or Unload Job window when no map name is in the **Map** text box
- If you click **Search** in the Load Record Maps or Unload Record Maps window

Figure 9-13 shows the Map Views window for a load map.

Figure 9-13
The Map Views Window for a Load Map



To see the load maps of a database

1. Select a project in the HPL main window.
2. Choose **Components**→**Maps** from the HPL main window.
3. Choose **Maps** →**Load Map** or **Maps**→ **Unload Map**.
4. After the Load Record Maps or Unload Record Maps window appears, click **Search**.

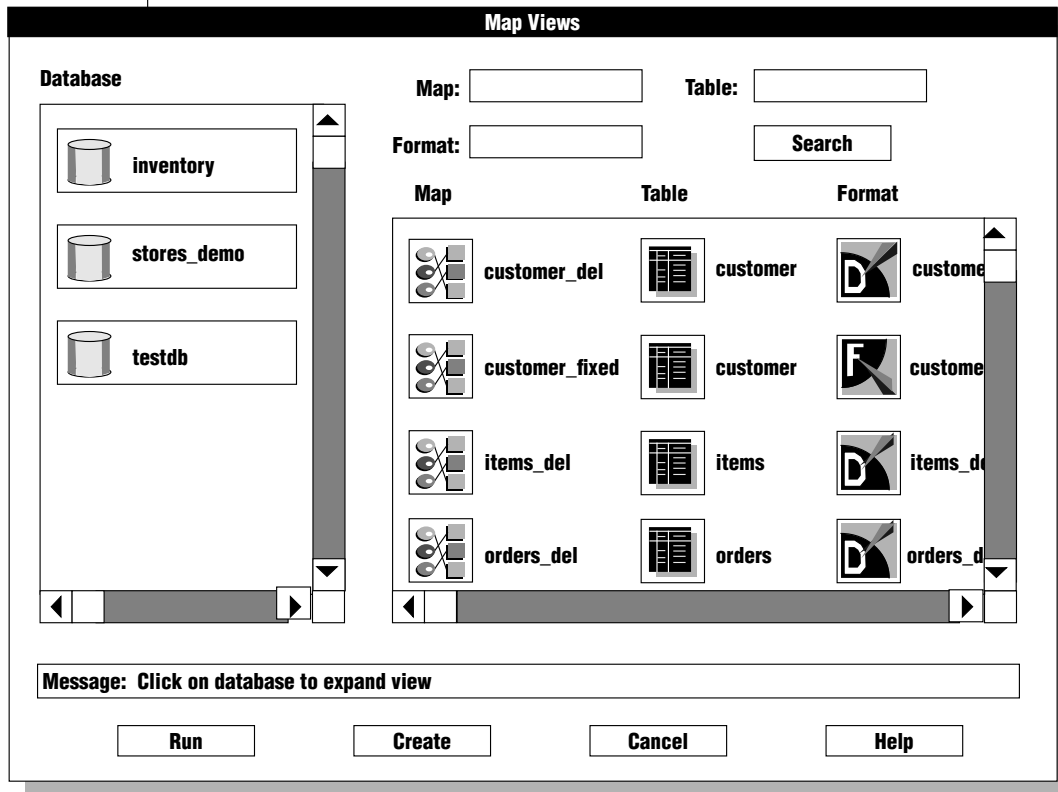
The Map Views window appears, as Figure 9-13 shows.

5. Select a database.

The **ipload** utility displays a list of the maps associated with that database, as [Figure 9-14](#) shows. The **Table** and **Format** columns show the database column and the format associated with each map. If you want to edit a specific map or format, click its button and the corresponding definition window appears.

Figure 9-14

The Map Views Window with the View Expanded









To see selected load maps

1. Open the Map Views window ([Figure 9-13 on page 9-23](#)).
2. Select a database.
3. Type the name or partial name of a map, table, and/or format in the **Map**, **Table**, or **Format** text box.
You can use wildcards in the name.
4. Click **Search**.

[Figure 9-15](#) shows the maps that you find when you search for any table that includes **orders** in its name.

Figure 9-15
The Maps That a Search Found

Map	Table	Format
 orders_del	 orders	 orders_del
 orders_fixed	 orders	 orders_fixed

Defining Filters

In This Chapter	10-3
Using a Filter	10-4
Creating a Filter	10-5
Editing a Filter	10-9
Filter Views	10-11
Filters with Code-Set Conversion	10-12

In This Chapter

Filters are similar to queries. However, queries select data from database tables, whereas *filters* select data from a data file. During the load process, **ipload** loads all of the records from a data file into a database table unless you use a filter to exclude some of the records.

A *filter* is a mechanism for prescreening data-file records for eligibility as database table entries. You can use the filter to include or exclude records explicitly during the load process. You define *match conditions* to filter the records. Match conditions are selection criteria that test one or more data-file fields for certain values or text.

You can define filters at any time. After you define a filter, you can specify it in the Load Job window. The Load Job window is illustrated in [Figure 12-2 on page 12-9](#).

Filters have the following restrictions:

- You cannot use filters with Ext Type data types.
- The DATE and DATETIME data filter can only be applied to the fixed ASCII and delimited format types.

Using a Filter

Suppose that you have a worldwide telemarketing data file that contains the name, country, yearly salary, and age of potential contacts, as the following example shows:

John Brown	US	125,000	57
Mary Smith	Argentina	83,000	43
Larry Little	US	118,000	42
Ann South	Canada	220,000	53
David Peterson	France	175,000	72
Richard North	Spain	350,000	39
Nancy Richards	Japan	150,000	54
William Parker	Egypt	200,000	64

To create a database that includes people who earn over \$100,000 a year, are over the age of 50, and live outside the UNITED STATES

1. Use the match condition **discard salary < 100,000** to exclude people who earn less than \$100,000 a year. The selected records are as follows:

John Brown	US	125,000	57
Larry Little	US	118,000	42
Ann South	Canada	220,000	53
David Peterson	France	175,000	72
Richard North	Spain	350,000	39
Nancy Richards	Japan	150,000	54
William Parker	Egypt	200,000	64

2. Use the match condition **keep age > 50** to include people over the age of 50. The remaining records are as follows:

John Brown	US	125,000	57
Ann South	Canada	220,000	53
David Peterson	France	175,000	72
Nancy Richards	Japan	150,000	54
William Parker	Egypt	200,000	64

3. Use the match condition **discard country = US** to exclude people living in the United States. The remaining records are the records that match all of the restrictions:

Ann South	Canada	220,000	53
David Peterson	France	175,000	72
Nancy Richards	Japan	150,000	54
William Parker	Egypt	200,000	64

If you want to use the same data file to create a database of only those people who live in the United States, or only those people under the age of 30, simply define another filter. There is no limit to the number of filters that you can define for a data file.

Creating a Filter

Before you can create a filter, you must create a format that describes the data file. For information about how to create a format, see [Chapter 7, “Defining Formats.”](#)

The **ipload** utility stores the filter information in the **filters** table of the **onpload** database. For more information about the **filters** table, see [page A-6](#).

To create a filter

1. Choose **Components→Filter** from the HPL main window.
The Filters window appears, as [Figure 10-1](#) shows.

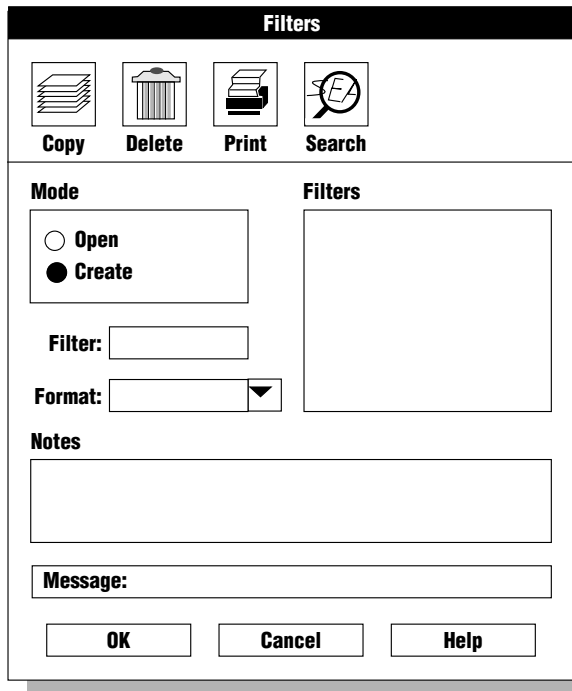


Figure 10-1
The Filters Window

2. Click **Create** in the **Mode** group.
3. Choose a name for the filter and type the name in the **Filter** text box.
4. Type the name of an existing format in the **Format** text box, or click the down arrow and choose a format from the selection list.

5. Click OK.

The filter-definition window appears. Figure 10-2 shows a partially completed filter-definition window.

Figure 10-2
The Filter-Definition Window

The screenshot shows a window titled "newfilter". At the top, there are three icons: a floppy disk labeled "Save As", a printer labeled "Print", and a notepad labeled "Notes". Below these icons, the window is divided into several sections:

- Fields:** A dropdown menu currently showing "country".
- Status:** Two radio buttons labeled "Keep" (unselected) and "Discard" (selected).
- Match Condition:** A text box containing "= "US"" with a vertical scrollbar on the right.
- Filter Items:** A table with two columns: "Filter Items" and "Status".

Filter Items	Status
salary	DISCARD
age	KEEP
- Operation:** A group box containing four radio buttons: "Add" (selected), "Insert", "Edit", and "Delete".
- Perform:** A button located to the right of the "Operation" group box.
- Message:** A text box at the bottom containing the text "Message: Enter field, KEEP/DISCARD status, and match condition".
- Buttons:** At the very bottom are three buttons: "OK", "Cancel", and "Help".

The filter-definition window lets you prepare a filter that specifies which data from the input file should be loaded into the database table.

The filter-definition window has the following parts.

Section	Description
Fields	Specifies the data-file field used in a match condition.
Status	Indicates whether you want to keep or discard records that meet the match condition.
Match Condition	Specifies the criteria for keeping or discarding a record.
Filter Items/Status	Lists existing filter items and their status. As you add match conditions, the conditions are added to this list.

To prepare the filter definition

1. Click **Add** in the **Operation** group to specify that you want to add a new match condition.
2. Type the name of the record field that you want to match in the **Fields** text box.
You can also click the down arrow to see a selection list.
3. Click **Keep** or **Discard** in the **Status** group.
This selection indicates whether the matching record should be entered into the database or discarded.
4. Type the match condition in the **Match Condition** text box using the appropriate logical operators and match characters.
See [Appendix D](#) for a list of the logical operators and match characters.
5. Click **Perform**.
6. Repeat steps 2 through 5 for each additional filter item.
7. Click **OK** to save the filter and return to the Filters window.
8. Click **Cancel** to return to the HPL main window.

Editing a Filter

After you create a filter, you might need to change it.

To edit an existing filter

1. Choose **Components→Filter** from the HPL main window.
The Filters window appears, as [Figure 10-1 on page 10-6](#) shows.
2. Click **Open** in the **Mode** group.
3. Select the filter that you want to modify.
4. Click **OK**.
The filter-definition window appears, as [Figure 10-2 on page 10-7](#) shows.
5. Click **Edit** in the **Operation** group.
6. Select the desired filter item from the list of items.
The field, status, and match conditions appear in their respective areas on the screen.
7. Change the desired information.
8. Click **Perform**.
9. Click **OK** to save your changes and return to the Filters window.
10. Click **Cancel** to return to the HPL main window.

To add an item to the filter

1. Choose **Components→Filter** from the HPL main window.
The Filters window appears.
2. Click **Open** in the **Mode** group.
3. Select the filter that you want to modify.
4. Click **OK**.
The filter-definition window appears.
5. Click **Add** in the **Operation** group.
6. Type the name of the record field in the **Fields** text box.
7. Type the match condition in the **Match Condition** text box.
8. Click **Keep** or **Discard** in the **Status** group to indicate the filter status.

9. Click **Perform**.
10. Click **OK** to save your changes and return to the Filters window.
11. Click **Cancel** to return to the HPL main window.

To insert an item in the filter sequence

1. Choose **Components→Filter** from the HPL main window.
The Filters window appears.
2. Click **Open** in the **Mode** group.
3. Select the filter that you want to modify.
4. Click **OK** to display the filter-definition window.
5. Click **Insert** in the **Operation** group.
6. From the list of items, select the filter item before which you want to insert the new item.
7. Type the name of the record field in the **Fields** text box.
8. Type the match condition in the **Match Condition** text box.
9. Click **Keep** or **Discard** in the **Status** group to indicate the filter status.
10. Click **Perform** to insert the new item before the selected filter item in the **Filter Items** list box.
11. Click **OK** to save your changes and return to the Filters window.
12. Click **Cancel** to return to the HPL main window.

To delete a filter

1. Choose **Components→Filter** from the HPL main window to display the Filters window.
2. Click **Open** in the **Mode** group.
3. Select the filter that you want to edit.
4. Click **OK** to display the filter-definition window.
5. Click **Delete** in the **Operation** group.
6. Select the item that you want to delete from the list of filter items.
7. Click **Perform**.
8. Click **OK** to save your changes and return to the Filters window.
9. Click **Cancel** to return to the HPL main window.

Filter Views

The Filter Views window lets you display a list of the filters and formats that are associated with a project. The Filter Views window also lets you create or edit a filter. The Filter Views window appears in the following situations:

- If you click **Filter** in the Load Job window when no filter name is in the **Filter** text box
- If you click **Search** in the Filters window

Figure 10-3 shows the Filter Views window. “Views Windows” on page 3-14 discusses the use of Views windows.

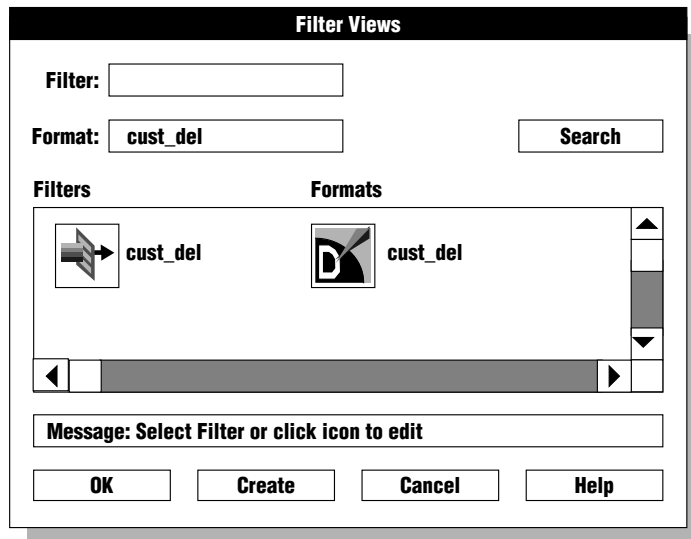


Figure 10-3
The Filter Views
Window

Filters with Code-Set Conversion

When you use a filter to select or discard data during the load, the HPL interprets the filter specification in the code set of the database server. The filtering process on data that undergoes code-set conversion occurs in the following order:

1. The **onpload** utility converts the input data to the code set of the database server.
2. The **onpload** utility performs the filtering operation.

If the code-set conversion process creates lossy errors, then the output of the filter operation can be unexpected. For information on lossy errors and how to define or evaluate a code-set conversion specification, see the *IBM Informix GLS User's Guide*.

Unloading Data from a Database

In This Chapter	11-3
Components of the Unload Job.	11-3
Choosing the Database Server.	11-4
Running Multiple Jobs	11-4
Unload Job Windows	11-5
Creating an Unload Job	11-5
Running the Unload Job.	11-8
Problems During an Unload Job	11-10
Specifying to Write to the End of the Tape	11-10
Using the Command-Line Information	11-11
Changing the Unload Options	11-12
Editing an Unload Job	11-14
Generate Options	11-15

In This Chapter

An *unload job* converts Informix database records to a specified format and then unloads those records to a file, tape, pipe (UNIX only), or device array. You can execute an unload job from the Unload Job window of **ipload**, or you can execute the **onpload** utility from the command line.

This chapter describes the Unload Job window. For instructions on using the **onpload** command-line utility, see [Chapter 16, “The onpload Utility.”](#)

Components of the Unload Job

Before you can unload data, you must first define the following components of the unload job:

- The device array that receives the unloaded data
See [Chapter 6, “Defining Device Arrays.”](#)
- The format that describes the organization of the data file into which you are unloading data
See [Chapter 7, “Defining Formats.”](#)
- The query that extracts the desired records from the database
See [Chapter 8, “Defining Queries.”](#)
- The unload map that describes the relationship between the columns of a database table and the fields of the data-file record
The map also specifies any necessary data translations, such as case conversion and justification. See [Chapter 9, “Defining Maps.”](#)

You can define these components in the following ways:

- Define each component from the Unload Job window.
- Define each component individually from the **Components** menu.
- Use the **Generate Job** option from the **Components** menu.
- Use the **Generate** button in the Unload Job window.

For information about how to use the generate options, see [Chapter 13, “Generate Options.”](#)

Choosing the Database Server

You must run the unload job on the *target server*. The target server is the database server that contains the database from which you unload the data. The database must be on the same database server as **onpload** that extracts data from it. You can run **ipload** on any database server on your network.

Running Multiple Jobs

You can run multiple unload jobs concurrently. However, because the HPL is designed to use as many system resources as possible, running concurrent jobs might overload the system. If you are using a UNIX **cron** job to run the load and unload jobs, let one job finish before you start the next.

The Unload Job window displays the target and **onpload** database servers in the upper right corner of the display.

Unload Job Windows

The Unload Job Select window lets you create a new unload job or select an existing job for editing.

The Unload Job window lets you create or modify the components of an unload job and run the unload job. You can change unload options before you run the unload job. The unload options include the isolation level and the maximum number of errors to permit before **onpload** aborts the unload job.

The **ipload** utility stores the information about the unload job in the **session** table of the **onpload** database. The **session** table draws information from other **onpload** tables, such as **maps**, **formats**, and so on. For more information about the tables of the **onpload** database, see [Appendix A](#).

Creating an Unload Job

Use the Unload Job Select and Unload Job windows to create a new unload job.

To create an unload job

1. Choose **Jobs→Unload** from the HPL main window.

The Unload Job Select window appears, as [Figure 11-1](#) shows.

Figure 11-1
The Unload Job Select Window

Unload Job Select

Delete **Notes** **Connect**

Selection Type

☐ Open ☒ Create

Job Name:

Command Line:

☐ Write/read to/from tape until end of device.

Job Information

Job	Type	Status	Server	Map	Datasource

Notes

Message: Select job(s) to open, delete, copy, or print

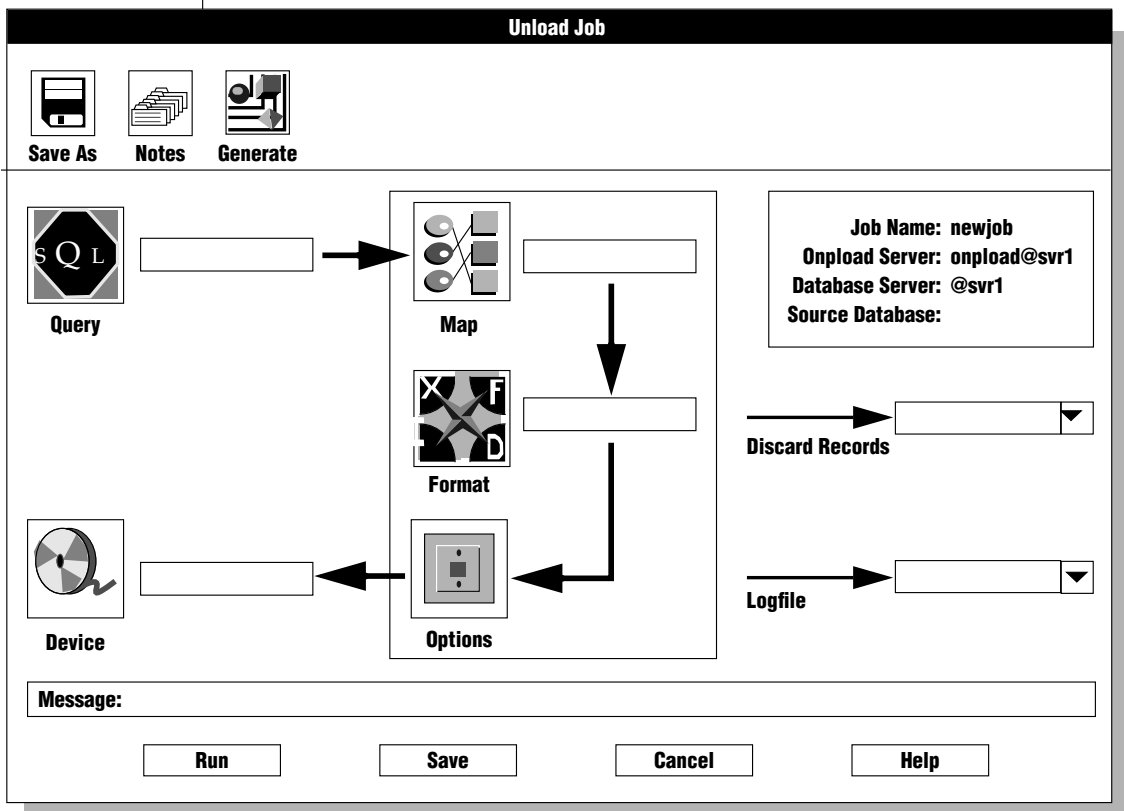
OK **Cancel** **Help**

2. Click **Create** in the **Selection Type** group.
3. Choose a name for this unload job and type the name in the **Job Name** text box.

4. Optionally check the **Write/read to/from tape until end of device** check box. For more information, see [“Specifying to Write to the End of the Tape” on page 11-10](#).
5. Click **OK**.

The Unload Job window appears, as [Figure 11-2](#) shows. [Chapter 3, “Using the High-Performance Loader Windows,”](#) provides detailed descriptions of the buttons in the Unload Job window.

Figure 11-2
The Unload Job Window



6. Type appropriate values for all of the unload components. If you click a component button, the corresponding view window opens, and you can create or select the component.

7. Specify the file that contains rejected records.

Use one of these methods:

- Type the name of the rejected file in the **Discards Records** text box.
- Click the down arrow next to the **Discard Records** text box to select the filename from the file-selection list.

8. Select the file that contains the unload status log.

Use one of these methods:

- Type the name of the log file in the **Logfile** text box.
- Click the down arrow next to the **Logfile** text box to select a filename from the file-selection window.

9. Click **Options** to change unload options.

For more information, see [“Changing the Unload Options” on page 11-12](#).

10. Click **Save** to save this unload job. (If you click **Run** to run the job immediately, the job is saved automatically.)
11. Now you can either run the unload job or exit and run the job later.
 - Click **Run** to run the job.
 - Click **Cancel** to exit to the Unload Job Select window.

Important: Use *Ext Type String Length* data type or *Ext Type Binary Length* data type if you unload data that contains null UDT values.



Running the Unload Job

If you click **Run** in the Unload Job window, the Active Job window appears, as [Figure 11-3](#) shows. The Active Job window displays the progress of your job and indicates when the job completes. When the Active Job window indicates that the job is complete, click **OK** to return to the Unload Job Select window.

The information that **onpload** displays in the Active Job window is also stored in the log file whose name you selected in step 8. For information on how to review the log files, see the [“Viewing the Status of a Load Job or Unload Job”](#) on page 14-9.

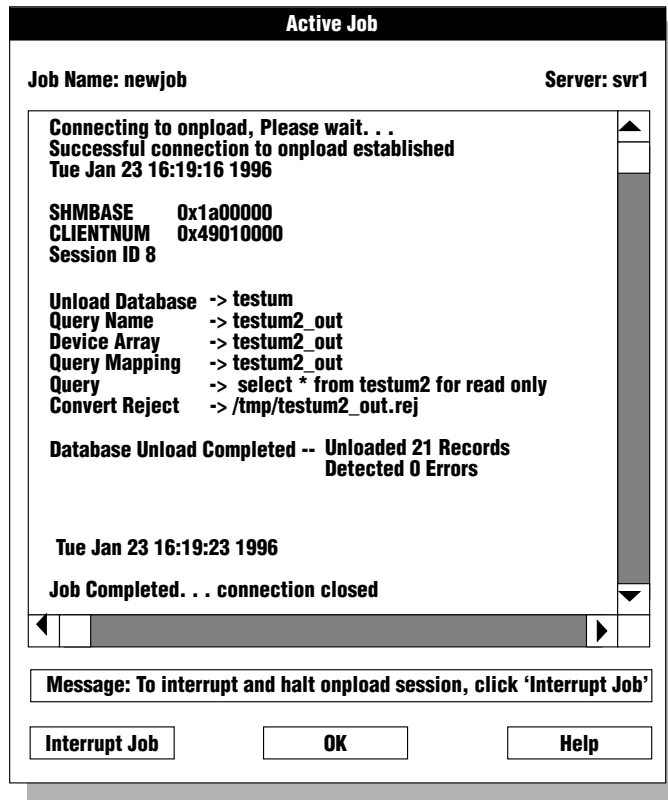


Figure 11-3
The Active Job Window

Problems During an Unload Job

If you encounter any problems during the unload, examine the various files that **onpload** creates. For information on how to review these files, see [Chapter 14, “Browsing.”](#)



Important: *If a write to a file fails because a disk is out of space, the operating system does not return information on how much of the write succeeded. In this situation, the **onpload** utility cannot accurately report the number of records that were actually written to disk. Thus, the number of records that are logged as unloaded in the log file is imprecise.*

Specifying to Write to the End of the Tape

On the Unload Job Select window, you can specify to write to the tape until the end of the device with the **Write/read to/from tape until end of device** checkbox. When a tape device is full, you are prompted to provide the next tape, until the unload job is complete.



Important: *If you select this option, you must also select it when loading from the Load Job window, or specify the **-Z** option from the command line; otherwise, the loaded and unloaded data might be inconsistent.*

If you check this checkbox, it supersedes any tape size information you enter in the device-array Definition window or at the command line.



Important: *You must provide the same tapes in the same order on the same devices for both the unload and the load jobs to ensure consistency.*

Using the Command-Line Information

If you select an existing job in the Unload Job Select window, the **Command Line** text box shows the **onpload** command that **ipload** generated for that unload job. [Figure 11-4](#) shows the **Command Line** portion of an Unload Job Select window. The **Command Line** text box displays the **onpload** command generated for the job shown in [Figure 11-3 on page 11-9](#).

Figure 11-4
Fragment of the Unload Job Select Window

The command line, **onpload -p testum -j testum2_out -fu**, contains the following elements.

Argument	Description
-p testum	The project where the job is stored
-j testum2_out	The name of the job
-fu	The job that unloads (rather than loads) data

You can copy the **onpload** command from the **Command Line** text box and paste it at a system prompt to run the unload job. If you need to run the unload job multiple times (for example, every evening at 5:00 P.M.), you can save the **onpload** command and execute it later.

You do not need to start **ipload** to run a job from the system prompt. Both **ipload** and **onpload** use the **onpload** database, but each one uses it independently.

Changing the Unload Options

The Unload Options window contains the following options.

Option	Description
Isolation Level	<p>The criteria for how the query selects records. The four levels of isolation (from highest to lowest) are as follows:</p> <ul style="list-style-type: none">■ Committed■ Cursor Stability■ Repeatable Read■ Dirty Read <p>The higher the isolation level, the lower the unload performance. For a more detailed definition of isolation levels, see the <i>IBM Informix Guide to SQL: Syntax</i>.</p>
Max Errors	<p>The maximum number of error conditions to be encountered. If the number of unload errors exceeds this number, the unload job stops.</p>

To change unload options

1. Display the Unload Job window.
See the instructions for [“Creating an Unload Job” on page 11-5](#).
2. Click **Options**.
The Unload Options window appears, as [Figure 11-5](#) shows.

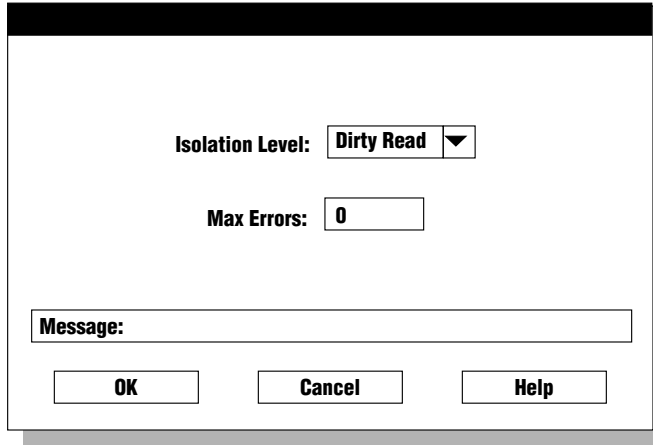


Figure 11-5
*The Unload
Options
Window*

3. Change the desired options.
4. Click **OK** to return to the Unload Job window.

Editing an Unload Job

After you save an unload job, you can return to the unload job and modify it.

To edit an unload job

1. Choose **Jobs→Unload** from the HPL main window.
2. Click **Open** in the **Selection Type** group.
3. Select a job from the **Job Information** list box.
4. Click **OK**.

The Unload Job window appears, as [Figure 11-2 on page 11-7](#) shows.

5. Make appropriate changes to the entries in the Unload Job window.
6. Click **Options** to change unload options.

For more information, see [“Changing the Unload Options” on page 11-12](#).

7. Click **Save** to save this unload job.
8. Now you can either run the unload job or exit and run the job later.
 - Click **Run** to run the job.
 - Click **Cancel** to exit.

Generate Options

Instead of individually creating the components that are required on the Unload Job window, you can use the generate options to create an unload job. You can click **Generate** in the Unload Job window, or you can choose **Components→Generate** from the HPL main window. [Chapter 13, “Generate Options,”](#) describes the generate options.

The generate options do not give you as much flexibility as the Unload Job window, but the options let you create the components quickly. In addition, the generate options let you create formats (Binary, Fixed Internal, and No Conversion) that are not available from the format-definition window.

Loading Data to a Database Table

In This Chapter	12-3
Components of the Load Job	12-3
Choosing the Database Server.	12-4
Running Multiple Jobs	12-4
Preparing User Privileges and the Violations Table	12-5
Setting User Constraints	12-5
Managing the Violations and Diagnostics Tables	12-5
Load Job Windows	12-7
Creating a Load Job	12-8
Running the Load Job	12-10
Making a Level-0 Backup	12-11
Problems During a Load Job	12-11
Specifying to Read to the End of the Tape.	12-12
Using the Command-Line Information	12-13
Changing the Load Options	12-14
Editing a Load Job.	12-16
Generate Options	12-16

In This Chapter

A *load job* loads data from a set of one or more files into a single database table. A record format, which defines each field of a record, specifies the layout of the input data. A *load map* specifies how the record fields are mapped to the columns of the target table. During the load process, the **onpload** utility converts data from record field to table column. This chapter describes the load process.

Components of the Load Job

The HPL lets you define the individual components of a data load individually or lets you use the generate option to define the components automatically. The components of the load job specify:

- The device array where the source data files resides
- The format of the data files
- The filter that accepts or rejects source-file records for the load
- The *map* that specifies the relationship between the data-file format and the database table schema

When you run a load job, you select which individual components to use. The collection of the various components for a specific load is called the load *job*. You can assign a name to a load job, save the job, and then retrieve and rerun it as often as you need to. You can modify an existing job or save it under another job name.

You can define as many different load jobs as you need. You can group your load jobs under one or more projects to make the tasks easier to manage.



Choosing the Database Server

You must run the load job on the *target server*. The target server is the database server that contains the database into which you load the data. The target database must be on the same database server as the **onpload** program that updates it.

Tip: The **onpload** database and the **ipload** interface can be on different computers. You can run the **ipload** interface on any computer that can connect to the database server that contains the **onpload** database.

Running Multiple Jobs

You can run only one express-load job at a time on the same table; however, you can run multiple unload jobs concurrently. Because the HPL is designed to maximize the use of system resources, running concurrent jobs might overload the system.

If you are using a UNIX **cron** job to run the load or unload jobs, let one job finish before you start the next. ♦

UNIX

Preparing User Privileges and the Violations Table

You must make sure that the user who runs a load job has sufficient privileges to manage the constraints and the violations table. The following table summarizes the actions that you must take. The following sections discuss these actions in more detail.

Table Status	User Privileges	Action
Owned by user		No further action is required.
Not owned by user	User has DBA privileges on the table.	No further action is required.
Not owned by user	User does not have DBA privileges on the table.	User must have: <ul style="list-style-type: none">■ Resource privileges on database■ Alter privileges on table Owner must start violations table.

For detailed information about user privileges and violations tables, see the *IBM Informix Guide to SQL: Syntax* and the *IBM Informix Guide to SQL: Reference*.

Setting User Constraints

To modify any constraint, index, or trigger, a user must have both Alter privileges on the table and the Resource privilege on the database. The user must also have these privileges to start or stop a violations table. You use the GRANT statement to set these privileges.

Managing the Violations and Diagnostics Tables

You can turn on or off the generation of constraint-violation information. If you turn on the generation of constraint-violation information, **onpload** writes the information to the *violations* and *diagnostics* tables. For more information, see [“Changing the Load Options” on page 12-14](#).

The HPL manages the violations and diagnostics tables in the following manner:

1. Starts the load job.
2. Starts the violations and diagnostics tables if they do not exist already. (If a violations and diagnostics table already exists, the HPL uses that table).

The HPL uses the following SQL statement to start the violations table:

```
START VIOLATIONS TABLE FOR tablename
```

3. Performs the load job.
4. Stops the violations and diagnostics tables if they were started at step 2.

The HPL uses the following SQL statement to stop the violations and diagnostics tables:

```
STOP VIOLATIONS TABLE FOR tablename
```

5. Drops the violations table if the violations table is empty.

The START VIOLATIONS statement creates the violations and diagnostics tables and associates them with the load table. The STOP VIOLATIONS statement dissociates the violations and diagnostics tables from the load table. For more information about the START VIOLATIONS and STOP VIOLATIONS statements, see the *IBM Informix Guide to SQL: Syntax*.

The violations table (*tablename_vio*) and the diagnostics table (*tablename_dia*) are always owned by the owner of the table with which it is associated. The Resource privilege lets a user start and stop a violations table, but it does not let the user drop a table that he or she does not own. Thus, the HPL cannot drop the violations table in step 5 if the user is not the owner.

Failure to drop the violations table does not cause the load job to fail. However, this failure leaves in the database a violations table that is not associated with a table. If the user tries to run the job again, the START VIOLATIONS TABLE statement in step 2 fails because the table *tablename_vio* already exists.

To solve this problem, the owner of the table or the database administrator must explicitly create the violations and diagnostics tables using the `START VIOLATIONS` statement. When the owner creates the violations table, the following actions take place:

- In step 2, the HPL uses the existing violations table.
- In step 4, the HPL does not stop the violations table because the table was not started in step 2.
- In step 5, the HPL does not drop the violations table because the user does not own the table.

After the load job is complete, an active violations table remains in the database. This table might be empty, but it causes no harm. When the user runs the load job a second time, the violations table is available, and the load job succeeds.

Load Job Windows

The Load Job Select ([Figure 12-1 on page 12-8](#)) and Load Job ([Figure 12-2 on page 12-9](#)) windows let you create, save, and execute a load job. The Load Job window visually represents the various components of a load. After you select the components, you can save the load job for future use or execute it immediately.

The **ipload** utility assigns pathnames for the log files that document the load and that capture records that do not pass the specified filter or that do not pass conversion.

When you use **ipload** to create a job, **ipload** stores information for the job in a row in the **session** table ([page A-17](#)) of the **onpload** database. The **ipload** utility stores information about the components of the load job in other tables of the **onpload** database, including **format**, **maps**, **filters**, and so on. When you use the **onpload** command, columns in the **session** table reference the components to assemble the information necessary for the job. These tables are documented in [Appendix A, “onpload Database.”](#)

Creating a Load Job

The Load Job Select window lets you create a new load job or select an existing job to edit.

To create a load job

1. Choose **Jobs→Load** from the HPL main window.

The Load Job Select window appears, as [Figure 12-1](#) shows.

Figure 12-1

The Load Job Select Window

Load Job Select

Delete **Notes** **Connect**

Selection Type

☐ Open ☒ Create

Job Name:

Command Line:

☐ Write/read to/from tape until end of device.

Job Information

Job	Type	Status	Server	Map	Datasource

Notes

Message: Enter a job name to create

OK **Cancel** **Help**

2. Click **Create** in the **Selection Type** group.
3. Select a name for the job and type it in the **Job Name** text box.
4. Optionally check the **Write/read to/from tape until end of device** check box. For more information, see [“Specifying to Read to the End of the Tape” on page 12-12.](#)
5. Click **OK**.

The Load Job window appears, as [Figure 12-2](#) shows.

Figure 12-2
The Load Job Window

The screenshot shows the 'Load Job' window with a central workflow area. On the left, there are icons for 'Device' (a CD) and 'Table' (a database table). Arrows indicate a sequence: Device → Format → Filter → Map → Options → Table. Each step has an associated icon and a text box for input. To the right of the workflow, there is a box for job details: 'Job Name: newjob', 'Onpload Server: onpload@svr1', 'Database Server: @svr1', and 'Target Database:'. Below this, there are two dropdown menus labeled 'Discard Records' and 'Logfile'. At the bottom, there is a 'Message:' text box and four buttons: 'Run', 'Save', 'Cancel', and 'Help'. The top of the window has a title bar 'Load Job' and three icons: 'Save As', 'Notes', and 'Generate'.

6. Type the appropriate values for the components of the load.
[“Icon Buttons” on page 3-28](#) describes the icons that represent the components of the load. For detailed information about these components, see the individual chapters on device arrays, formats, filters, and maps.
7. Select a base name for the files that contain rejected records and type it in the **Discard Records** text box.
[“Reviewing Records That the Conversion Rejected” on page 14-6](#) gives information about rejected records.
8. Choose a name for the file that contains the load job status log and type it in the **Logfile** text box.
For more information about the log file, see [“Viewing the Status of a Load Job or Unload Job” on page 14-9](#).
9. Click **Options** to change the load options.
For more information on these options, see [“Changing the Load Options” on page 12-14](#).
10. Click **Save** to save this unload job. (If you click **Run** to run the job immediately, the job is saved automatically.)
11. Now you can either run the load job or exit and run the job later.
 - Click **Run** to run the job.
 - Click **Cancel** to exit to the Load Job Select window.

Running the Load Job

If you click **Run** in the Load Job window, the Active Job window appears, as [Figure 2-17 on page 2-30](#) shows. The Active Job window displays the progress of your job and indicates when the job completes. When the Active Job window indicates that the load job is complete, click **OK** to return to the Load Job Select window.



Tip: Before you run a load job, you might want to view the data-file records according to a specified format to check your definitions. For more information, see [“Previewing Data-File Records” on page 14-3](#).

After you run an express-mode load, you must make a level-0 backup before you can access the table that you loaded.

Making a Level-0 Backup

Express-mode loads do not log loaded data. After an express-mode load, **onpload** sets the table to read-only as a protective measure. To make the table available for write access, you must do a level-0 backup for all the dbspaces that the fragments of the loaded table occupy. The level-0 backup allows data recovery for the table in case of future database corruption.

If you do not need to provide for data recovery, you can use **/dev/null** as the backup device for the level-0 backup. This strategy makes the table available for write access without actually backing up the data. If a user attempts to write into the table before you make a level-0 backup, the database server issues an error message.

If you run several express-load jobs on *different* tables in a database, you can complete all of the loads before you perform the level-0 backup. However, if you try to do a second load on the same table without making a level-0 backup, the database server issues an error message.

For discussions of table fragments and dbspaces, see the *Administrator's Guide*. For information about making backups, see either the *Archive and Backup Guide for IBM Informix Dynamic Server* or the *IBM Informix Backup and Restore Guide*, depending on the system you use.

Problems During a Load Job

If you encounter any problems during the load, examine the various files that **onpload** creates. For information on how to review these files, see [Chapter 14, "Browsing."](#)



Warning: Because of operating-system limitations, the **onpload** utility cannot load successfully from a file (on disk) that is larger than 2 gigabytes on a platform that is less than a 64-bit platform. If you try to read a file that is larger than 2 gigabytes, **onpload** HPL fails only after it processes the first 2 gigabytes of data. The HPL log file reports the following error:

```
Cannot read file /some_dir/a_long_file - aio error code 27
```

Specifying to Read to the End of the Tape

On the Load Job Select window, you can specify to read from the tape until the end of the device with the **Write/read to/from tape until end of device** checkbox. When a tape device is empty, you are prompted to provide a new tape, until the load job is complete.



Important: If you select this option, you must also select it when unloading from the Unload Job window, or specify the **-Z** option from the command line; otherwise, the loaded and unloaded data might be inconsistent.

If you check this checkbox, it supersedes any tape size information you enter in the device-array Definition window or at the command line.



Important: You must provide the same tapes in the same order on the same devices for both the unload and the load jobs to ensure consistency.

Using the Command-Line Information

If you select an existing job in the Load Job Select window, the **Command Line** text box shows the **onpload** command that **ipload** generated for that load job. [Figure 12-3](#) shows the **Command Line** portion of a Load Job Select window. The **Command Line** text box displays the **onpload** command generated for the load job that [Figure 2-15 on page 2-27](#) shows.

Figure 12-3
Fragment of the Load Job Select Window

The command line, **onpload -p practice -j newjob -fl**, contains the following elements.

Element	Description
-p practice	The project where the job is stored
-j newjob	The name of the job
-fl	The job that loads (rather than unloads) data

You can copy the **onpload** command from the **Command Line** text box and paste it at a system prompt to run the load job. If you need to run the load job multiple times, you can save the **onpload** command and execute it later.

You do not need to start **ipload** to run a job from the system prompt. The **ipload** and **onpload** utilities both use the **onpload** database, but each utility uses it independently.

Changing the Load Options

Before you run a load job, you can review or change any load options. The load options include specifying the number of records to load, the starting record number, and the loading mode.

The **ipload** utility stores option information in the **session** table of the **onpload** database. For more information on the **session** table, see [Appendix A](#).

The Load Options window contains the following option text boxes.

Option	Description
Load Mode	The mode for the load: express, deluxe, or deluxe without replication
Generate Violations Records	Whether or not to generate violations records
Tapes	The number of tapes that contain source data
Number Records	The number of records to process in the data file
Start Record	The record number in the data file from which to start loading
Max Errors	The maximum number of error conditions to be encountered. If the number of load errors exceeds this number, the load stops.
Commit Interval	The number of records to load before logging the transaction. If you set the commit interval to 0, onpload uses the default value of 10. You can use this option only with deluxe mode.

To change load options

1. Display the Load Job window.
See [“Creating a Load Job” on page 12-8](#).

2. Click **Options**.

The Load Options window appears, as [Figure 12-4](#) shows.

Figure 12-4*The Load Options Window*

Load Options

Load Mode: Express ▼
 Deluxe with Replication
 Deluxe without Replication

Generate Violations Records: ☐

Tapes: 0

Number Records: 0

Start Record: 0

Max Errors: 0

Commit Interval: 0

Message: Table is locked during Express mode load

OK Cancel Help

3. Change the desired option(s).
4. Click **OK** to return to the Load Job window.

Editing a Load Job

After you create and save a load job, you can later return and modify that load job.

To edit a load job

1. Choose **Jobs→Load** from the HPL main window.
The Load Job Select window appears, as [Figure 12-1 on page 12-8](#) shows.
2. Click **Open** in the **Selection Type** group.
3. Select a job from the **Job Information** list box.
4. Click **OK**.
The Load Job window appears, as [Figure 12-2 on page 12-9](#) shows.
5. Make appropriate changes to the entries in the Load Job window.
6. Click **Options** to change load options.
7. Click **Save** to save this load job.
8. Run or cancel the load, as follows:
 - Click **Run** to run the data load.
 - Click **Cancel** to exit.

Generate Options

Instead of individually creating the components that are required on the Load Job window, you can use the generate options to create a load job. You can click **Generate** in the Load Job window, or you can choose **Components→Generate Job** from the HPL main window. [Chapter 13, “Generate Options,”](#) describes the use of the generate options.

The generate options do not give you as much flexibility as creating each component individually, but these options let you create the components quickly. (After you generate the components, you can edit the components individually by accessing them through the **Components** menu.) In addition, the generate options let you create formats (Fixed Internal and No Conversion) that are not available from the format-definition window.

Generate Options

In This Chapter	13-3
Types of Generate Tasks	13-3
Generating from the Load Job Window	13-4
Using the Autogenerate Load Components Window	13-4
Generating from the Unload Job Window	13-6
Using the Autogenerate Unload Components Window	13-6
Generating from the Components Menu	13-10
Generate Window	13-10
Generate Group	13-11
Format Type Group	13-11
Generating Load and Unload Components	13-12
Using the No Conversion Job Option	13-14

In This Chapter

The generate options of **ipload** let you automatically generate components of a load or unload job. The generate options can save you time when you create new formats, maps, queries, and load and unload jobs.

When you generate a load or unload job for an Informix database, **ipload** creates a format for the data file and a map that associates the columns of the table with the fields of the data-file records. Although the generated components might not match your database schema or data-file records exactly, the components created by the generate options provide useful starting points for building HPL components. After you generate default components, you can modify the components to match your specific needs.

Types of Generate Tasks

The **ipload** utility lets you perform the following tasks:

- Generate load components from the Load Job window
- Generate unload components from the Unload Job window
- Generate both load and unload components from the **Components** menu

Generating from the Load Job Window

The **Generate** button in the Load Job window lets you save time when the format of the data file corresponds to the format of the database table. When you generate from the Load Job window, **ipload** makes the following assumptions about the file (or device array) that contains the data:

- The file is an ASCII file.
- The file uses the same locale as the database.
- The file uses a vertical bar (|) for the field delimiter and a new line for the record delimiter.
- The fields in each record of the file correspond one-to-one to the columns of the target table.
- All records in the file should be loaded.

Using the Autogenerate Load Components Window

When you generate from the Load Job window, **ipload** creates a format, a map, a job, and, if needed, a device array.

To generate a job from the Load Job window

1. Choose **Jobs→Load** from the HPL main window.
The Load Job Select window appears, as [Figure 12-1 on page 12-8](#) shows.
2. Click **Create** in the **Selection Type** group.
3. Select a name for the load job and type it in the **Job Name** text box.
4. Click **OK**.

The Load Job window appears, as [Figure 12-2 on page 12-9](#) shows.

5. Click **Generate**.

The Autogenerate Load Components window appears, as [Figure 13-1](#) shows.

Figure 13-1
The Autogenerate
Load Components
Window

6. Click **Device Array** or **File** to indicate the location of the source data.

To load from an existing device array, click **Device Array** and type the name of the device array.

To load from a file, click **File** and type the full pathname of the file. The **ipload** utility automatically generates a device array that includes the file.

7. In the **Load To** group, type the name of the database and table that will receive the data.

8. Click **OK** to generate the components of the load and return to the Load Job window.

9. If needed, click **Filter** to prepare a filter.

10. If you want, change the pathnames in the **Discard Records** and **Logfile** text boxes.

11. Click **Save** to save the components and the job.

12. Click **Run** to execute job or **Cancel** to exit.

Generating from the Unload Job Window

The **Generate** button in the Unload Job window lets you save time when the format of the data file is similar to the format of the database table. When you generate from the Unload Job window, **ipload** makes the following assumptions about the file (or device array) into which the data is unloaded:

- The file is an ASCII file.
- The file uses the same locale as the database.
- The file uses a vertical bar (|) for the field delimiter and a new line for the record delimiter.

Using the Autogenerate Unload Components Window

When you generate from the Unload Job window, **ipload** creates a format, a map, a job, and, if needed, a device array. You can generate an unload that uses a query to select from one or more tables or that unloads an entire table.

To generate a job that uses a query

1. Follow the instructions in [“Creating a Query” on page 8-4](#) to create a query.
2. Choose **Jobs→Unload** from the HPL main window.
The Unload Job Select window appears, as [Figure 11-1 on page 11-6](#) shows.
3. Click **Create** in the **Selection Type** group.
4. Select a name for the unload job and type it in the **Job Name** text box.
5. Click **OK**.

The Unload Job window appears, as [Figure 11-2 on page 11-7](#) shows.

6. Click **Generate**.

The Autogenerate Unload Components window appears, as [Figure 13-2](#) shows.

Autogenerate Unload Components

Unload From

☒ **Table** **Database:** stores_demo ▼

☐ **Query** **Table:** customer ▼

Query: ▼

Unload To

☐ **Device Array** ▼

☒ **File** /work/cust_out ▼

Message: Enter file name to unload into

OK Cancel Help

Figure 13-2
The Autogenerate
Unload Components
Window

7. Click **Query** in the **Unload From** group.

8. Enter the name of the query.

You can use the down arrow to see selection lists. When you unload from a table, you do not enter a query.

9. Click **Device Array** or **File** in the **Unload To** group.

If you click **Device Array**, you can use the down arrow to see a list of the available device arrays.

If you click **File**, **ipload** creates a device array of the same name as the unload job and inserts the specified file into that device array.

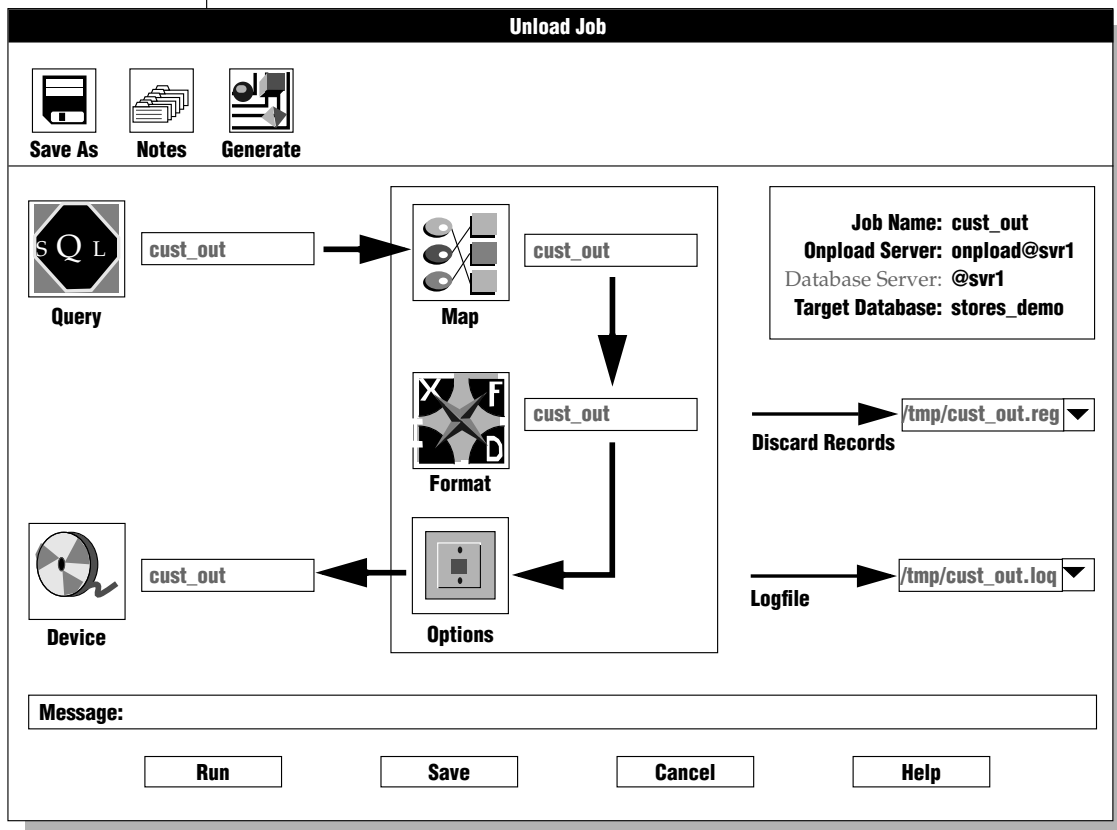
Important: If you are executing **onpload** from the command line during an unload, you must first create the file.



10. Click **OK** to generate the components of the unload.

The display returns to the Unload Job window. The **ipload** utility completes the Unload Job window. If you chose **cust_out** for the unload job name (step 4), the Unload Job window appears as [Figure 13-3](#) shows.

Figure 13-3
The Unload Job Window



11. Click **Save** to save this Unload Job.

You can click **Run** to run the job, or click **Cancel** to exit and run the job later.

12. To run the job, click **Run**.
The Active Job window appears, as [Figure 2-17 on page 2-30](#) shows.
13. When the Active Job window displays `Job Completed`, click **Cancel** to return to the main HPL window.

To generate a job that unloads an entire table

1. Choose **Jobs→Unload** from the HPL main window.
The Unload Job Select window appears, as [Figure 11-1 on page 11-6](#) shows.
2. Click **Create** in the **Selection Type** group.
3. Choose a name for the unload job and type it in the **Job Name** text box.
4. Click **OK**.
The Unload Job window appears, as [Figure 11-2 on page 11-7](#) shows.
5. Click **Generate**.
The Autogenerate Unload Components window appears, as [Figure 13-2 on page 13-7](#) shows.
6. Click **Table** in the **Unload From** group.
7. Enter the desired database and table in the **Database** and **Table** text boxes, respectively.
You can use the down arrows to see selection lists. When you unload from a table, you do not enter a query.
8. Click **Device Array** or **File** in the **Unload To** group.
If you click **Device Array**, you can use the down arrow to see a list of the available device arrays.
If you click **File**, **ipload** creates a device array of the same name as the unload job and inserts the specified file into that device array.
9. Click **OK** to generate the components of the unload.
The Unload Job window reappears, with the components of the job completed.
10. Click **Save** to save this Unload Job.
You can click **Run** to run the job, or click **Cancel** to exit and run the job later.

11. To run the job, click **Run**.

The Active Job window appears, as [Figure 2-17 on page 2-30](#) shows.

12. When the Active Job window displays **Job Completed**, click **Cancel** to return to the main HPL window.

Generating from the Components Menu

To generate all of the components for both load and unload jobs in one operation, choose **Components→Generate Job** from the main HPL window. This Generate option lets you choose formats that are not available from the format-definition window.

Generate Window

The Generate window appears, as [Figure 13-4](#) shows. This window generates all of the components required for a load job and an unload job: format, load map, unload map, query, and device array. The Generate window lets you specify the characteristics of the components that **ipload** creates.

Figure 13-4

The Generate Window

The screenshot shows a window titled "Generate". It contains two columns of radio button options. The first column, labeled "Generate", has three options: "Load/Unload Job" (selected), "No Conversion Job", and "Maps and Formats Only". The second column, labeled "Format Type", has five options: "Delimited" (selected), "Fixed Internal", "Fixed ASCII", "Fixed Binary", "COBOL", and "COBOL (byte)". To the right of these columns are three text input fields with dropdown arrows: "Generate Name:", "Database:", "Table:", and "Device:". At the bottom of the window is a message box that reads "Message: Select options for automatically generating jobs, maps, queries, and formats". Below the message box are three buttons: "OK", "Cancel", and "Help".

Generate	
Generate	Format Type
<input checked="" type="radio"/> Load/Unload Job	<input checked="" type="radio"/> Delimited
<input type="radio"/> No Conversion Job	<input type="radio"/> Fixed Internal
<input type="radio"/> Maps and Formats Only	<input type="radio"/> Fixed ASCII
	<input type="radio"/> Fixed Binary
	<input type="radio"/> COBOL
	<input type="radio"/> COBOL (byte)

Generate Name:

Database:

Table:

Device:

Message: Select options for automatically generating jobs, maps, queries, and formats

OK Cancel Help

Generate Group

The **Generate** group specifies the type of generate to perform. The **Generate** group has the following choices.

Choice	Effect	See Page
Load/Unload Job	Generates both load and unload jobs	13-12
No Conversion Job	Generates a job that treats an entire database record as one entity	13-14
Maps and Formats only	Generates only a format, a load map, and an unload map	

Format Type Group

The **Format Type** group specifies the format of the data file. The **Format Type** group has the following choices.

Choice	Description	See Page
Delimited	The fields of a data-file record are separated by a field delimiter, and records are separated by a record delimiter. The default delimiters are vertical bar () and new line, respectively.	7-30
Fixed Internal	The data file uses Informix internal format. The only changes to the data that you can make when you use this format are ALTER TABLE changes: modify the order of columns, delete or add columns, or change the data type. The HPL loads and unloads data in this format more efficiently than data in the Delimited and Fixed ASCII formats.	7-27
Fixed ASCII	All records are the same length. Each record contains characters in fixed-length fields. This format is the same as the Fixed format choice of the Record Formats window.	7-5

(1 of 2)

Choice	Description	See Page
Fixed Binary	The data-file records contain data in fixed-length fields. Character-oriented data is in character fields. Numeric data (integer, float, and so on) is in machine-dependent binary values. Use this format for loading or unloading data for an application that has or requires data in binary format. Data in binary format is much more compact than data in ASCII format.	7-5
COBOL	The data file is formatted according to COBOL 86 standards. All COBOL data types are supported.	7-24
COBOL (byte)	The data file is formatted using byte alignments for COMP-4 data type. All COBOL data types are supported.	

(2 of 2)



Tip: To generate EBCDIC data, select the *Delimited* or *Fixed ASCII* format and use the format options to change the code set. See *“Format Options”* on page 7-28.

Generating Load and Unload Components

When you choose **Components→Generate**, you can generate all of the components required for a load job and an unload job: format, load map, unload map, query, and device array.

To generate the components for loading or unloading an Informix database

1. Choose **Components→Generate Job** from the main HPL window.
The Generate window appears, as [Figure 13-4 on page 13-10](#) shows.
2. Click **Load/Unload Job** in the **Generate** group.
3. Select a format for the data file in the **Format Type** group.

4. Select a name for the generated components and type it in the **Generate Name** text box.

This name is used for each of the components that this option creates.

5. Type the name of the database that you will load or unload in the **Database** text box or click the down arrow to select a database from the selection list.
6. Type the name of the table within the database in the **Table** text box or click the down arrow to select a table from the selection list.
7. Type the name of a device array in the **Device** text box or click the down arrow to select a device from the selection list.

If you enter the name of a device (file) instead of a device array, **ipload** creates a device array of the same name as the unload job and inserts the specified device into that device array.

When you specify a file instead of a device array in the **Device** text box, **ipload** makes the following assumptions about the data file:

- The file is an ASCII file.
 - The file uses the same locale as the database.
 - The file uses a vertical bar (|) for the field delimiter and a new line for the record delimiter.
 - The fields in the data file are in the same order as the columns of the target table.
8. Click **OK**.

Figure 13-5 shows appropriate choices for generating load and unload jobs for delimited output from the **state** table of the **stores_demo** database. After **ipload** creates the components, you can run the job or use the component-definition windows to make any necessary changes.

Figure 13-5
The Generate Window

Generate

Generate

- ☒ Load/Unload Job
- ☐ No Conversion Job
- ☐ Maps and Formats Only

Format Type

- ☒ Delimited
- ☐ Fixed Internal
- ☐ Fixed ASCII
- ☐ Fixed Binary
- ☐ COBOL
- ☐ COBOL (byte)

Generate Name:

Database:

Table:

Device:

Message: Select options for automatically generating jobs, maps, queries, and formats

OK **Cancel** **Help**

Using the No Conversion Job Option

The **No Conversion Job** option uses the Informix internal format to unload data from a table. Jobs loaded or unloaded with this option are sometimes called *raw loads* or *raw unloads*. The **No Conversion Job** option treats an entire database record as one entity, using the Informix internal format. It does not generate formats or maps. The **No Conversion Job** option is the fastest option that you can use for loading and unloading data. Use this option to transport data or when you need to reorganize the disks on your computer. No-conversion jobs are always completed in express mode. You cannot use the **onpload** command line to convert the running job to deluxe mode when using a no-conversion job.

When you run a job that you created with the **No Conversion Job** option, **ipload** displays a Fast Job Startup window instead of the usual load or unload job window.

To use the Fast Job Startup window

1. Choose **Components→Generate Job** from the main HPL window.
2. Click **No Conversion Job** in the **Generate** group.
3. Select a name for the job and type it in the **Generate Name** text box.
4. Type the names of the database, table, and device array in the **Database**, **Table**, and **Device** text boxes, respectively.
5. Click **OK**.

The display returns to the HPL main window.

6. Choose **Jobs→Load** (or **Jobs→Unload**) from the main HPL window.
The Load Job Select or Unload Job Select window appears.
7. Select the job from the **Job Information** list box.
8. Click **OK**.

The Fast Job Startup window appears, as [Figure 13-6](#) shows.

The screenshot shows a window titled "Fast Job Startup". Inside, the "Job Name" is set to "a_fast_job". Below this, there are three dropdown menus: "Database" with "stores_demo" selected, "Table" with "customer" selected, and "Device Array" with "some_array" selected. At the bottom, there is a message box that says "Message: Enter database to unload". Below the message box are three buttons: "Run", "Cancel", and "Help".

Figure 13-6
*The Fast Job
Startup
Window for a
Load Job*

9. Click **Run** to execute the job.
The Active Job window appears, as [Figure 2-17 on page 2-30](#) shows.
10. When the Active Job window displays **Job Completed**, click **Cancel** to return to the main HPL window.

Browsing

In This Chapter	14-3
Browsing Options	14-3
Previewing Data-File Records	14-3
Using the Record Browser Window	14-3
Reviewing Records That the Conversion Rejected	14-6
Viewing the Violations Table	14-7
Viewing the Status of a Load Job or Unload Job.	14-9
Viewing the Log File	14-9
Sample Log File	14-11

In This Chapter

The browsing options of the HPL let you preview records from the data file, review records after you perform a load or unload, and let you review various files associated with the HPL.

Browsing Options

You can use the browsing options to:

- Preview records from a data file
- Review records that the filter or the conversion reject
- View the violations table
- View the status of a load job or unload job

Previewing Data-File Records

Before you actually execute a load job, you can use the Record Browser window to check your definition of the format. The display shows errors such as incorrect field lengths or missing fields. You can edit the format to correct your format definitions (see [“Editing a Format” on page 7-10](#)).

Using the Record Browser Window

The Record Browser window lets you review records in a specified format, search the list of available formats, or edit a format.

To review data-file records in a selected format

1. In the HPL main window, select the project that contains your load job.
2. Choose **Browsers→Record**.

The Record Browser window appears, as [Figure 14-1](#) shows.

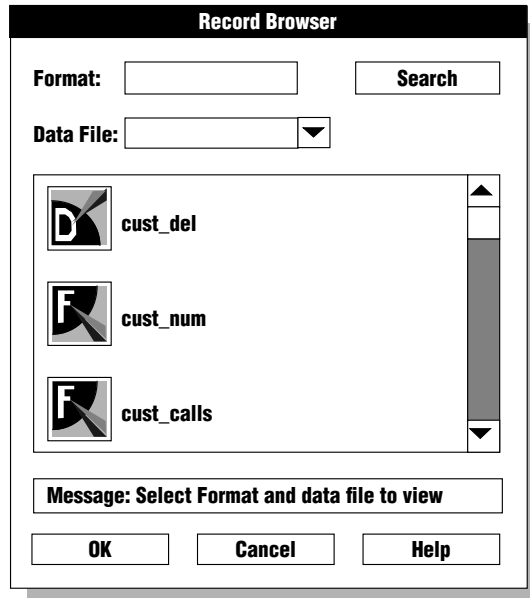


Figure 14-1
The Record Browser Window

3. Type the name of the format to be applied to the source data file or click the format name in the list box.
4. In the **Data File** text box, type the name of the data file that you plan to load, or click the down arrow and select a file from the selection list.

5. Click OK.

The second Record Browser window appears, as [Figure 14-2](#) shows. This Record Browser window displays each of the fields in the format, followed by the value of the field for the given Record Number.

Figure 14-2
The Record Browser Window

Record Browser

File Name: /work/data/mydata
Format: test3

Record Number:

col1:
col2:
col3:

Message:

6. You can take the following actions:
 - Type the record number that you want to view. Click **Select**.
 - Click **Next** to display the next record.
 - Click **Previous** to display the previous record.
7. When you finish browsing, click **Cancel** to return to the HPL main window.

To search for and edit a format

1. In the HPL main window, select the project that contains your load job.
2. Choose **Browsers→Record**.
The Record Browser window appears, as [Figure 14-1 on page 14-4](#) shows.
3. In the **Format** text box, type the format name or partial format name that you want to find.
You can use wildcards (for example, *cust*).
4. Click **Search**.
The **ipload** utility displays all formats of the current project that include the letters *cust*.
5. Click **Cancel** to return to the HPL main window.

To edit a format

1. Select the project that contains your load job.
2. Choose **Browsers→Record** from the HPL main window.
3. Click a format button to edit the format.
The **ipload** utility displays the format-definition window. For information about editing a format, see [“Editing a Format” on page 7-10](#).

Reviewing Records That the Conversion Rejected

When you execute a load job, **onpload** creates a file that contains information about records of the data file that the conversion rejected. This file is named *basename.rej* where *basename* is the base name that you selected in step 7 of [“Creating a Load Job” on page 12-8](#). When you use a generate option to create the components for a load job, *basename* is */tmp/jobname* where *jobname* is the name that you selected for the unload job.

To review rejected records

1. On the HPL main window, select the project that contains your load job.
2. Choose **Browser→Record**.
The Record Browsers window appears, as [Figure 14-1 on page 14-4](#) shows.
3. Type the name of the format to be applied to the rejected-records file in the **Format** text box, or click the format name in the list box.
4. Type the name of the rejected-records file in the **Data File** text box, or click the down arrow and select a data file from the selection list.
5. Click **OK**.
The second Record Browser window appears, as [Figure 14-2 on page 14-5](#) shows.
6. You can take the following actions:
 - Type the record number that you want to view. Click **Select**.
 - Click **Next** to display the next record.
 - Click **Previous** to display the previous record.
7. Click **Cancel** to return to the HPL main window.

Viewing the Violations Table

The load job also creates two tables in the target database that contain information about records that passed the conversion but that the database server rejected. The tables are named *tablename_vio* and *tablename_dia*, where *tablename* is the name of the table being loaded.

Viewing the violations table lets you browse through records that passed the filter and conversion but that the database server rejected. The HPL writes these records into the violations table (*tablename_vio*). The data in the violations table has the same format as the database table.

The *IBM Informix Guide to SQL: Syntax* discusses in detail the information found in the violations table.

To view the violations table

1. Choose **Browsers→Violations** from the HPL main window.
The Violations Table Browser window appears, as [Figure 14-3](#) shows.

Figure 14-3

The Violations Table Browser Window

Violations Table Browser

Database: ▼

Table: ▼

Message: Enter database/table then click 'Select' to view violations records

2. Type the name of the database and table for which you want to review the violations, or click the down arrows to make your choices from selection lists.

3. Click **Select**.

[Figure 14-4](#) shows the first record of a violations table.

Figure 14-4

A Record in the Violations Table Browser Window

The screenshot shows a window titled "Violations Table Browser". At the top, there are two dropdown menus: "Database:" with "stores_demo" selected and "Table:" with "items" selected. To the right of the "Table:" dropdown is a "Select" button. Below these is a large text area containing the following text:

```
informix_tupleid: 1
informix_optype:
  objname: u104_10

order_num: 1
stock_num: 1
manu_code: HR
```

At the bottom of the window, there is a message box that says "Message: Enter database/table then click 'Select' to view violations records". Below the message box are four buttons: "Next", "Previous", "Cancel", and "Help".

4. Click **Next** and **Previous** to move forward and backward through the violations table.
5. Click **Cancel** to return to the HPL main window.

Viewing the Status of a Load Job or Unload Job

When a load or unload job is complete, **onpload** writes a record of the load or unload job into a log file. For information on the messages that the log file can contain, see [Appendix G](#).

Viewing the Log File

The default name for a log file is `/tmp/jobname.log`, where *jobname* is the name that you chose for the job.

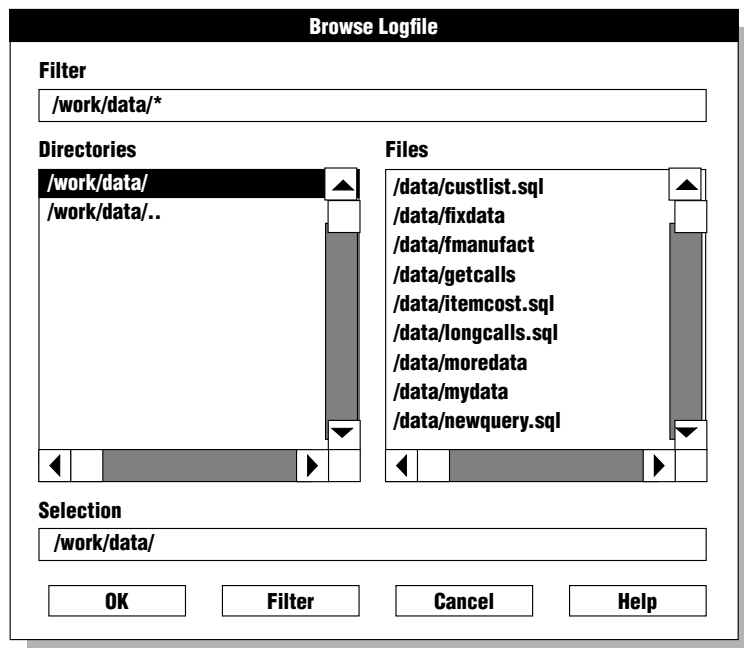
You can specify a different name for the log file in the Load Job window (Figure 12-2 on page 12-9) or Unload Job Window (Figure 11-2 on page 11-7).

To view the log file

1. Choose **Browsers→Logfile** from the HPL main window.

The Browse Logfile window appears, as Figure 14-5 shows. When the window appears, the **Filter** text box and the **Selection** text box show the directory from which **ipload** was started.

Figure 14-5
The Browse Logfile Window



2. In the **Filter** text box, type the full pathname of the directory that contains the log.
You can use wildcards to select only certain files from that directory.
3. Click **Filter**.
The **Files** list box shows a list of the files that match the path that you entered in the **Filter** text box.

4. In the **Files** list box, click the name of the file that you want to examine.
The full pathname of the selected file appears in the **Selection** text box.
5. Click **OK**.
A Browse window appears that displays the contents of the selected file.
6. Review the log using the scroll bar to move through the log.
7. Click **Cancel** to return to the HPL main window.

Alternatively, if you know the full pathname of the log file, you can simply type the pathname in the **Selection** text box and click **OK**.

Sample Log File

The following example shows a sample log file entry:

```
Sat Mar 11 13:52:42 1995

Session ID 1

Unload Database -> stores_demo
Query Name      -> f_manufact
Device Array    -> fmanufact
Query Mapping   -> f_manufact
Query           -> select * from manufact
Convert Reject  -> /work/data/f_manu_unl

Database Unload Completed -- Unloaded 9 Records  Detected 0 Errors

Sat Mar 11 13:52:50 1995
```

You can review the log file to determine load status and to see where any errors occurred. The log file is a simple ASCII file. You can print it if necessary.

Managing the High-Performance Loader

In This Chapter	15-3
Modes	15-3
Deluxe Mode	15-4
Express Mode	15-5
How Express Mode and Deluxe Mode Work.	15-6
Deluxe Mode	15-6
Express Mode	15-6
Foreign-Key Constraints	15-7
Comparison Between Express-Mode and Deluxe-Mode Load Operation.	15-8
Violations	15-9
Rejected Records from the Input File	15-9
Constraint Violations	15-10
Viewing Error Records	15-10
Performance	15-10
Configuration Parameters	15-11
Express-Mode Limitations	15-12
onstat Options for onpload.	15-12
Devices for the Device Array	15-13
Usage Models	15-13
Reorganizing Computer Configuration	15-13
Altering the Schema of a Table	15-14
Loading and Unloading Data	15-14
Settings for a No-Conversion Load or Unload	15-15
Express-Mode Load with Delimited ASCII	15-15
Performance Hints	15-16
Choose an Efficient Format	15-17
Ensure Enough Converter Threads and VPs	15-17

Ensure Enough Memory	15-18
Ensure Enough Buffers of Adequate Size.	15-18
Increase the Commit Interval.	15-19
Limitation When Using the Excalibur Text DataBlade Module Indexes .	15-19

In This Chapter

This chapter discusses the following aspects of managing the HPL:

- Modes
- Violations
- Performance
- Using the Excalibur Text DataBlade module

Modes

The HPL offers two load modes, deluxe mode and express mode, and one unload mode. The express mode is faster, and the deluxe mode is more flexible. [Figure 15-1](#) shows the load and unload modes of the HPL.

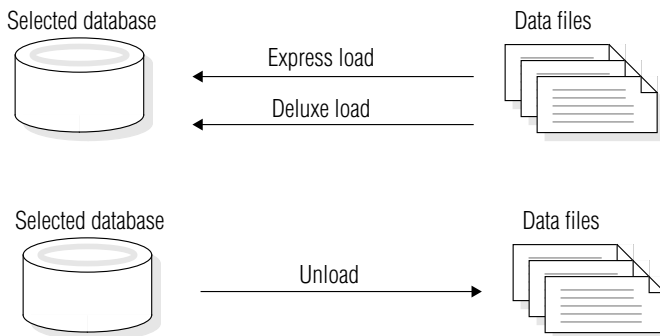


Figure 15-1
*The Load and
Unload Modes of
the HPL*

Deluxe Mode

The deluxe mode has the following features:

- Performs row-by-row referential and constraint checking as the data is loaded
- Allows loading of data while other users are working (no table locking) if the table already has an associated violations table prior to the load, and if either of the following conditions apply:
 - The index is set to *filtering mode without error*
 - The table does not have unique indexes.
- Allows users to access and update the table during a load
Loaded data is immediately visible to the user.
- Logs data, but also offers a no-logging option
- Updates indexes
- Evaluates triggers
- Sets constraints to FILTERING WITHOUT ERROR
- Sets the isolation mode as if for an insert cursor
- Simulates an INSERT statement, except that the HPL allows the load to handle parallel data streams
- Allows loading of data without replication

The deluxe mode has the following limitations:

- Does not support loads without conversion
- Does not support loads that have conversion but do not generate a violations table
- Does not support the **-fv** option

When you use the deluxe mode without replication to load data into the table that was used to define the replicates, the data in that table is not replicated.

Express Mode

The express mode has the following features:

- Locks tables for exclusive use by the load utility
- Disables referential and constraint checking on the table
- Requires a level-0 backup
- Sets all objects (such as indexes or constraints) to disabled before loading
- Reenables all objects after loading, if possible, and flags objects that cannot be reenabled in the violations and diagnostic tables
- Supports loading of raw tables

Express mode has the following limitations:

- Does not support:
 - Logging mode or ANSI mode
 - Tables that contain smart large objects (binary large objects or BLOBs, character large objects or CLOBs)
 - Tables that contain simple large objects (TEXT and BYTE) or extended data types
 - Loading of data on a heterogeneous data replication (HDR) replicated table from a database that has transactions
 - A table that has a fragmentation strategy and has a WITH ROW IDS clause to enable using row IDs with fragmentation
 - Tables with primary key constraints when child table records refer to the load table
 - Rows that are larger than the system page size
For information about page size, see the *Administrator's Guide*.
 - The static-hash access method
- Does not invoke triggers on the loaded data

Important: If your load job has any of these conditions, you must use deluxe mode to load your data.



How Express Mode and Deluxe Mode Work

This discussion describes how HPL implements the deluxe and express modes. You do not need to understand this discussion to use the HPL.

Deluxe Mode

A deluxe-mode load simulates an INSERT statement, except that the HPL allows the load to handle parallel data streams. For more information on the INSERT statement, see the *IBM Informix Guide to SQL: Tutorial* and *IBM Informix Guide to SQL: Syntax*.

Express Mode

The sequence of events when you run an express-mode load is as follows:

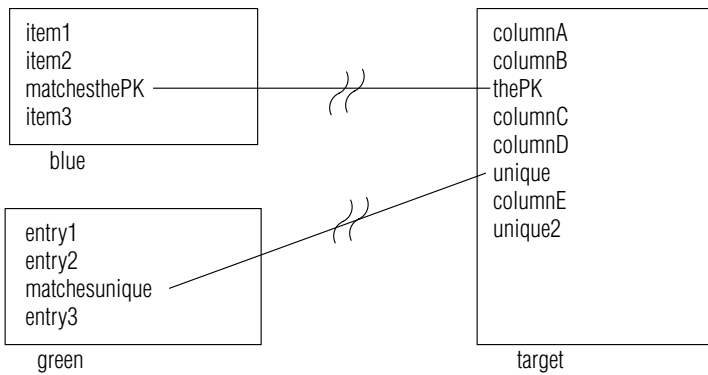
1. The **onpload** utility locks the table with a shared lock.
Other users can read data in already existing rows.
2. The **onpload** utility creates new extents and fills them with the new rows. However, **onpload** does not update the database structures that track extents.
The new extents are not visible to the user.
3. At the *end* of the express load, **onpload** updates the internal structures of the database.
4. The **onpload** utility sets the table to read-only.
This setting occurs because in express mode **onpload** does not log data, and therefore, the table is in an unrecoverable state.
5. The **onpload** utility unlocks the table and enables the constraints.
The new rows become visible to the user for read only.
6. The user performs a level-0 backup.
For more information, see [“Making a Level-0 Backup” on page 12-11](#).
7. Your database server sets the table to read/write.

If the load fails, **onpload** discards the extents and clears the internal information that says the table is unrecoverable.

Foreign-Key Constraints

Express mode cannot disable primary constraints or unique constraints that are referenced as foreign keys that are active on other tables. If you want to load data into such a table, you must first use `SET CONSTRAINTS DISABLED` statements to disable the foreign-key constraints in the referencing table or tables. After the load is finished, reenable the foreign-key constraints.

Figure 15-2 shows an example of foreign-key constraints. The table **target** has a primary key (**thePK**) and a unique key (**unique**) that table **blue** and table **green** reference. Before you perform an express-mode load into the table **target**, you must disable the foreign-key constraints in both table **blue** and table **green**.



Remove the constraints before you use express-mode load.

Figure 15-2
Foreign-Key
Constraints

Comparison Between Express-Mode and Deluxe-Mode Load Operation

The following table contrasts the operation of express mode and deluxe mode. For additional information on the differences between these two modes, see [“Deluxe Mode”](#) and [“Express Mode”](#) on page 15-5.

Express Mode		Deluxe Mode	
Action	Performed by	Action	Performed by
Set objects to disabled	onpload	Set constraints to filtering	onpload
Load records from the data file into the table (including rows that would cause violations if constraints were on)	onpload	Load records from data file into the table. Write records that violate constraints into the violations table <i>and not</i> into the target table	onpload
Enable objects. Detect violators and <i>copy</i> them into the _vio table.	onpload	Set constraints to non-filtering	onpload
Perform a level-0 backup to make the database writable	user		
Resolve violators	user	Resolve violators	user

Violations

When you load records from a data file, some of the records might not meet the criteria that you established for the database table. For example, the data file might contain:

- Null values where the table specifies NOT NULL
- Values in an incorrect format (for example, alphabetic characters in a numeric field)
- Records that do not have the expected number of fields

The way that the HPL treats these errors depends on the mode (deluxe or express) and the type of job (load or unload).

The HPL separates errors into the following two classes:

- Rejected records from the input file
 - These records include:
 - Records that the filter rejected
 - Records that cannot be converted
- Constraint violations

Rejected Records from the Input File

Input-file records that the HPL rejects because they could not be converted include records in an incorrect format, records that do not have the expected number of fields, and records whose fields contain null values for columns that do not allow null values. The **onpload** utility writes these records into a file with the suffix **.rej**. The **onpload** utility writes records that are rejected because they do not match the filter criteria into a file with the suffix **.flt**.

Constraint Violations

When the **onpload** utility starts a deluxe-mode load, it invokes the following SQL statement:

```
SET CONSTRAINTS ON mytable FILTERING
```

This statement has two results:

- The utility adds two tables, **mytable_vio** and **mytable_dia**, to the database that contains **mytable**.
- All of the constraints that are associated with the table are set to filtering. Filtering mode causes any record that does not meet the constraint requirements to be added to the **mytable_vio** table instead of generating an error.

The use of filtering mode for constraints is covered in detail in the *IBM Informix Guide to SQL: Syntax*.

Viewing Error Records

Choose from the **Browsers** menu in the HPL main window to look at the error records that **onpload** generates. For more information on how to use the browser options, see [“Browsing Options” on page 14-3](#).

Performance

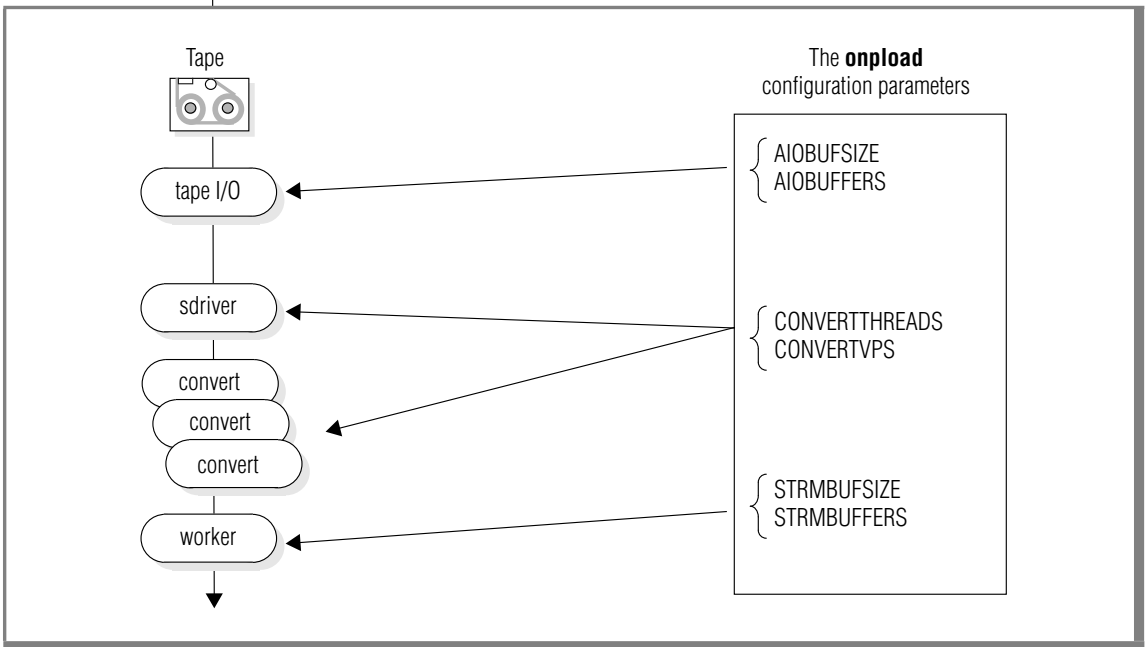
You can improve the HPL performance by preparing an environment that is optimized for the particular load or unload job that you are performing. You should consider the following aspects of your load and unload jobs:

- Configuration-parameter values
- Mode (express or deluxe)
- Devices for the device array
- Usage models

Configuration Parameters

The **onpload** configuration parameters control the number of threads that **onpload** starts and the number and size of the buffers that are used to transfer data. [Figure 15-3](#) shows which part of the **onpload** process is affected by each configuration parameter.

Figure 15-3
The **onpload** Configuration Parameters



The **AIOBUFSIZE** and **AIOBUFFERS** parameters control the number and size of the buffers that **onpload** uses for reading from the input device. **CONVERTTHREADS** and **CONVERTVPS** control the amount of CPU resources to apply to data conversion. **STRMBUFSIZE** and **STRMBUFFERS** control the number and size of the buffers used to transport data between **onpload** and the database server.

The **onpload** configuration parameters are stored in the following files:

- **\$INFORMIXDIR/etc/\$PLCONFIG** ♦
- **%INFORMIXDIR%\etc\%PLCONFIG** ♦

UNIX

Windows

[Appendix B](#) describes each configuration parameter and gives the default value for each parameter. For information on how to choose appropriate values for the configuration parameters, see [“Usage Models” on page 15-13](#) and [“Performance Hints” on page 15-16](#).

Express-Mode Limitations

You cannot use express mode in certain situations. For example, express mode does not support the loading of simple large objects (**Simple LOs**) or Ext Type Data and smart-large-object data (**Ext Type** data types), ensuring constraints, or invoking triggers.

In addition, you cannot use DB-Access to select data from ANSI database tables if the data was loaded in express mode. If you attempt to select data from such tables, the database server rejects the SELECT statements because the tables were not archived. Only read-only access is allowed to the tables until a level-0 archive is performed.

For a list of express-mode limitations, see [“Express Mode” on page 15-5](#). For more information on characteristics of the express mode and the deluxe modes, see [“Modes” on page 15-3](#).

onstat Options for onpload

The **onstat** utility has two options that you can use to observe the behavior of database server threads during express-mode loads. Use the following command to display light-append information:

```
onstat -g lap
```

The **onstat -j** option provides an interactive mode that lets you gather special information about an **onpload** job. The **-j** option is documented in [Appendix F](#) of this manual.

The **onstat** utility is documented in your *Administrator's Guide*.

Devices for the Device Array

On a data-load job, each device runs independently of other devices. Thus mixing fast and slow devices does not adversely affect the speed of the load.

In most unload jobs, all devices receive equal amounts of data. Thus the speed of all devices is limited by the speed of the slowest device. If you have several fast devices and one or two slow devices, it might be advantageous to remove the slow devices.

When CPU resources are plentiful during an HPL job, the device controllers are a potential bottleneck. If you have configured extra converter threads and extra converter VPs, CPU use should be close to 100 percent. If CPU use is not close to 100 percent, the cause might be one of the following situations:

- The device controller is managing too many devices.
- The devices themselves are slow.

Usage Models

Three major usage models are envisioned for the HPL, as follows:

- Reorganizing computer configuration
- Altering the schema of a table
- Loading or unloading external data

Reorganizing Computer Configuration

If you are not changing the table schema, use a no-conversion job to unload and load when you need to reorganize the configuration of your computer or change to a different computer. The no-conversion mode is the fastest means of performing an unload or load because rows are unloaded in Informix internal format with no conversion and reloaded in the same fashion.



Important: You cannot use a no-conversion job when the source and target computers use different internal byte representations. For information about the byte-order type, see *“Machines Window”* on page 5-12.

For information about preparing for a no-conversion unload/load with **ipload**, see [“Using the No Conversion Job Option” on page 13-14](#). To set no-conversion mode when you are using the **onpload** utility at the command line, use the **-fn** option. For more information, see [Chapter 16, “The onpload Utility.”](#)

Altering the Schema of a Table

When you need to alter a table (add, drop, or change the data type of columns), use the Fixed Internal format. In Fixed Internal format, rows are unloaded in Informix internal format on a column-by-column basis. Thus you can drop, add, or modify columns and still minimize conversion overhead. For more information, see [“Format Type Group” on page 13-11](#).

Loading and Unloading Data

When you load or unload data from an external source, you must assess the type of data and the amount of conversion that is required so that you can choose appropriate configuration parameters.

The following sections show possible configuration parameters for two different types of load jobs. Suppose your hardware has the following configuration:

- Eight CPU symmetric multiprocessors, each about 40 MIPS
- Several (say 16) 300-megabyte disks (for the database)
- Four 1.2-megabyte tape devices (to load to and unload from)
- 512-megabyte memory

You also have this information about your system:

- The database server is running with seven CPU virtual processors.
- The data that you want to load has a row of about 150 bytes with a mix of **INT**, **DATE**, **DECIMAL**, **CHAR**, and **VARCHAR** data types. (This is similar to the **LINEITEM** table in the TPC-D database).

Settings for a No-Conversion Load or Unload

A no-conversion load or unload is not highly CPU intensive because no conversion is involved. In this case, the load or unload is expected to be limited by the speed of the tape devices. No-conversion loads and unloads are always completed in express mode. For more information, see [“Using the No Conversion Job Option” on page 13-14](#).

The following table lists sample values of configuration parameters for a raw load.

Configuration Parameter	Suggested Value	Comment
CONVERTVPS	4	This process is not CPU intensive.
CONVERTTHREADS	1	This process is not CPU intensive.
STRMBUFSIZE	128	Choose some multiple of the AIOBUFSIZE, up to about 8*AIOBUFSIZE.
STRMBUFFERS	5	Should be CONVERTTHREADS + 4.
AIOBUFSIZE	32	Choose a buffer size to match the best block size for the tape drive. Large buffers increase performance if sufficient memory is available.
AIOBUFFERS	5	CONVERTTHREADS + 4 or 2*CONVERTTHREADS, whichever is larger

Express-Mode Load with Delimited ASCII

In an express-mode load with delimited ASCII, the load or unload job is limited by the available CPU. The conversion to or from Informix types to ASCII is the most expensive portion of these operations. Hence, the following configuration might be more appropriate.

The following table lists sample values of configuration parameters for an express-mode load with delimited ASCII.

Configuration Parameter	Sample Value	Comment
CONVERTVPS	8	One convert VP per CPU
CONVERTTTHREADS	2	Four devices * 2 thread/device = 8 threads
STRMBUFSIZE	32	Should be some multiple of AIOBUFSIZE. For this CPU-intensive case, 1 or 2*AIOBUFSIZE is sufficient.
STRMBUFFERS	4	CONVERTTTHREADS + 4
AIOBUFSIZE	32	Choose a buffer size to match the best block size for the tape drive. Large buffers increase performance if sufficient memory is available.
AIOBUFFERS	5	CONVERTTTHREADS + 4 or 2*CONVERTTTHREADS, whichever is larger

Performance Hints

In general, the performance of the HPL depends on the underlying hardware resources: CPU, memory, disks, tapes, controllers, and so on. Any of these resources could be a bottleneck, depending on the speed of the resource, the load on the resource, and the particular nature of the load or unload.

For example, load and unload jobs that perform no conversions consume minimal CPU resources. These jobs are thus likely to be limited by device or controller bandwidth. On the other hand, ASCII loads and unloads are CPU intensive because of the overhead of conversion to and from ASCII. This section discusses some topics that you should consider when you try to improve performance.

Choose an Efficient Format

When you load data to or unload data from a non-Informix source, you can use fixed or delimited format in an appropriate code set such as ASCII or EBCDIC. In general, ASCII loads and unloads are the fastest. If you are using **onpload** for a machine or schema reorganization, choose the no-conversion format.

Delimited and fixed ASCII formats are comparable in behavior except when VARCHAR data is present. If the schema contains VARCHAR data and the length of the VARCHAR data varies greatly, you might want to choose delimited format.

Ensure Enough Converter Threads and VPs

As mentioned earlier, loads and unloads other than raw and fast-format ones are likely to be CPU intensive due to conversion overhead. In such cases, conversion speed is likely to determine the load or unload speed. It is thus important to use sufficient conversion resources (that is, enough converter threads and VPs).

The number of converter threads that is required for a device depends on the relative speeds of the device and the CPU as well as the data types in the table being loaded or unloaded. CHAR and VARCHAR formats are the cheapest to convert. INT, DATE, SMFLOAT, and FLOAT are more expensive. DECIMAL and MONEY are among the most expensive formats to convert.

The **PLCONFIG** file specifies the number of converter threads per device. You can override this value on the **onpload** command line with the **-M** option.

The number of converter VPs should be based on the conversion intensity of the load or unload and the number of physical CPUS on the computer. If the load or unload is expected to be convert intensive, you might want to specify the number of convert VPs to be the number of physical CPUS (or one fewer) to take advantage of all of the available CPUS. You can set the number of converter VPs in the **onpload** configuration file.

The database server and **onpload** client VPs might both be competing for the same physical CPU resources. To reduce contention, run only the number of VPs that are necessary on both the database server and **onpload** sides. However, if the number of database server VPs is already specified, you might have a choice only in the number of **onpload** VPs. In this case, the suggestions in the previous paragraph apply.

Too few converter threads and VPs can make conversion a bottleneck. On the other hand, too many converter threads can waste time in scheduling threads in and out of the VPs. In general, more than ten converter threads per VP is too many.

Ensure Enough Memory

To maximize available memory and scan resources, HPL automatically sets the **PDQRIORITY** environment variable to 100, if it is not already set. If the **PDQRIORITY** environment variable is set, HPL uses that value.

If the **PDQRIORITY** environment variable is set to 0 or if the **PDQRIORITY** environment variable is disabled on the database server, then HPL cannot unload multiple devices.

Ensure Enough Buffers of Adequate Size

The **onpload** utility lets you set the size and number of buffers in the **plconfig** configuration file. For adequate performance, you should provide at least two (preferably three) AIO and stream buffers per converter thread. The size of AIO buffers should be at least as large as the device block size, and the size of the stream buffers should be large (32 or 64 kilobytes), as the following table shows.

Configuration Parameter	Size	Comment
STRMBUFSIZE	64	Should be some multiple of AIOBUFSIZE for best performance

Configuration Parameter	Size	Comment
STRMBUFFERS	2 * CONVERTTHREADS	
AIOBUFSIZE	64	
AIOBUFFERS	2 * CONVERTTHREADS	

Increase the Commit Interval

In the deluxe modes, the commit interval specifies the number of records that should be loaded before the transaction is committed (see [“The Load Options Window” on page 12-15](#)). Low values (frequent commits) degrade performance and high values improve performance. If you increase the commit interval, you might need to increase the size of your logical-log buffers.

While larger commit intervals can speed up loads, larger commit intervals require larger logical-log space and increase the checkpoint time. These side effects impact other system users during **onpload** operations.

Limitation When Using the Excalibur Text DataBlade Module Indexes

You cannot create a load job with multiple devices to insert rows into a table that has an Excalibur Text DataBlade module index. A multiple-device load fails because the Excalibur Text DataBlade module does not handle concurrency correctly.

To avoid this limitation:

- Create a load with a single device
- Complete the load using multiple devices and then create the Excalibur Text DataBlade module index on the table

The onpload Utility

In This Chapter	16-3
Understanding the onpload Utility	16-3
Starting the onpload Utility	16-3
Using the onpload Utility	16-4
The onpload Utility Syntax.	16-4

In This Chapter

This chapter describes the syntax of the **onpload** utility. It includes descriptions of available options as well as descriptions of methods you can use to invoke **onpload**.

Understanding the onpload Utility

After you create the **onpload** database with the **ipload** interface, the **onpload** utility allows you to perform loads and unloads directly from the command line.

The **onpload** command uniquely specifies a row in the **session** table in the **onpload** database. Each row in the **session** table specifies all the components and options that are associated with a job.

Starting the onpload Utility

You can start **onpload** from the command line or from **ipload**. When you click **Run** in the Load Job window (Figure 12-2 on page 12-9) or in the Unload Job window (Figure 11-2 on page 11-7), **ipload** uses information from the **onpload** database to start **onpload**.

Typically, you use **ipload** to start a job when you plan to perform that particular load or unload once. For a job that needs to be run periodically, such as a weekly report, you might choose to run the job from the command line.

The information that you give to **onpload** as command-line arguments takes precedence over any information that is in the **onpload** database. The information from the command-line arguments is effective only for a single load. The command-line arguments do not affect the values in the **onpload** database or in the PLCONFIG configuration file.

Using the onpload Utility

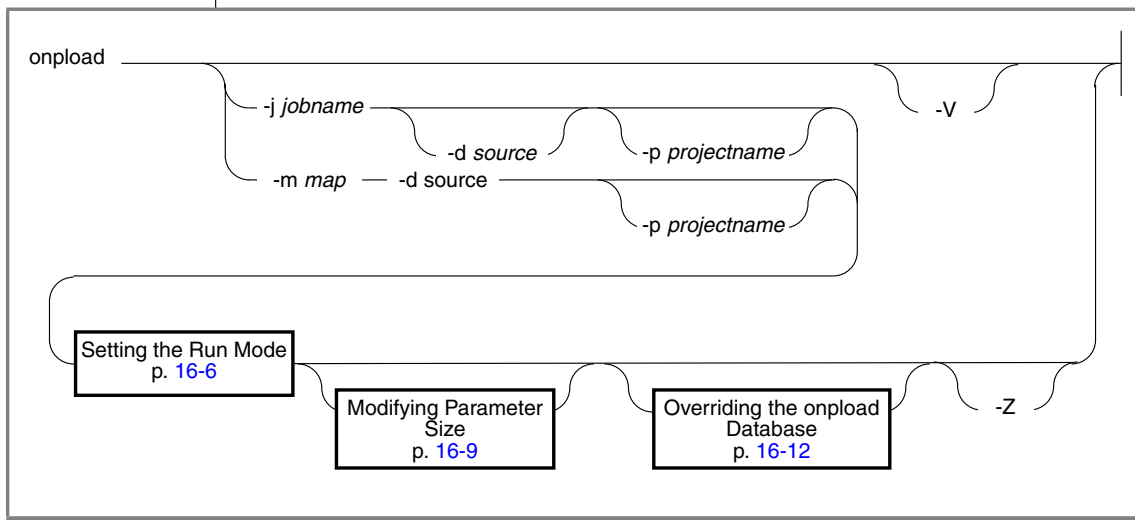
In most cases, use the Load Job window (page 12-9) or Unload Job window (page 11-7) to prepare the load or unload job. After you prepare the job, the **Command Line** text box on the Load Job Select window (page 12-8) or the Unload Job Select window (page 11-6) shows you the command line that **ipload** prepared for the job. You can copy that command line and use it to run the job at a later time. You can also use the command-line options shown in this chapter to modify the basic command line that **ipload** prepared.



Tip: Enter **onpload** with no options at the command line to display a command-line listing of all **onpload** options and their functions.

The following sections give additional information about the syntax and individual options of the **onpload** utility.

The onpload Utility Syntax



Element	Purpose	Key Considerations
-V	Displays the current version number and the software serial number.	Restrictions: This option is available only from the command line.
-d <i>source</i>	Sets the pathname of the file, tape, or pipe (on UNIX only) or the name of the device array to use for the load or unload session	<p>Additional Information: If the -f option is not set to a, d, or p, onpload assumes that the data source is a file.</p> <p>Additional Information: To use ipload, see “Interpreting the -d and -f Options Together” on page 16-8.</p>
-j <i>jobname</i>	Names a load or unload job from the onpload database	Additional Information: To set using ipload , see “Components of the Unload Job” on page 11-3 and “Components of the Load Job” on page 12-3.
-m <i>map</i>	Names a map from the onpload database	Additional Information: To use ipload , see “Using a Map” on page 9-3.
-p <i>projectname</i>	Identifies the project where the format and map are stored	Additional Information: To use ipload , see “Project Organization” on page 4-3.
-Z	Enables writing to or reading from a tape until the end of the device. The onpload utility prompts you for additional tapes until the load or unload is completed	<p>Restrictions: If you use this feature, you must use it for both the load and unload commands. Otherwise, the unloaded data might not match the loaded data.</p> <p>Additional Information: If the -Z option is not set, the onpload uses the tape size provided on the command line. If the -Z option is set, it supersedes the tape size information provided.</p> <p>Additional Information: This option is equivalent to checking the Write/read to/from tape until end of device checkbox on the Load Job Select or Unload-Job Select windows.</p>

The **pload** command line assumes the default project unless otherwise specified with the **-p** option.

For example, you might use the Load Job window to prepare the following command:

```
onpload -p zz -j bigload -fl
```

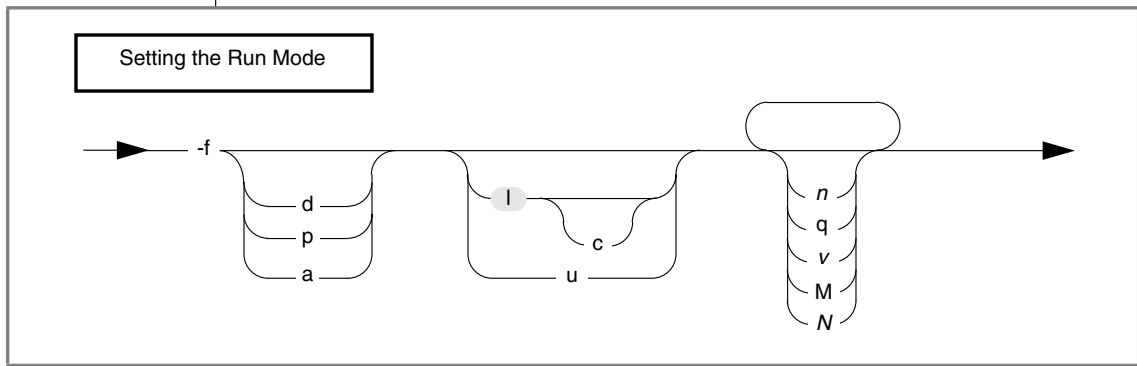
If you receive a tape that you know contains data with bugs, you might choose to modify the command to allow errors and to save the log in a special place, as follows:

```
onpload -p zz -j bigload -fl -e 1000 -l /mylogs/buggytape.log
```

For information on the **-fl** option, see [“Setting the Run Mode with the -f Option” on page 16-6](#).

Setting the Run Mode with the -f Option

The **-f** option lets you set the type of source data and the type of mode. Possible modes are: express load, deluxe load, deluxe load without replication, or unload.



Element	Purpose	Key Considerations
M	Displays the program module or line number in messages.	Restrictions: This flag is available only from the command line. Additional Information: This flag is used for debugging.
N	Allows deluxe mode load without replication	Restrictions: This flag works only if the -j job argument, which specifies a session table job to run, is left blank.
a	Treats data source as a device-array.	Additional Information: The definition of the device array is extracted from the onpload database. To use ipload , see “Device Arrays” on page 6-3 .
c	Sets mode to deluxe mode.	Additional Information: If this flag is not set, onpload uses express mode. To use ipload , see “Modes” on page 15-3 .

(1 of 2)

Element	Purpose	Key Considerations
d	Treats data source as a device (tape or file).	Additional Information: To set this option using ipload , see “Device Arrays” on page 6-3 .
l	Loads data into database.	Additional Information: This is the default flag, as opposed to u , which unloads data from the database. To use ipload , see “Components of the Load Job” on page 12-3 .
n	Specifies that onpload does not need to perform data conversion.	Restrictions: The target table for the load must have the same schema as the table from which the data is extracted. Additional Information: If onpload generated the input data file as an Informix format data file, you do not need to perform data conversion when you reload data. To use ipload , see “Using the No Conversion Job Option” on page 13-14 .
p	Treats data source as a program to execute and reads interface to the program by way of a pipe (on UNIX only).	Additional Information: To use ipload , see “Device Arrays” on page 6-3 .
q	Tells onpload not to generate status messages while a job is running.	None.
u	Unloads data from database.	Additional Information: If this flag is not set, onpload loads data into the database. To use ipload , see “Components of the Unload Job” on page 11-3 .
v	Tells onpload not to generate violations records.	Restrictions: This flag is available only from the command line, and is not supported with deluxe mode load.

(2 of 2)

Typing the -f Flags

When you combine **-f** flags into one group, do not put spaces between the flags. For example, use **-f acq**.

If you prefer, you can use multiple occurrences of the **-f** option instead of combining all of the possible **-f** flags into one group. For example, the following two command lines are equivalent:

```
onpload -m mymap -d mydev -finc
```

```
onpload -m mymap -d mydev -fl -fn -fc
```

Interpreting the -d and -f Options Together

The argument of the **-d** option gives the name of the data source. You can specify the device type of the data source with flags of the **-f** option, as follows:

- If the command line does not specify a device type, **onpload** treats the data source as the pathname of a cooked file on disk. Because no device type is specified, the following **onpload** command treats **filename** as the name of a file:

```
onpload -d filename -m mapname
```

- The **-fd** option in the following command causes **onpload** to treat **/dev/rmt/rst11b** as the name of a tape device:

```
onpload -d /dev/rmt/rst11b -m mapname -fd
```

The tape device name should be Berkeley Software Distribution (BSD) compliant.

- The **-fa** option in the following command causes **onpload** to treat **tapearray3** as the name of a device array. The device array is described in the **onpload** database.

```
onpload -d tapearray3 -m mapname -fa
```

- The **-fp** option in the following command causes **onpload** to treat **apipename** as the name of a pipe. When **onpload** starts executing, it causes the pipe process to start executing.

```
onpload -d apipename -m mapname -fp
```

◆

The same semantics apply for an unload job. If you use the **u** flag of the **-f** option to indicate an unload job, the interpretation of the data-source name is as described previously. For example, the following command specifies that **onpload** should unload data to the device **/dev/rmt/rst11**:

```
onpload -d /dev/rmt/rst11 -m mapname -fdu
```


Modifying Parameter Size

The options that are described in this section let you enter size information that overrides existing parameters in the **onpload** database.

Modifying Parameter Size

— *-A tapehead* —

— *-B blocksize* —

— *-G swapbytes* —

— *-l commit_int* —

— *-a iobufsize* —

— *-b bufsize* —

— *-e maxerrors* —

— *-i prog_interval* —

— *-n numrecs* —

— *-s startrec* —

— *-t numtapes* —

Element	Purpose	Key Considerations
-A <i>tapehead</i>	Tells onpload to skip the specified number of bytes on the tape before it starts reading data records.	<p>Restrictions: This option is available only from the command line.</p> <p>References: For specific details on this option, see “Session Table” on page A-17.</p>
-B <i>blocksize</i>	Sets the tape I/O block size (bytes).	<p>Additional Information: If the data source is a device array, this setting is ignored. To use ipload, see “The Tape Parameters Group” on page 6-8.</p>
-G <i>swapbytes</i>	Sets the number of bytes in a swap group.	<p>Restrictions: This option is available only from the command line.</p> <p>Additional Information: This option globally reverses the byte order in the input data stream. Each group of bytes is swapped with the group of bytes that follows it.</p>
-I <i>commit_int</i>	Sets the number of records to process before doing a commit.	<p>Restrictions: This option applies only to deluxe mode.</p> <p>Additional Information: To use ipload, see “The Load Options Window” on page 12-15.</p>
-a <i>iobufsize</i>	Sets the size (kilobytes) of the asynchronous I/O buffers, the memory buffers used to transfer data to and from tapes and files.	<p>Restrictions: This option is available only from the command line.</p> <p>Additional Information: This value overrides the value (AIOBUFSIZE) set in the HPL configuration file (plconfig).</p> <p>References: For specific details on this option, see “AIOBUFSIZE” on page B-4.</p>
-b <i>bufsize</i>	Sets the size (kilobytes) of the server stream buffer, the memory buffer used to write records to the database.	<p>Restrictions: This option is available only from the command line.</p> <p>Additional Information: Larger buffers result in more efficient data exchange with the database. This value overrides the value (STRMBUFSIZE) set in the HPL configuration file (plconfig).</p> <p>References: For specific details on this option, see “STRMBUFSIZE” on page B-10.</p>
-e <i>maxerrors</i>	Sets the error threshold that causes the load or unload session to shut down.	<p>Additional Information: If no number is specified, the default is to process all records. To use ipload, see “The Load Options Window” on page 12-15.</p>

(1 of 2)

Element	Purpose	Key Considerations
-i <i>prog_interval</i>	Sets the number of records to process before making an entry in the log file specified by the -l option.	<p>Restrictions: This option is available only from the command line.</p> <p>Additional Information: If no log file is specified, progress messages are sent to stdout. If the -i option is omitted, the default number is 1000. To set, see “Using the -i Option” on page 11.</p>
-n <i>numrecs</i>	Sets the number of records to load.	<p>Additional Information: If no number is specified, all records are processed. <i>The -n option does not affect unload operations because pload cannot maintain row ordering.</i> To use ipload, see “The Load Options Window” on page 12-15.</p>
-s <i>startrec</i>	Sets the starting record to load.	<p>Additional Information: This option is used to skip records. For example, setting -s to 10 starts the loading at the tenth record, so that if the file contains 20 records, 11 records are loaded. Setting -s to 0 or 1 the load starts with the first record. If you do not set this option, the load starts with the first record.</p> <p><i>The -s option does not affect unload operations because pload cannot maintain row ordering.</i> To use ipload, see “The Load Options Window” on page 12-15.</p>
-t <i>numtapes</i>	Specifies the number of tapes to load.	<p>Additional Information: If you do not set this option, the default value is 1. To use ipload, see “The Load Options Window” on page 12-15.</p>

(2 of 2)

Using the -i Option

The **-i** option lets you specify the number of records to process before **onpload** reports the progress in an entry in the log file. The **onpload** utility calculates progress message count as follows:

```
row_count = (total_rows/prog_interval) * prog_interval
```

The **onpload** utility updates the row count only once for each stream buffer of data that it processes. Thus, reducing the row count on the **-i** option does not necessarily increase the number of progress messages in the log file. For example, if the stream buffer holds 910 rows of data, setting *row_count* to 10, 100, and 900 has the same effect: **onpload** writes one progress message.

Overriding the onpload Database Values

The options that are described in this section let you enter size information that overrides existing parameters in the **onpload** database.

Overriding the onpload
Database Values

-C *caseconvert*

-D *override_db*

-F *filter*

-L *trace_level*

-M *converters*

-R *rejectfile*

-S *servername*

-T *target_db*

-I *logfile*

Element	Purpose	Key Considerations
-C <i>caseconvert</i>	Sets the case-conversion option that converts all character information.	<p>Additional Information: If you do not set this option, no case conversion is done. Possible flags include:</p> <ul style="list-style-type: none"> U or u = uppercase L or l = lowercase P or p = proper names <i>blank</i> = no conversion <p>To use ipload, see “The Mapping Options Window” on page 9-15.</p>
-D <i>override_db</i>	Overrides the database specified in the map used for the load.	<p>Additional Information: To use ipload, see “The Load Record Maps Window” on page 9-7.</p>
-F <i>filter</i>	Identifies the filter that onpload uses for screening load records.	<p>Additional Information: To use ipload, see “Using a Filter” on page 10-4.</p>
-L <i>trace_level</i>	Sets the amount of information logged during the load.	<p>Restrictions: This option is available only from the command line. The value of <i>trace_level</i> must be an integer from 1 to 5.</p> <p>Additional Information: The default value is 1. Higher values result in more output. Do not use this option unless you are doing serious debugging.</p>
-M <i>converters</i>	Sets the maximum number of conversion threads per device.	<p>Restrictions: This option is available only from the command line.</p> <p>Additional Information: This value overrides the value of CONVERTTTHREADS set in the HPL configuration file (plconfig). If the value for <i>converters</i> is greater than 1, onpload can dynamically allocate more conversion threads as needed to process data.</p> <p>References: For specific details on this option, see “CONVERTVPS” on page B-6.</p>

(1 of 2)

Element	Purpose	Key Considerations
-R <i>rejectfile</i>	Identifies the file destination for rejected records.	Additional Information: The file is named <i>rejectfile.rej</i> . To use ipload , see “The Load Job Window” on page 12-9 .
-S <i>servername</i>	Sets the onpload database server.	Restrictions: The onpload utility must be located on the same network as the onpload database and on the same server as the target database. You can only run jobs on remote database servers using the ipload utility. Additional Information: To use ipload , see “The Connect Server Window” on page 5-4 .
-T <i>target_db</i>	Sets the target database server.	Restrictions: The onpload utility must be located on the same network as the onpload database and on the same server as the target database. You can only run jobs on remote database servers using the ipload utility. Additional Information: To use ipload , see “The Connect Server Window” on page 5-4 .
-l <i>logfile</i>	Specifies the name of a file to which onpload sends messages.	Additional Information: If you do not specify a log file, onpload sends messages to stdout . To use ipload , see “The Unload Job Window” on page 11-7 and “The Load Job Window” on page 12-9 .

(2 of 2)

The onpladm Utility

In This Chapter	17-5
Description of the onpladm Utility	17-5
The onpladm Utility Features	17-5
Specification-File Conventions	17-6
Error Handling	17-9
Defining Jobs	17-10
Creating Jobs	17-10
Creating Conversion Jobs	17-10
Creating No-Conversion Jobs	17-17
Modifying a Job by Using a Detailed Specification File	17-21
Describing a Job	17-22
Listing All Jobs in a Project	17-23
Running a Job	17-24
Deleting a Job	17-25
Defining Device Arrays	17-26
Creating a Device Array	17-26
Modifying a Device Array	17-27
Describing a Device Array	17-27
Listing Project Device Arrays	17-28
Deleting a Device Array	17-29
Defining Maps	17-29
Creating Maps	17-29
Creating a Map by Using a Quick Command	17-30
Creating a Map With a Detailed Specification File	17-32
Deleting a Map	17-37
Describing a Map	17-38

Modifying a Map Using A Detailed Specification File.	17-38
Listing All Maps in a Project	17-39
Defining Formats	17-39
Creating a Format	17-39
Modifying a Format Using a Specification File	17-43
Describing a Format	17-44
Listing all Formats in a Project.	17-44
Deleting a Format	17-46
Defining Queries.	17-46
Creating a Query	17-46
Modifying a Query	17-47
Describing a Query	17-48
Listing all Queries in a Project.	17-48
Deleting a Query	17-50
Defining Filters	17-50
Creating a Filter.	17-50
Modifying a Filter	17-51
Describing a Filter	17-52
Listing all Filters in a Project	17-52
Deleting a Filter.	17-54
Defining Projects.	17-55
Creating a New Project	17-55
Running All Jobs in a Project	17-56
Listing all Projects	17-57
Deleting a Project	17-57
Defining Machine Types	17-58
Creating a Machine Type	17-58
Modifying a Machine Type	17-59
Describing a Machine.	17-60
Listing all Existing Machine Types	17-61
Deleting a Machine Type	17-61
Defining Database Operations	17-62
Creating a Database Project.	17-62
Configuring Target-Server Attributes	17-65

Setting Target-Server Attribute Values17-66

Listing Target-Server Defaults17-67

In This Chapter

This chapter describes the syntax of the **onpladm** utility. The **onpladm** command-line interface is equivalent to the **ipload** utility.

Description of the onpladm Utility

You can use the **onpladm** utility from the command line to create, modify, and delete High-Performance Loader (HPL) objects. The HPL objects include projects, jobs, maps, formats, queries, filters, device arrays, and machines.

You can use the **onpladm** utility on both UNIX and Windows computers. For information on how to run **onpladm** commands on a combination of UNIX and Windows computers, see [“Running onpladm on UNIX with Dynamic Server Running on Windows”](#) on page I-2.

The onpladm Utility Features

You can create and maintain the **onpload** database with the **onpladm** utility and the **ipload** utility. The **onpload** database stores information about the HPL objects of load and unload jobs.

The first time that you issue a **create** command, the **onpladm** utility creates the **onpload** database, if it does not yet exist.

The **onpladm** utility can perform the functions that the following table describes.

Object	Operation
Project	Create, delete, describe, list, run
Job	Create, delete, describe, list, modify, run
Map	Create, delete, describe, list, modify
Format	Create, delete, describe, list, modify
Query	Create, delete, describe, list, modify
Device array	Create, delete, describe, list, modify
Computer	Create, delete, describe, list, modify
Target server	Configure and list default settings
Filter	Create, delete, describe, list, modify

The **onpladm** utility also allows you to:

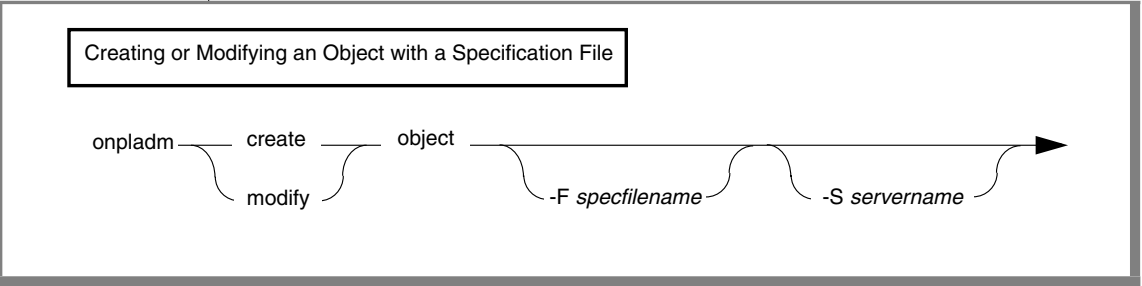
- Create jobs and other objects with a single, minimum-input command.
- Specify object characteristics in specification files.
- Create, load, or unload jobs for all tables in a database.
- Create jobs for an entire database and group the jobs into a project with a single command.
- Execute individual jobs or projects independently.

Specification-File Conventions

You use specification files to create, modify, and describe the HPL objects. When you enter the **onpladm** command to describe an HPL object, the contents of the specification file appear.

When you create a job or map with a quick command, the **onpladm** utility uses default attributes to create the job or map. If you create a job or map with a specification file, you can specify attribute values.

The following diagram illustrates the syntax to create or modify an object using a specification file.



Element	Purpose	Key Considerations
<code>-F specfilename</code>	Sets the specification file	Additional Information: The default value is the standard output.
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Use the following conventions when you create specification files:

- Begin object definitions with `BEGIN OBJECT` and end them with `END OBJECT`.
- If object definitions contain variable items, begin each variable item with `BEGIN SEQUENCE` and end each item with `END SEQUENCE`.

For example, you might use the following specification file to create a device array that consists of a file and a pipe:

```
BEGIN OBJECT DEVICEARRAY mydevice
# Optional Attributes
BEGIN SEQUENCE
TYPE FILE
FILE /work/data.unl
TAPEBLOCKSIZE 0
TAPEDEVICESTRIDE 0
PIPECOMMAND
END SEQUENCE
BEGIN SEQUENCE
TYPE PIPE
FILE
TAPEBLOCKSIZE 0
TAPEDEVICESTRIDE 0
PIPECOMMAND /work/bin/datacreate.sh
END SEQUENCE
END OBJECT
```

For more information about attributes and their possible values, refer to the description of each kind of specification file.

- Precede comments in specification files with a pound sign (#).
- List attributes in the exact order in which the specification-file format displays them.
- Use the following syntax to refer to the attributes of an object or the attributes of elements of an object:

Attribute_name Attribute_value

Important: Do not use this for `BEGIN` and `END` statements and comment statements.

You must always provide an attribute name. You must provide both the attribute name and the attribute value to describe a required attribute, but you only have to provide the attribute name if the attribute is optional.



Attributes and their values depend on their object type. For more information on attributes and object types, see the corresponding specification files.

- Enclose attribute values that contain spaces in double quotation marks.
- Precede double quotation marks in attribute values with a double quotation mark.

For example, to enter a MATCH condition for "CA" in a filter object, include the following line:

```
MATCH = " "CA" "
```

For more information on file conventions, see individual specification-file formats.

Error Handling

The **onpladm** commands have two return values:

- 0 if the command is successful. A success message appears.
- 1 if the command fails. An error message appears.

Defining Jobs

You can create, modify, describe, list, run, and delete jobs with the **onpladm** utility. For more information on load jobs, see [“Components of the Load Job” on page 12-3](#).

Creating Jobs

You can create two types of jobs:

- **Conversion jobs**
Use a conversion job to process data that requires conversion before loading and that is not in internal format. Conversion jobs have associated maps and format objects.
- **No-conversion jobs (fast jobs)**
Use a no-conversion job to process data that does not require conversion before loading and that uses the Informix internal format. No maps or formats are associated with a no-conversion job. Consequently, a no-conversion job is faster than a conversion job. No-conversion jobs are always completed in express mode.

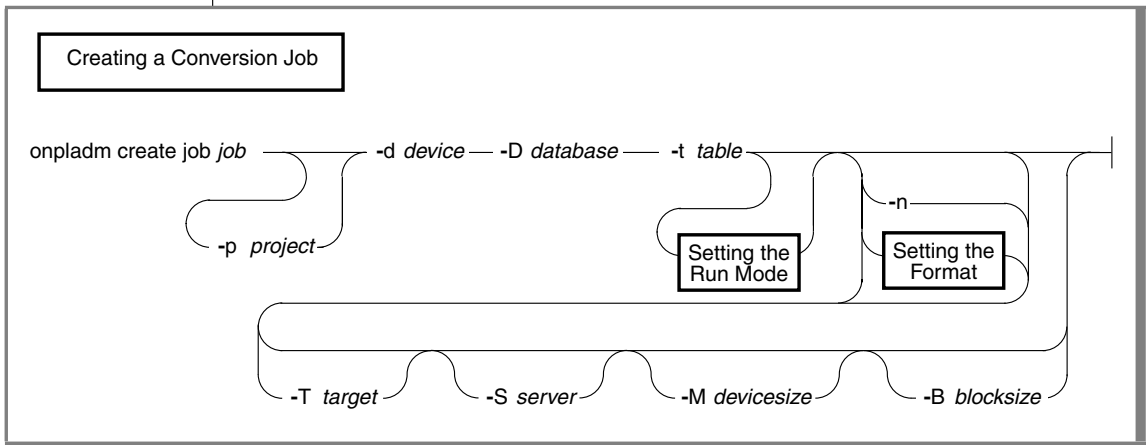
Creating Conversion Jobs

When you create a conversion load or unload job using command-line parameters, all maps, formats, and query objects automatically have the same name as the job name. When you create a conversion job with detailed specification files, the HPL objects can have different names from the job name.

Creating Conversion Jobs Using a Quick Command

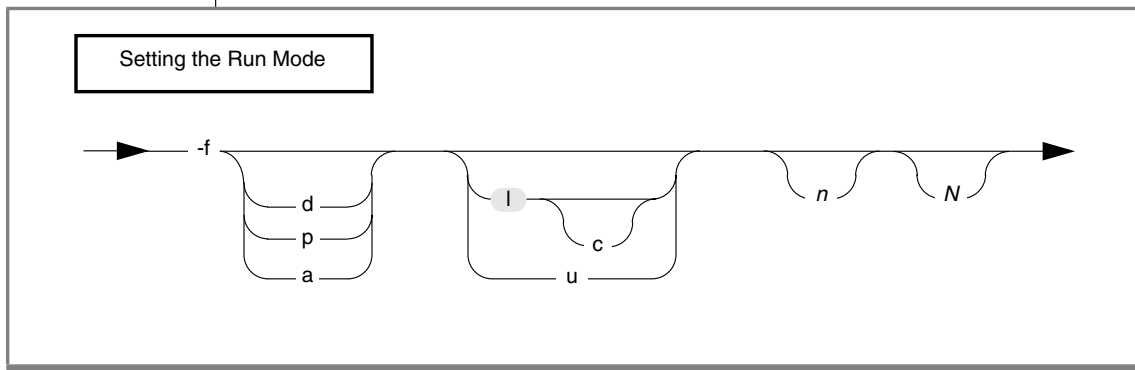
When you create a conversion job using a quick command, the **onpladm** utility creates all HPL objects associated with the job. The HPL objects that it creates have the same name as the job.

The following diagram illustrates the syntax to create a conversion job from the command line.



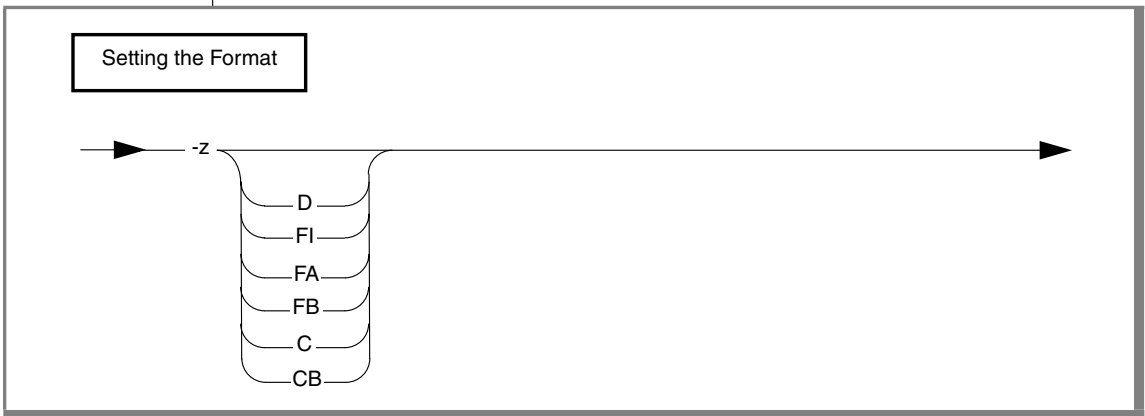
Element	Purpose	Key Considerations
-B <i>blocksize</i>	Sets the tape I/O block size (bytes)	Additional Information: No default value.
-d <i>device</i>	Sets the name of device, such as a file, device array, tape, or pipe	Additional Information: No default value.
-D <i>database</i>	Name of the target database that contains the information to be loaded or unloaded	Additional Information: No default value.
<i>job</i>	Names a load or unload job from the onpload database	None
-M <i>devicesize</i>	Tape device size in kilobytes	Additional Information: The device size must be greater than zero.
-n	Sets no-conversion express job	None
-p <i>project</i>	Identifies the project where the format and map are stored	Additional Information: The default is the project created when the onpload database is built.
-S <i>server</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-t <i>table</i>	Name of the table to be loaded or unloaded	None
-T <i>target</i>	Name of the target server to which the data will download	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

The following diagram illustrates the syntax to set the run mode with the **-f** option.



Element	Purpose	Key Considerations
a	Treats data source as a device-array	Additional Information: The definition of the device array is extracted from the onpload database.
c	Sets mode to deluxe mode.	Additional Information: If this flag is not set, onpladm uses express mode.
d	Treats data source as a tape device	None
l	Loads data into database.	Additional Information: This is the default flag, as opposed to u , which unloads data from the database.
n	Specifies that onpladm does not need to perform data conversion.	Restrictions: The target table for the load must have the same schema as the table from which the data is extracted. Additional Information: If onpladm generated the input data file as an Informix format data file, you do not need to perform data conversion when you reload data.
N	Allows deluxe mode load without replication	Restrictions: This flag works only if the <i>job</i> argument, which specifies a session table job to run, is left blank.
p	Treats data source as a program to execute and reads interface to the program by way of a pipe (on UNIX only)	None
u	Unloads data from database	Additional Information: If this flag is not set, onpladm loads data into the database.

The following diagram illustrates the syntax to set the format type with the **-z** option.



Element	Purpose	Key Considerations
D	Sets the format to delimited	Additional Information: This is the default value. See “Delimited Records” on page 7-19.
FI	Sets the format to fixed internal	Additional Information: See “Fixed-Length Records” on page 7-4.
FA	Sets the format to fixed ASCII	Additional Information: See “Fixed-Length Records” on page 7-4.
FB	Sets the format to fixed binary	Additional Information: See “Fixed-Length Records” on page 7-4.
C	Sets the format to COBOL	Additional Information: See “COBOL Records” on page 7-24.
CB	Sets the format to COBOL (byte)	Additional Information: See “COBOL Records” on page 7-24.

Creating Conversion Jobs Using Detailed Specification Files

You can also specify job details in specification files that you reference from the command line.

When you create a job by using a specification file, you must create all associated HPL objects; the **onpladm** utility does not create these objects for you.

Creating a Conversion-Load Job

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a conversion-load job from a specification file.

Use the following syntax to create a specification file for a conversion-load job:

```
BEGIN OBJECT LOADJOB jobname
# Compulsory Attributes
PROJECT projectname
DEVICE device_array_name
MAP mapname

# Optional Attributes
FILTER filtername
SERVER targetservername
DATABASE targetdatabasename
FLTFILE filtered_records_filename
REJECTFILE rejected_records_filename
LOGFILE job_progress_logfilename
RUNMODE runmode_type
GENERATEVIORECS violation_records_option
TAPES sourcetape_num
NUMRECORDS records_num
STARTRECORD start_record
MAXERRORS max_error_num

END OBJECT
```

The following table lists the attributes and their attribute values.

Attribute	Attribute Value
<i>device_array_name</i>	Device-array name You must create the device array; onpladm does not create it for you.
<i>filtername</i>	Filter name
<i>jobname</i>	Job name
<i>job_progress_logfilename</i>	Path to the job-progress log file
<i>mapname</i>	Map name You must create the map before you use this option; onpladm does not create the map for you.

(1 of 2)

Attribute	Attribute Value
<i>max_error_num</i>	Maximum number of errors; if exceeded, load ends
<i>projectname</i>	Name of an existing project
<i>records_num</i>	Number of records to be processed in the data file
<i>rejected_records_filename</i>	Rejected-records filename (rejected by database server)
<i>filtered_records_filename</i>	Filtered-records filename (rejected by filter)
<i>runmode_type</i>	Type of run mode: E = Express mode D = Deluxe mode without replication DR = Deluxe mode with replication
<i>sourcetape_num</i>	Number of tapes that contain source data
<i>start_record</i>	Number of the first record to begin a load
<i>targetdatabasename</i>	Name of the database that the records will be loaded or unloaded to; if set, this value overrides the database value in the load or unload map
<i>targetservername</i>	Name of the target database server
<i>violation_records_option</i>	Specify: Y to generate violations records N not to generate violations records

(2 of 2)

Creating a Conversion Unload Job

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a conversion unload job from a specification file.

Use the following syntax to create a conversion unload job:

```
BEGIN OBJECT UNLOADJOB jobname
# Compulsory Attributes
PROJECT projectname
DEVICE device_array_name
MAP mapname

# Optional Attributes
FILTER filtername
SERVER targetservername
DATABASE targetdatabasename
REJECTFILE rejected_records_filename
LOGFILE job_progress_logfilename
ISOLATIONLEVEL isolation_level
MAXERRORS max_error_num

END OBJECT
```

The following table lists the arguments and their attribute values.

Attribute	Attribute Value
<i>device_array_name</i>	Device-array name You must create the device array; onpladm does not create it for you.
<i>filtername</i>	Filter name
<i>isolation_level</i>	Unload isolation level: DR = Dirty Read CR = Committed Read CS = Cursor Stability RR = Repeatable Read
<i>jobname</i>	Job name
<i>job_progress_logfilename</i>	Path to the job-progress log file
<i>mapname</i>	Map name You must create the map before you use this option; onpladm does not create the map for you.

Attribute	Attribute Value
<i>max_error_num</i>	Maximum number of errors; if exceeded, unload ends
<i>projectname</i>	Name of an existing project
<i>rejected_records_filename</i>	Rejected-records filename (rejected by database server)
<i>targetdatabasename</i>	Name of the database that the records will be loaded or unloaded to; if set, this value overrides the database value in the load or unload map
<i>targetservername</i>	Name of the target database server

(2 of 2)

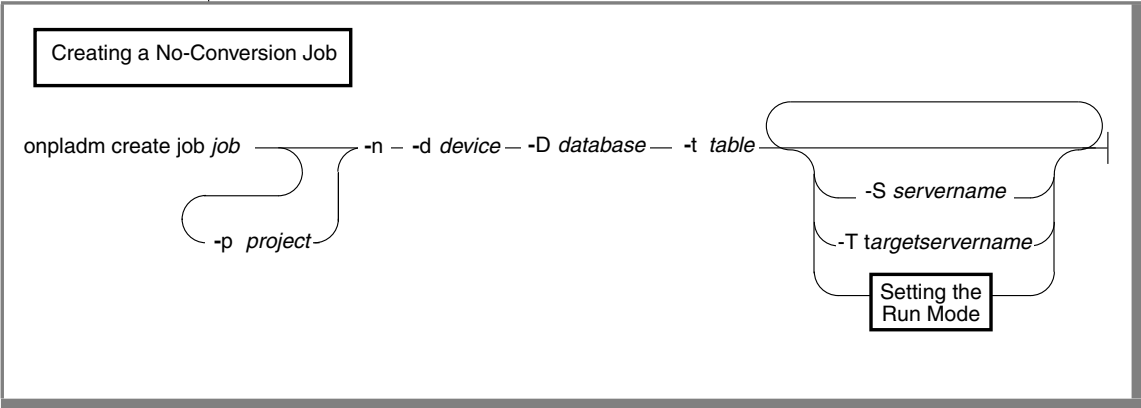
Creating No-Conversion Jobs

A no-conversion job is faster than a conversion job because **onpload** uses the internal format of the target database and does not create new maps or formats. No-conversion jobs are always run in express mode.

Creating No-Conversion Jobs Using a Quick Command

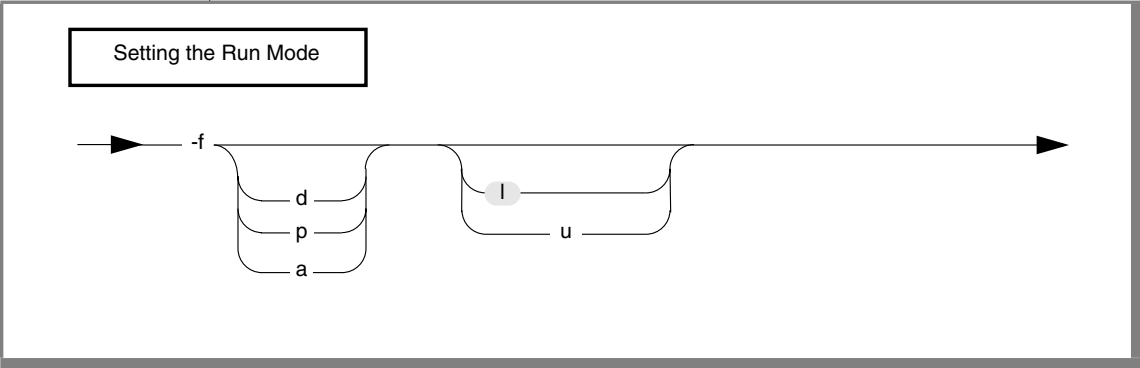
When you create a job using a quick command, the **onpladm** utility creates all HPL objects associated with that job.

The following diagram illustrates the syntax to create a no-conversion job.



Element	Purpose	Key Considerations
-d <i>device</i>	Sets the name of device, such as a file, device array, tape, or pipe	Additional Information: No default value.
-D <i>database</i>	Name of the target database that contains the information to be loaded or unloaded	Additional Information: No default value.
<i>job</i>	Names a load or unload job from the onpload database	None
-n	Sets no-conversion express job	None
-p <i>project</i>	Identifies the project where the format and map are stored	Additional Information: The default is the project created when the onpload database is built.
-S <i>server</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-t <i>table</i>	Name of the table to be loaded or unloaded	None
-T <i>target</i>	Name of the target server to which the data will download	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

The following diagram illustrates the syntax to set the run mode with the **-f** option.



Element	Purpose	Key Considerations
a	Treats data source as a device-array.	Additional Information: The definition of the device array is extracted from the onpload database.
d	Treats data source as a tape device.	None
l	Loads data into database.	Additional Information: This is the default flag, as opposed to u , which unloads data from the database.
p	Treats data source as a program to execute and reads interface to the program by way of a pipe (on UNIX only).	None
u	Unloads data from database.	Additional Information: If this flag is not set, onpladm loads data into the database.

Creating No-Conversion Jobs Using Detailed Specification Files

You can also specify job details in specification files that you reference from the command line.

When you create a job by using a specification file, you must create all associated HPL objects; the **onpladm** utility does not create these objects for you.

Creating a No-Conversion Load Job

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a no-conversion load job with specification files.

Use the following syntax to create a specification file for a no-conversion load job:

```
BEGIN OBJECT FASTLOADJOB jobname
# Compulsory Attributes
PROJECT projectname
DEVICE device_array_name
DATABASE targetdatabasename
TABLE tablename

# Optional attributes
SERVER targetservername

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>device_array_name</i>	Device-array name You must create the device array; onpladm does not create it for you.
<i>jobname</i>	Job name
<i>projectname</i>	Name of an existing project
<i>tablename</i>	Table name
<i>targetdatabasename</i>	Name of the database that the records will be loaded or unloaded to; if set, this value overrides the database value in the load map
<i>targetservername</i>	Name of the target database server

Creating a No-Conversion Unload Job

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a no-conversion unload job with specification files.

Use the following syntax to create a specification file for a no-conversion unload job:

```
BEGIN OBJECT FASTUNLOADJOB jobname
# Compulsory Attributes
PROJECT projectname
DEVICE device_array_name
DATABASE targetdatabasename
QUERY queryname

# Optional attributes
SERVER targetservername

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>device_array_name</i>	Device-array name You must create the device array; onpladm does not create it for you.
<i>jobname</i>	Job name
<i>projectname</i>	Name of an existing project
<i>queryname</i>	The SQL SELECT statement, in quotation marks, that contains unload criteria
<i>targetdatabasename</i>	Name of the database that the records will be loaded or unloaded to; if set, this value overrides the database value in the load map
<i>targetservername</i>	Name of the target database server

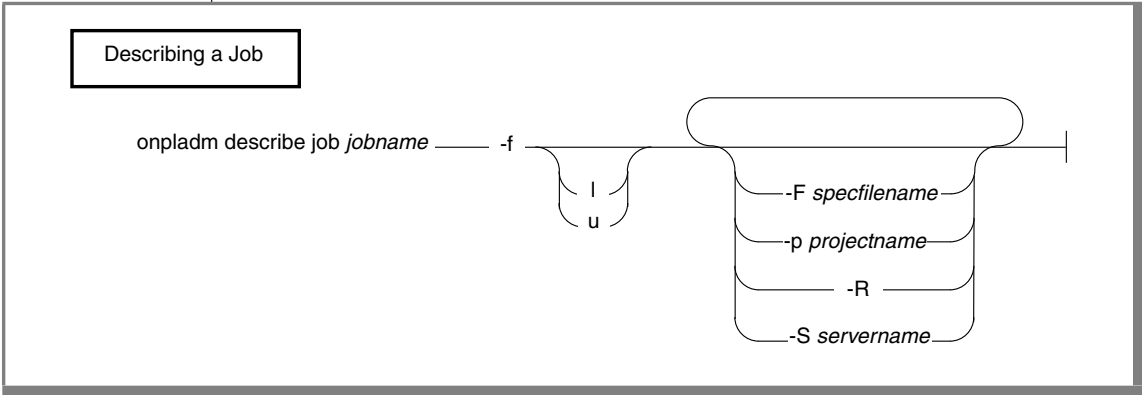
Modifying a Job by Using a Detailed Specification File

Use the syntax shown in [“Specification-File Conventions”](#) on page 17-6 to modify a job by using a specification file.

For information about the job-specification file, refer to [“Creating No-Conversion Jobs Using Detailed Specification Files”](#) on page 17-19.

Describing a Job

The following diagram illustrates the syntax to describe a job.



Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
<i>jobname</i>	Names a job from the onpload database	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-R	Deletes all associated objects if they are not referenced by other objects	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file used to describe a job, see [“Creating Conversion Jobs Using Detailed Specification Files”](#) on page 17-13.

Listing All Jobs in a Project

The following diagram illustrates the syntax to display all the jobs in a project.

Displaying All Jobs in a Project

onpladm list job

-p *projectname*

-f

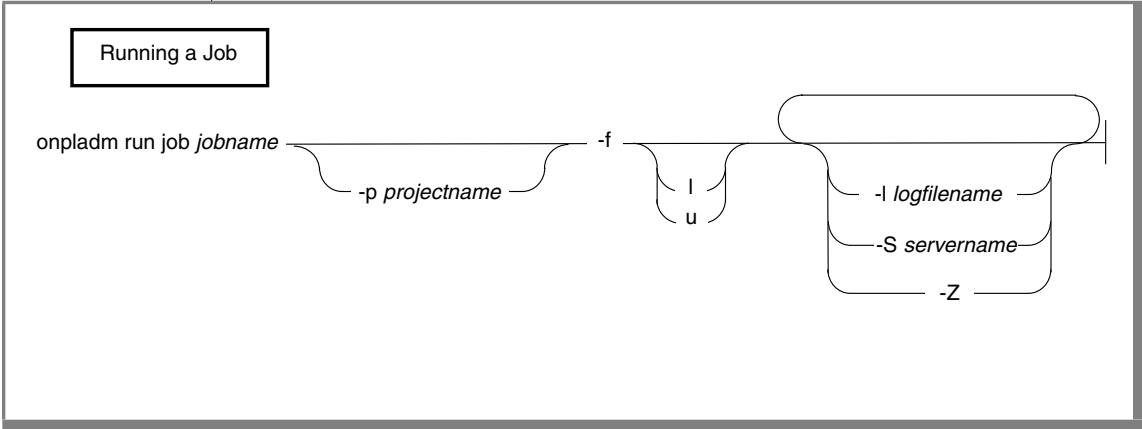
l
u

-S *servername*

Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Running a Job

The following diagram illustrates the syntax to run a job and display the job progress or send it to a log file.



Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
<i>jobname</i>	Names a load or unload job from the onpload database	None
-l logname	Sets the path for a log file where job progress is recorded	None
-p projectname	Identifies the project where the format and map are stored	None
-S servername	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-Z	Enables writing to or reading from a tape until the end of the device. The onpladm utility prompts you for additional tapes until the load or unload is completed.	Restrictions: If you use this feature, you must use it for both load and unload jobs. Otherwise, the unloaded data might not match the loaded data. Additional Information: If the -Z option is not set, the onpladm uses the tape size specified with the -M option of the create job command. If the -Z option is set, it supersedes the tape size information provided.

For information on how to execute this command for a database server running on Windows, see [“Running onpladm on UNIX with Dynamic Server Running on Windows” on page I-2](#).

Deleting a Job

The following diagram illustrates the syntax to delete a job.

Deleting a Job

onpladm delete job *jobname*

-f

l

u

-p *projectname*

-S *servername*

-R

Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
<i>jobname</i>	Names a load or unload job from the onpload database	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-R	Deletes all associated objects if they are not referenced by other objects	None

Defining Device Arrays

You can create, modify, describe, list, and delete device arrays with the **onpladm** utility. For more information on device arrays, see [“Device Arrays” on page 6-3](#) and [“Devices for the Device Array” on page 15-13](#).

Creating a Device Array

You can only create a device array by using a detailed specification file.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a device array.

Use the following syntax to create a device array:

```
BEGIN OBJECT DEVICEARRAY device_array_name
# Compulsory Attributes
BEGIN SEQUENCE
TYPE device_type
FILE device_path
TAPEBLOCKSIZE tapeblock_size
TAPEDEVICESTRIDE tapedevice_size
PIPECOMMAND pipe_commandname
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>device_array_name</i>	Device-array name You must create the device array; onpladm does not create it for you.
<i>device_path</i>	Path to the device; only valid if device type is file or tape
<i>device_type</i>	Type of device, in uppercase letters (for instance, PIPE, FILE, or TAPE)

(1 of 2)

Attribute	Attribute Value
<i>tapeblock_size</i>	Tape-block size
<i>tapedevice_size</i>	Tape-device size in megabytes
<i>pipe_commandname</i>	Pipe command name

(2 of 2)

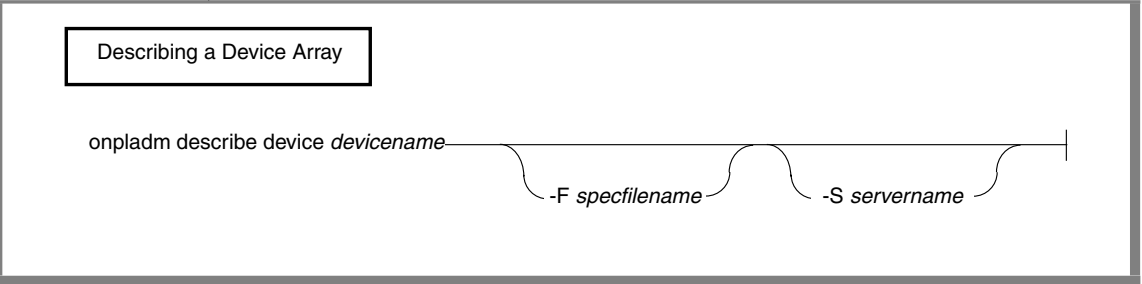
Modifying a Device Array

You can only modify a device array with a specification file. For information about the device array specification file, refer to [“Creating a Device Array” on page 17-26](#).

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify a device array.

Describing a Device Array

The following diagram illustrates the syntax to describe a device array.



Element	Purpose	Key Considerations
<i>devicename</i>	Sets the name of device, such as a file, device array, tape, or pipe	None
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file used to describe a device array, see [“Creating a Device Array” on page 17-26](#).

Listing Project Device Arrays

To list all the device arrays in a project, use the following syntax.

Listing Project Device Arrays

```
onpladm list device -p projectname -S servername
```

Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Device Array

The following diagram illustrates the syntax to delete a device array.

Deleting Project Device Arrays



Element	Purpose	Key Considerations
<i>devicename</i>	Sets the name of device, such as a file, device array, tape, or pipe	None
<i>-S servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Maps

You can create, modify, describe, list, and delete maps with the **onpladm** utility. For more information on maps, see [“Load and Unload Maps” on page 9-3](#).

Creating Maps

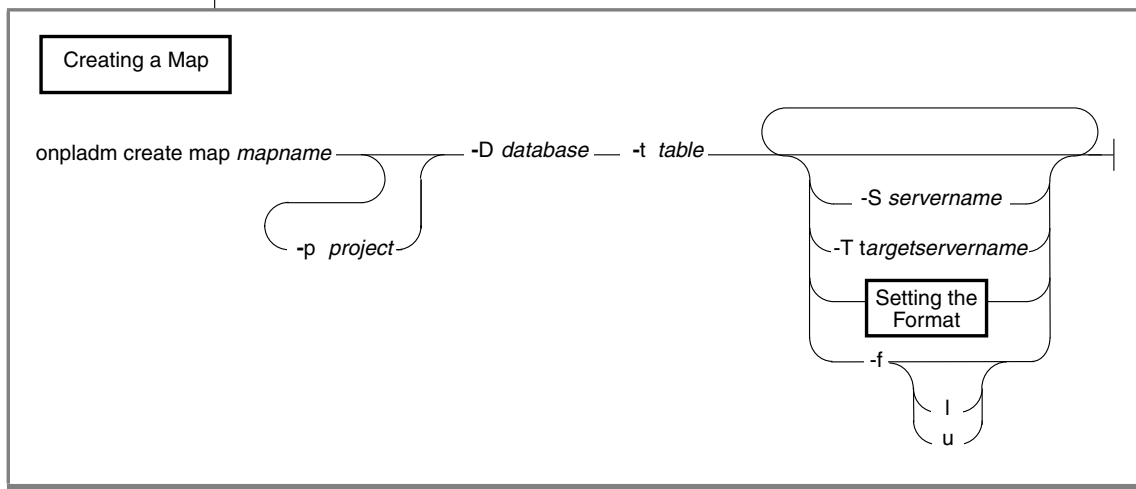
You can create maps by using a quick command or a specification file. For more information on specification-file conventions, see [“Specification-File Conventions” on page 17-6](#).

Creating a Map by Using a Quick Command

When you create a map by using a quick command, the **onpladm** utility creates a format object with the same name as the map, plus the suffix **-fmt**. The generated format name (as for all **onpladm** objects) has a maximum length of 18 characters.

For example, if the map name is **mymap**, the format name is **mymap-fmt**. If the map name is **123456789123456789**, the format name is **12345678912345-fmt**.

The **create map** command also creates a query object for the unload map. The following diagram illustrates the syntax to create a map from the command line.



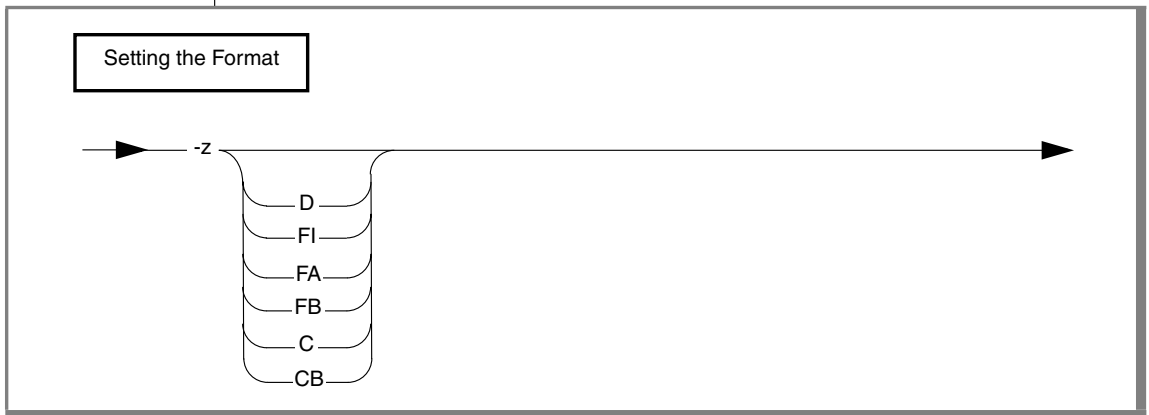
Element	Purpose	Key Considerations
-D database	Name of the target database that contains the information to be loaded or unloaded	Additional Information: No default value.
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
mapname	Sets the map	None

(1 of 2)

Element	Purpose	Key Considerations
-p <i>project</i>	Identifies the project where the format and map are stored	Additional Information: The default is the project created when the onpload database is built.
-S <i>server</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-t <i>table</i>	Name of the table to be loaded or unloaded	None
-T <i>target</i>	Name of the target server to which the data will download	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

(2 of 2)

The following diagram illustrates the syntax to set the format type with the **-z** option.



Element	Purpose	Key Considerations
D	Sets the format to delimited.	Additional Information: This is the default value. See “Delimited Records” on page 7-19.
FI	Sets the format to fixed internal.	Additional Information: See “Fixed-Length Records” on page 7-4.
FA	Sets the format to fixed ASCII.	Additional Information: See “Fixed-Length Records” on page 7-4.

(1 of 2)

Element	Purpose	Key Considerations
FB	Sets the format to fixed binary.	Additional Information: See “Fixed-Length Records” on page 7-4.
C	Sets the format to COBOL.	Additional Information: See “COBOL Records” on page 7-24.
CB	Sets the format to COBOL (byte).	Additional Information: See “COBOL Records” on page 7-24.

(2 of 2)

Creating a Map With a Detailed Specification File

When you create a map, the **onpladm** utility creates a format object with the same name as the map.

When you create a map with a specification file, you must create all associated HPL objects; the **onpladm** utility does not create these objects for you.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create maps with specification files.

Use the following syntax to creating a load map with a specification file:

```
BEGIN OBJECT LOADMAP mapname

# Compulsory Attributes
PROJECT projectname
FORMAT formatname
DATABASE targetdatabasename
TABLE targettablename

#Optional Attributes
BEGIN SEQUENCE
COLUMNNAME columnname
FIELDNAME fieldname
JUSTIFICATION justification
CASECONVERT caseconversion
DEFAULTVALUE defaultvalue
TRANSFERBYTES byte_transfer
COLUMNOFFSET column_offset
FIELDOFFSET field_offset
FIELDMINIMUM field_minimum
FIELDMAXIMUM field_maximum
FILLCHARACTER fillcharacter
PICTURE picture
FUNCTION record_function
STORAGECODING storage_format
BLOBCOLUMN blob_columnname
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>blob_columnname</i>	The column that contains the name of the file where BYTE or TEXT data is stored See “Simple LO Data in a Separate File” on page 7-18.
<i>byte_transfer</i>	Number of bytes to transfer from record field to database column
<i>caseconversion</i>	Enter UPPER for all uppercase data, LOWER for all lowercase data, and PROPER for data with an initial capital letter
<i>columnname</i>	Name of the column to be mapped

(1 of 2)

Attribute	Attribute Value
<i>column_offset</i>	Amount of offset from the beginning of the column to the location on the column from which data transfer begins
<i>defaultvalue</i>	Value when no field is mapped to the column
<i>fieldname</i>	Field corresponding to the record format to be mapped
<i>field_maximum</i>	Largest acceptable numeric-column value
<i>field_minimum</i>	Smallest acceptable numeric-column value
<i>field_offset</i>	Amount of offset from the start of the field record to the location in the record from which data transfer begins
<i>fillcharacter</i>	Character used to pad contents of a field
<i>formatname</i>	Associated format name You must create the format; onpladm does not create it for you.
<i>justification</i>	Enter LEFT, RIGHT, or CENTER to position text within a record
<i>mapname</i>	Map name
<i>picture</i>	Reformats and masks data from the field of a record before data is transferred to database
<i>projectname</i>	Name of existing project
<i>record_function</i>	User-defined function in a shared library that is called for every record that is processed See Appendix E, “Custom Conversion Functions.”
<i>storage_format</i>	The format in which to store BYTE or TEXT data
<i>targetdatabasename</i>	Name of the database that the records will be loaded and unloaded to
<i>targettablename</i>	Target-table name

(2 of 2)

Use the following syntax to create an unload map with a specification file:

```
BEGIN OBJECT UNLOADMAP mapname

# Compulsory Attributes
PROJECT projectname
FORMAT formatname
DATABASE targetdatabasename
QUERY queryname

# Optional Attributes
BEGIN SEQUENCE
COLUMNNAME columnname
FIELDNAME fieldname
JUSTIFICATION justification
CASECONVERT caseconversion
DEFAULTVALUE defaultvalue
TRANSFERBYTES byte_transfer
COLUMNOFFSET column_offset
FIELDOFFSET field_offset
FIELDMINIMUM field_minimum
FIELDMAXIMUM field_maximum
FILLCHARACTER fillcharacter
PICTURE picture
FUNCTION record_function
STORAGECODING storage_format
BLOBCOLUMN blob_columnname
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>blob_columnname</i>	The column that contains the name of the file where BYTE or TEXT data is stored See “Simple LO Data in a Separate File” on page 7-18.
<i>byte_transfer</i>	Number of bytes to transfer from record field to database column
<i>caseconversion</i>	Enter UPPER for all uppercase data, LOWER for all lowercase data, and PROPER for data with an initial capital letter
<i>columnname</i>	Name of the column to be mapped

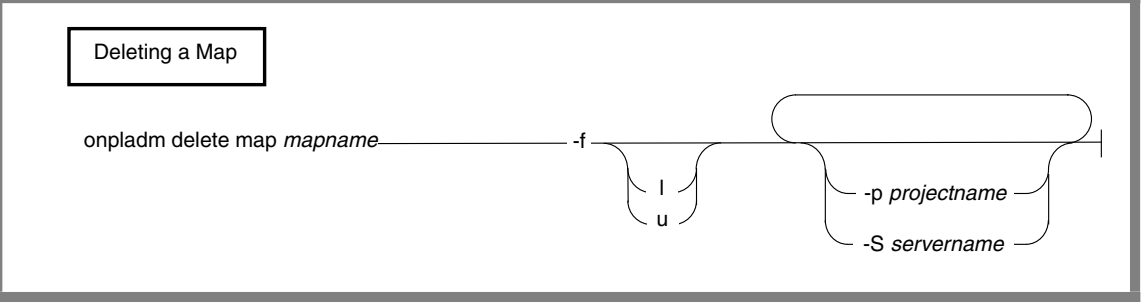
(1 of 2)

Attribute	Attribute Value
<i>column_offset</i>	Amount of offset from the beginning of the column to the location on the column from which data transfer begins
<i>defaultvalue</i>	Value when no field is mapped to the column
<i>fieldname</i>	Field corresponding to the record format to be mapped
<i>field_maximum</i>	Largest acceptable numeric-column value
<i>field_minimum</i>	Smallest acceptable numeric-column value
<i>field_offset</i>	Amount of offset from the start of the field record to the location in the record from which data transfer begins
<i>fillcharacter</i>	Character used to pad contents of a field
<i>formatname</i>	Associated format name
<i>justification</i>	Enter LEFT, RIGHT, or CENTER to position text within a record
<i>mapname</i>	Map name
<i>picture</i>	Reformats and masks data from the field of a record before data is transferred to database
<i>projectname</i>	Name of existing project
<i>queryname</i>	Query name
<i>record_function</i>	User-defined function in dynamically linked library that is called for every record that is processed See Appendix E, “Custom Conversion Functions.”
<i>storage_format</i>	The format in which to store BYTE or TEXT data
<i>targetdatabasename</i>	Name of the database that the records will be loaded and unloaded to

(2 of 2)

Deleting a Map

The following diagram illustrates the syntax to delete a map.

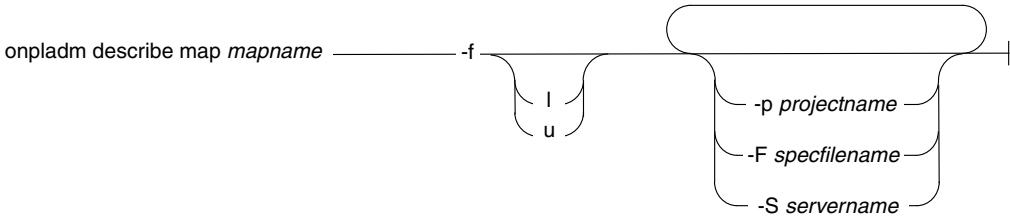


Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
<i>mapname</i>	Sets the map name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Describing a Map

The following diagram illustrates the syntax to describe a map.

Describing a Map



Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>mapname</i>	Sets the map name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file used to describe a map, see [“Creating a Map With a Detailed Specification File” on page 17-32.](#)

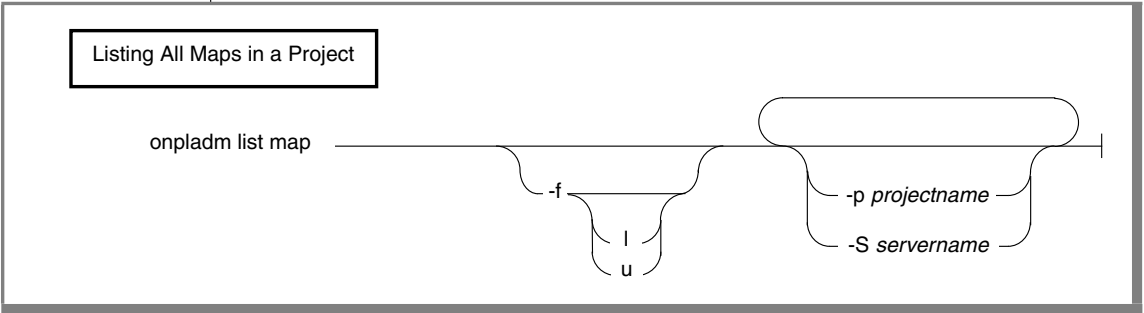
Modifying a Map Using A Detailed Specification File

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify an existing map with a specification file.

For information about the map-specification file, refer to [“Creating a Map With a Detailed Specification File” on page 17-32.](#)

Listing All Maps in a Project

The following diagram illustrates the syntax to list all the maps in a project.



Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Formats

You can create, modify, describe, list, and delete formats with the **onpladm** utility. For more information, see [“Formats” on page 7-3](#).

Creating a Format

You can only create formats by using a detailed specification file.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a format by using a detailed specification file.

Use the following syntax to create a fixed-format object from a specification file:

```
BEGIN OBJECT FIXEDFORMAT formatname

# Compulsory Attributes
PROJECT projectname
CHARACTERSET data_codeset
MACHINE machine_type
BEGIN SEQUENCE
  FIELDNAME fieldname
  DATATYPE datatype
  BYTES field_bytes
  DECIMALS decimal_places
  OFFSET offset_bytes
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>data_codeset</i>	Code set used to translate data in the data table For more information on data code sets, see the following file on your product CD: \$INFORMIXDIR/gls/cm3/registry
<i>datatype</i>	Type of field data See “Data Types Allowed in a Fixed Format” on page 7-8.
<i>decimal_places</i>	Number of decimal places for Float and Double data types
<i>field_bytes</i>	Number of bytes that the field occupies in the record
<i>fieldname</i>	Field name in the format
<i>formatname</i>	Format name

(1 of 2)

Attribute	Attribute Value
<i>machine_type</i>	Type of computer that produced the data, such as a SPARCstation See “Listing all Existing Machine Types” on page 17-61.
<i>offset_bytes</i>	Number of bytes of the field offset in the record
<i>projectname</i>	Name of existing project

(2 of 2)

Use the following syntax to create a COBOL-format object from a specification file:

```

BEGIN OBJECT COBOLFORMAT formatname

# Compulsory Attributes
PROJECT projectname
CHARACTERSET data_codeset
MACHINE machine_type
DRIVER driver_type
BEGIN SEQUENCE
FIELDNAME fieldname
PICTURE picture_description
USAGE usage_description
END SEQUENCE

END OBJECT

```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>data_codeset</i>	Code set used to translate data in the data table For more information on data code sets, see the following file on your product CD: \$INFORMIXDIR/gls/cm3/register
<i>driver_type</i>	Type of driver: COBOL (default) or COBOL_b
<i>fieldname</i>	Field name in the format
<i>formatname</i>	Format name
<i>machine_type</i>	Type of computer that produced the data, such as a SPARCstation

(1 of 2)

Attribute	Attribute Value
<i>picture_description</i>	Picture description that matches the record FD from the COBOL program See Appendix C, “Picture Strings.”
<i>projectname</i>	Name of existing project
<i>usage_description</i>	Number of bytes that the field occupies in the record

(2 of 2)

Use the following syntax to create a delimited-format object:

```
BEGIN OBJECT DELIMITEDFORMAT formatname

# Compulsory Attributes
PROJECT projectname
CHARACTERSET data_codeset
RECORDSTART recordstart_delimit_character
RECORDEND recordend_delimit_character
FIELDSTART fieldstart_delimit_character
FIELDEND fieldend_delimit_character
FIELDSEPARATOR fieldseparator_delimit_character
BEGIN SEQUENCE
FIELDNAME format_fieldname
FIELDTYPE field_datatype
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>data_codeset</i>	Code set used to translate data in the data table For more information on data code sets, see the following file on your product CD: \$INFORMIXDIR/gls/cm3/registry
<i>field_datatype</i>	Type of field data See “Data Types Allowed in a Delimited Format” on page 7-20.
<i>fieldend_delimit_character</i>	Delimiting character that specifies the end of the field, in hexadecimal or decimal format Begin a hexadecimal character with 0x.

(1 of 2)

Attribute	Attribute Value
<i>fieldstart_delimit_character</i>	Delimiting character that specifies the start of the field, in hexadecimal or decimal format Begin a hexadecimal character with 0x.
<i>fieldseparator_delimit_character</i>	Delimiting character that specifies the field separator, in hexadecimal or decimal format Begin a hexadecimal character with 0x.
<i>format_fieldname</i>	Format field name
<i>formatname</i>	Format name
<i>recordend_delimit_character</i>	Delimiting character that specifies the end of the record, in hexadecimal or decimal format Begin a hexadecimal character with 0x.
<i>recordstart_delimit_character</i>	Delimiting character that specifies the start of the record, in hexadecimal or decimal format Begin a hexadecimal character with 0x.
<i>projectname</i>	Name of existing project

(2 of 2)

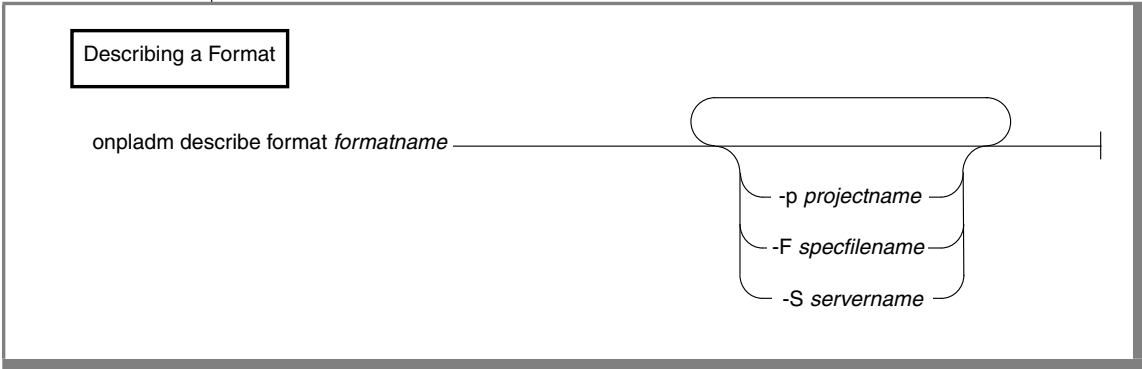
Modifying a Format Using a Specification File

You can only modify a format using a specification file. Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify a format.

For more information on the corresponding specification-file format, see [“Creating a Format” on page 17-39](#).

Describing a Format

The following diagram illustrates the syntax to describe a format to a file.



Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>formatname</i>	Sets the format name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file used to describe a format, see [“Creating a Format” on page 17-39](#).

Listing all Formats in a Project

The following diagram illustrates the syntax to list all formats in a project to standard output.

Listing All Formats in a Project



Element	Purpose	Key Considerations
<code>-p projectname</code>	Identifies the project where the format and map are stored	None
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Format

The following diagram illustrates the syntax to delete a format.

Deleting All Formats in a Project

onpladm delete format *formatname*

-p *projectname*

-S *servername*

Element	Purpose	Key Considerations
<i>formatname</i>	Sets the format name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Queries

You can create, modify, describe, list, and delete queries with the **onpladm** utility.

Creating a Query

You can only create a query by using a detailed specification file.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a query.

Use the following syntax to create specification file for a query:

```
BEGIN OBJECT QUERY queryname
# Compulsory Attributes
PROJECT projectname
DATABASE targetdatabasename
SELECTSTATEMENT sql_statement

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>queryname</i>	Query name
<i>projectname</i>	Name of an existing project
<i>sql_statement</i>	SQL SELECT statement, in quotation marks, that contains unload criteria
<i>targetdatabasename</i>	Name of the database that the records will be loaded or unloaded to

Modifying a Query

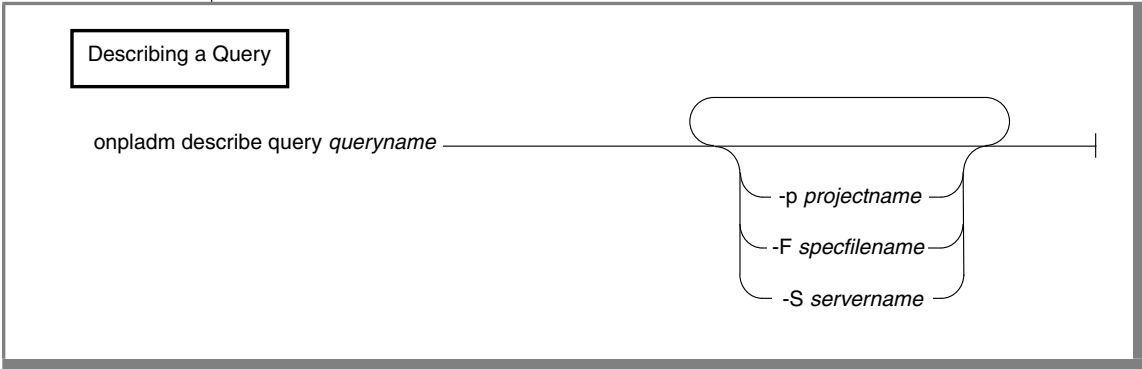
You can only modify a query by using a detailed specification file. For information about the query specification file, refer to [“Creating a Query” on page 17-46](#).

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify a query by using a specification file.

For more information on the corresponding specification-file format to modify a query, see [“Creating a Query” on page 17-46](#).

Describing a Query

The following diagram illustrates the syntax to describe a query.



Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>queryname</i>	Sets the query name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the format of the specification file used to describe a query, see [“Creating a Query” on page 17-46](#).

Listing all Queries in a Project

The following diagram illustrates the syntax to list all queries in a project to standard output.

Listing All Queries in a Project

onpladm list query

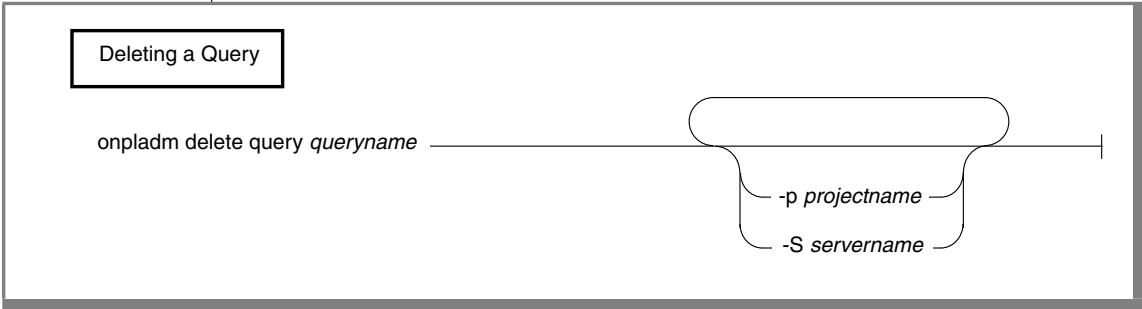
-p *projectname*

-S *servername*

Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Query

The following diagram illustrates the syntax to delete a query.



Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
<i>queryname</i>	Sets the name of the query	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Filters

You can create, modify, describe, list, and delete filters with the **onpladm** utility. For more information, see [“Using a Filter” on page 10-4](#).

Creating a Filter

You can only create a filter by using a specification file.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a filter by using a specification file.

Use the following syntax to create a filter by using a specification file:

```
BEGIN OBJECT FILTER filtername
# Compulsory Attributes
PROJECT projectname
FORMAT formatname
BEGIN SEQUENCE
FIELDNAME data_file_fieldname
STATUS record_status
MATCH match_criteria
END SEQUENCE

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>data_file_fieldname</i>	Data-file field to be used in the match condition
<i>formatname</i>	Associated format name
<i>filtername</i>	Filter name
<i>match_criteria</i>	Match criteria, in quotation marks. See Appendix D, “Match Condition Operators and Characters.”
<i>projectname</i>	Name of existing project
<i>record_status</i>	Type a K to keep records that meet a match condition or D to discard them.

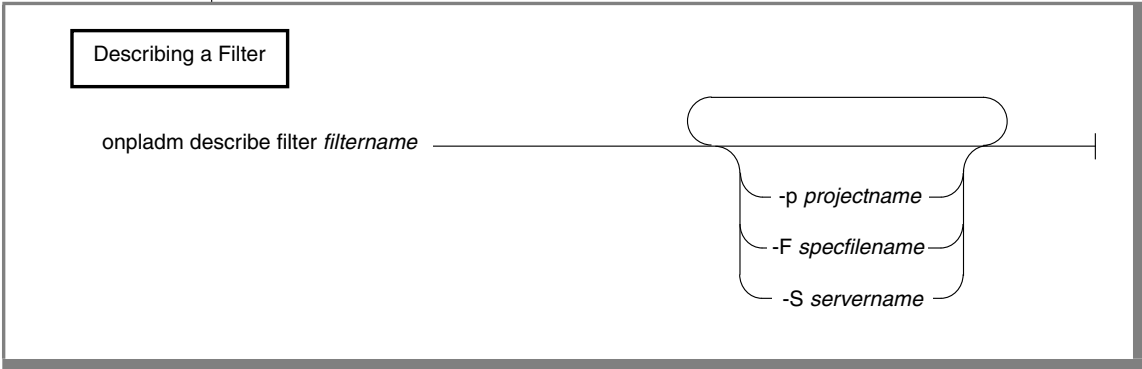
Modifying a Filter

You can only modify a filter by using a detailed specification file. For information about the filter specification file, see [“Creating a Filter” on page 17-50](#).

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify a filter.

Describing a Filter

The following diagram illustrates the syntax to describe a filter.



Listing All Filters in a Project

onpladm list filter

-p *projectname*

-S *servername*

Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Filter

The following diagram illustrates the syntax to delete a filter.

Deleting a Filter

onpladm delete filter *filtername*

-p *projectname*

-S *servername*

Element	Purpose	Key Considerations
<i>filtername</i>	Sets the name of the filter	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Projects

You can create a project, run all jobs in a project, list all projects, and delete a project with the **onpladm** utility. For more information, see [“Project Organization” on page 4-3](#).

Creating a New Project

The following diagram illustrates the syntax to create a new, empty project to which you can add the HPL jobs.

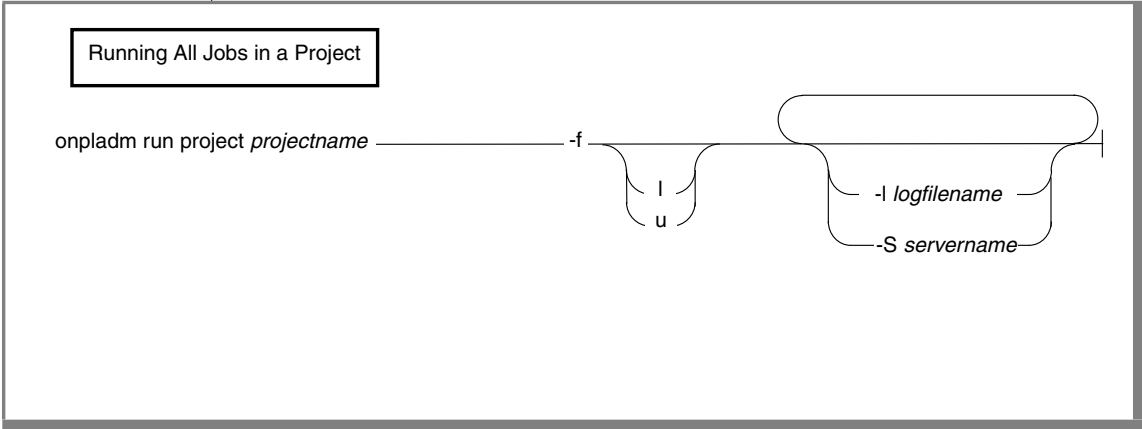
Creating a Project



Element	Purpose	Key Considerations
<code>-p projectname</code>	Identifies the project where the format and map are stored	None
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Running All Jobs in a Project

The following diagram illustrates the syntax to run all load or unload jobs in a project.



Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
-l <i>logfilename</i>	Sets the path for a log file where job progress is recorded	None
<i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information on how to execute this command for a database server running on Windows, see [“Using the onpladm Utility on Windows” on page I-2](#).

Listing all Projects

The following diagram illustrates the syntax to list all your projects.

Listing All Projects

onpladm list project _____
-S *servername*

Element	Purpose	Key Considerations
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Project

The following diagram illustrates the syntax to delete a project, all of its corresponding jobs, and other HPL objects that it holds.

Deleting a Project

onpladm delete project *projectname* _____
-S *servername*

Element	Purpose	Key Considerations
<i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Machine Types

You can create, modify, describe, list, and delete machine types with the **onpladm** utility.

Creating a Machine Type

You can only create a machine type by using a specification file.

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to create a machine type.

Use the following syntax to create a machine object by using a specification file:

```
BEGIN OBJECT MACHINE machinename
# Compulsory Attributes
BYTEORDER machinetype_byteorder
SHORTSIZE shortinteger_bytes
INTERSIZE integer_bytes
LONGSIZE longinteger_bytes
FLOATSIZE float_bytes
DOUBLESIZE double_bytes

END OBJECT
```

The following table lists the attributes and their values.

Attribute	Attribute Value
<i>double_bytes</i>	Double integer size
<i>float_bytes</i>	Float size
<i>integer_bytes</i>	Integer size
<i>machinename</i>	Machine name

(1 of 2)

Attribute	Attribute Value
<i>longinteger_bytes</i>	Long integer size
<i>machinetype_byteorder</i>	Computer byte-order type Enter MSB for most-significant bit or LSB for least-significant bit.
<i>shortinteger_bytes</i>	Short integer size

(2 of 2)

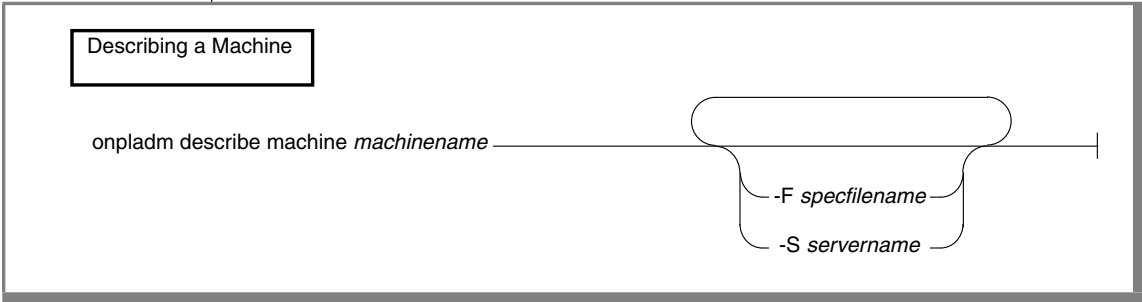
Modifying a Machine Type

You can only modify a machine type by using a detailed specification file. For information on the machine type specification file, refer to [“Creating a Machine Type” on page 17-58](#).

Use the syntax shown in [“Specification-File Conventions” on page 17-6](#) to modify a machine type.

Describing a Machine

The following diagram illustrates the syntax to describe a machine.



Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>machinename</i>	Sets the machine name	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file used to describe a machine type, see [“Creating a Machine Type” on page 17-58](#).

Listing all Existing Machine Types

The following diagram illustrates the syntax to list the machine types.

Listing All Machines



Element	Purpose	Key Considerations
<i>-S servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a Machine Type

You can only delete a machine type by using a specification file.

The following diagram illustrates the syntax to delete a machine type.

Deleting a Machine



Element	Purpose	Key Considerations
<i>machinename</i>	Sets the machine type	None
<i>-S servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Defining Database Operations

You can perform the following database operations with the **onpladm** utility:

- Create load and unload jobs for all tables in a database with a single command.
- Load or unload all tables in a database.

Creating a Database Project

When you create a database project with a single command, the **onpladm** utility:

- Creates all load or unload jobs for every table in a database
- Creates all the associated project HPL objects required for the jobs
- Groups the load or unload jobs and associated project HPL jobs into a single project that has the same name as the database

When you create a database project, you must specify the data-files source directory name or the tape-device path.

If you specify the data-files source directory, the files that the **onpladm** utility creates have the following format:

```
PREFIX_DATABASE_TABLE.unl
```

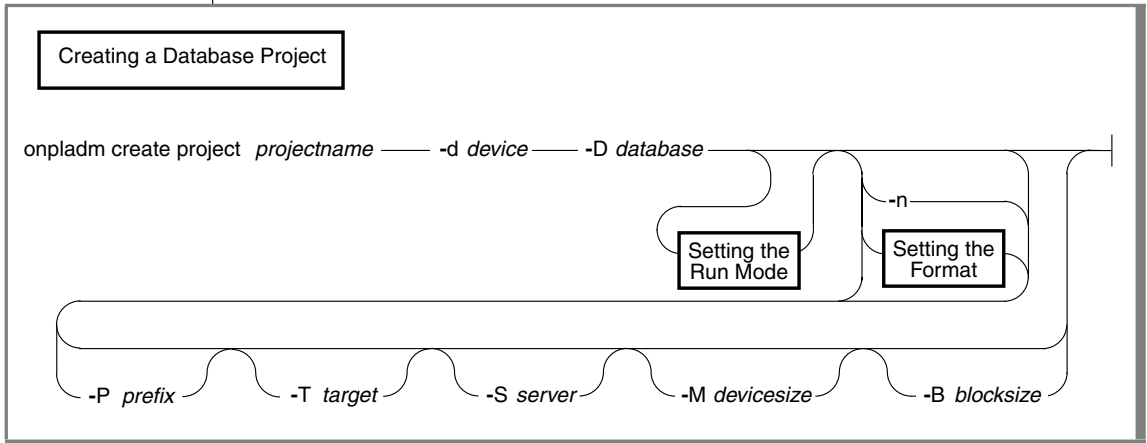
PREFIX is an option that you specify on the command line, *DATABASE* is the name of the target database, and *TABLE* is the name of the target-table name.

If you do not specify a prefix, the **onpladm** utility creates files of the following format:

```
DATABASE_TABLE.unl
```

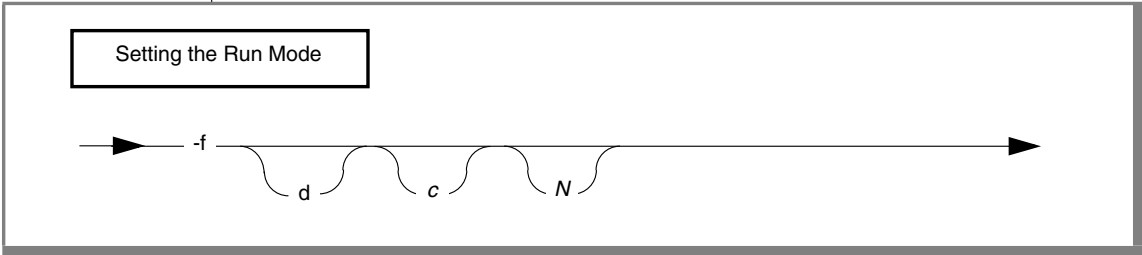
The **onpladm** utility truncates the format filename if it is longer than the maximum filename length of 18 characters.

The following diagram illustrates the syntax to create a project for all the tables in a database.



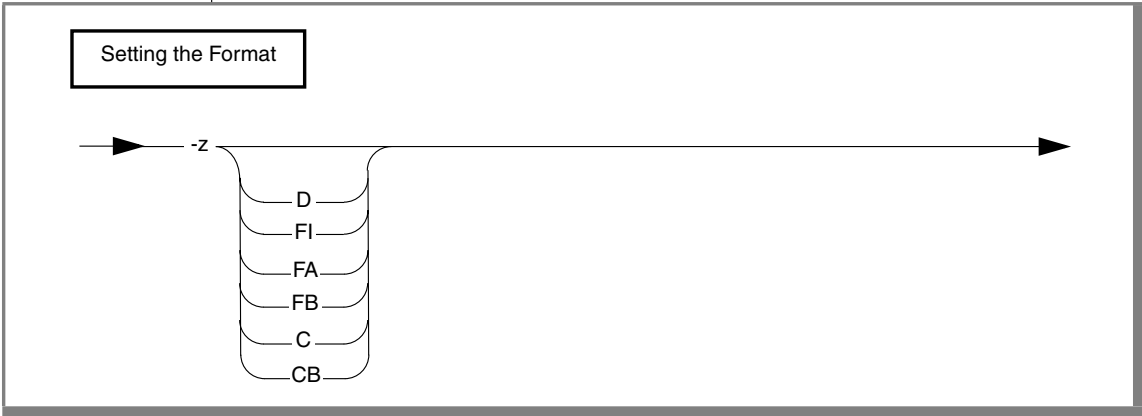
Element	Purpose	Key Considerations
-B <i>blocksize</i>	Sets the tape I/O block size (bytes)	Additional Information: No default value.
-d <i>device</i>	Sets the name of device, such as a file, device array, tape, or pipe	Additional Information: No default value.
-D <i>database</i>	Name of the target database that contains the information to be loaded or unloaded	Additional Information: No default value.
-M <i>devicesize</i>	Tape device size in kilobytes	Additional Information: The device size must be greater than zero.
-n	Sets no-conversion express job	None
-P <i>prefix</i>	Filename prefix for the files to be used as devices	None
<i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>server</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.
-T <i>target</i>	Name of the target server to which the data will download	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

The following diagram illustrates the syntax to set the run mode with the **-f** option.



Element	Purpose	Key Considerations
c	Sets mode to deluxe mode.	Additional Information: If this flag is not set, onpladm uses express mode.
d	Treats data source as a tape device.	None
N	Allows deluxe mode load without replication	None

The following diagram illustrates the syntax to set the format type with the **-z** option.



Element	Purpose	Key Considerations
D	Sets the format to delimited.	Additional Information: This is the default value. See “Delimited Records” on page 7-19.
FI	Sets the format to fixed internal.	Additional Information: See “Fixed-Length Records” on page 7-4.
FA	Sets the format to fixed ASCII.	Additional Information: See “Fixed-Length Records” on page 7-4.
FB	Sets the format to fixed binary.	Additional Information: See “Fixed-Length Records” on page 7-4.
C	Sets the format to COBOL.	Additional Information: See “COBOL Records” on page 7-24.
CB	Sets the format to COBOL (byte).	Additional Information: See “COBOL Records” on page 7-24.

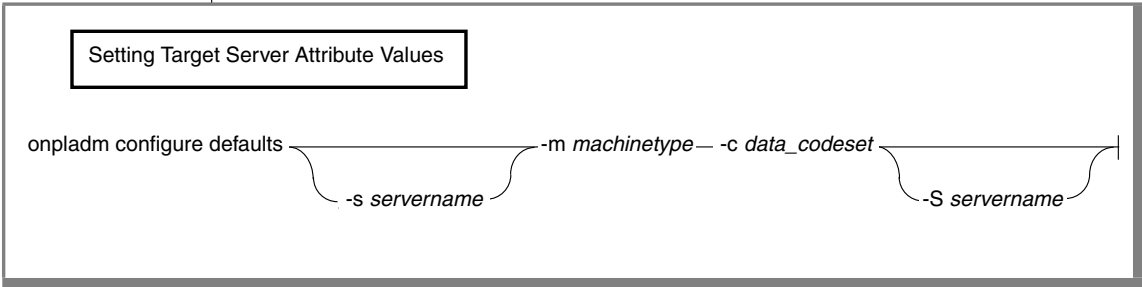
Configuring Target-Server Attributes

To configure target-server attributes, you must:

- Set target-server attribute values.
- List target-server defaults.

Setting Target-Server Attribute Values

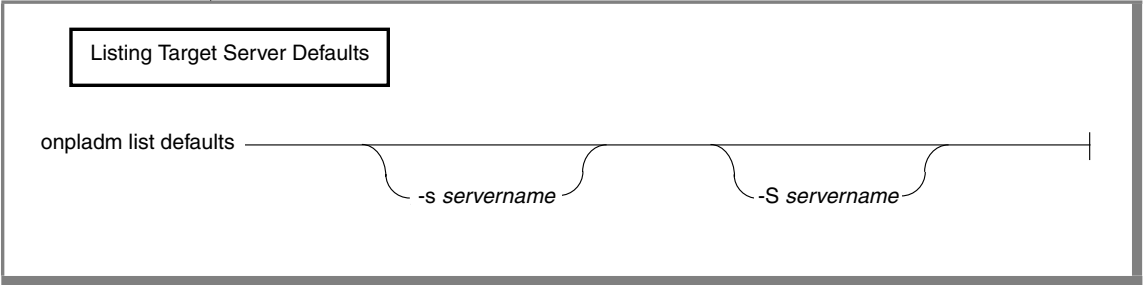
The following diagram illustrates the syntax to set default values for target server attributes.



Element	Purpose	Key Considerations
-c data_codeset	Character set for data files	Additional Information: The character set of the database is determined by the DB_LOCALE environment variable. For information about locales and code sets, see the <i>IBM Informix GLS User's Guide</i> .
-m machinetype	Machine type	None
-s servername	Database server for which defaults are set	Additional Information: Default server row entry.
-S servername	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Listing Target-Server Defaults

The following diagram illustrates the syntax to list target-server default values.



Element	Purpose	Key Considerations
-s servername	Database server for which defaults are set	Additional Information: Default server row entry.
-S servername	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

onpload Database

The tables in the **onpload** database hold information that the **onpload** utility uses. This appendix describes the tables in the **onpload** database that you create or modify with **ipload**.

When you start **ipload**, **ipload** looks for a database named **onpload** on the database server that your **INFORMIXSERVER** environment variable specifies. If the **onpload** database is not present, **ipload** creates an **onpload** database as a non-ANSI database.

When **ipload** creates a new **onpload** database, it populates some of the tables in the database with default values. You can use DB-Access to *examine* the values in the tables. However, it is strongly recommended that you always use **ipload** to *change* the **onpload** database.

Defaults Table

The **defaults** table contains default values that the HPL uses. When **ipload** creates the **onpload** database, it inserts a single row into this table. This row specifies the default configuration assumptions for the database server, the type of computer, and the data code set.

Column	Type	Description
node	CHAR(18)	The name of a database server
machine	CHAR(18)	Specifies the default machine type (foreign key to the machines table)
datatype	CHAR(18)	The code set of the data file
dbgls	CHAR(18)	Reserved Used previously for the code set of the target database

You can specify a set of defaults for each database server. If this table does not contain an entry for a database server, the database uses the defaults that the record named **default** specifies.

Use the Defaults window to modify this table. For more information, see [“Modifying the onpload Defaults” on page 5-5](#).

Delimiters Table

The **ipload** utility uses the values in the **delimiters** table to display the field-delimiter values that the Delimiter Options window shows. When **ipload** creates the **onpload** database, it inserts values into this table. The values in the **delimiters** table are for reference and do not change. For more information, see [“Modifying Delimited-Format Options” on page 7-30](#) and [“The Delimiter Options Window” on page 7-31](#).

Column	Type	Description
hex	CHAR(2)	Hexadecimal representation of the delimiter
octal	CHAR(4)	Octal representation of the delimiter
ascii	CHAR(15)	ASCII characters (printable) that form the delimiter
control	CHAR(10)	Control character sequence that generates the delimiter

Device Table

The **device** table defines the elements of a device array. Use the device array definition window to modify this table. For more information, see [“Using the Device-Array Definition Window” on page 6-6](#).

Column	Type	Description
name	CHAR(18)	Name of the device array described in this row (primary key)
seq	INTEGER	Device number within the device array (primary key)
type	CHAR(5)	Device type (pipe (only on UNIX), file, or tape)
file	CHAR(128)	File or device to be accessed by this array element

(1 of 2)

Column	Type	Description
blocksize	INTEGER	I/O blocksize (tape devices only)
devicesize	INTEGER	Capacity of device (tape devices only)
pipecommand	CHAR(128)	The pipe command to invoke when onpload starts to access to the device element (only on UNIX)
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses
header	TEXT	The tape header for a device that DDR uses

(2 of 2)

Driver Table

The **onpload** utility uses different set routines, called *drivers*, to handle different file formats. For example, the delimited driver handles delimited file formats. The routines in a driver process data unloaded from or loaded into the data file. The **onpload** utility includes drivers for widely used data-file formats. You can prepare additional, custom drivers for other formats and bind them into the **onpload** shared library. The set of available drivers is stored in the **driver** table.

Column	Type	Description
drivername	CHAR(18)	Name of driver (primary key)
drivertype	CHAR(1)	Data-file format: Fixed, Delimited, COBOL

You can use the procedure that Appendix H describes to build custom drivers. A custom driver takes data from a file and constructs input for **onpload** that is in a format that **onpload** recognizes (Fixed, Delimited, COBOL).

To add custom drivers to the **driver** table, see [“Drivers Window” on page 5-9](#).

Filteritem Table

The **filteritem** table defines the conditions to be applied to load data to filter out records. Each filter item is attached to a particular field of a record in a data file. Use the filter options to modify this table. For more information, see [“Creating a Filter” on page 10-5](#).

Column	Type	Description
formid	INTEGER	Filter identifier (foreign key to the filters table)
seq	INTEGER	Specifies the order in which the filter items (the match expression) are applied
fname	CHAR(18)	The name of the field that this filter affects
option	CHAR(7)	Specifies the disposition of a record (discard or keep) when the match criterion is true
match	CHAR(60)	Match expression that is applied to data field

Filters Table

The **filters** table assigns a unique number to each group of filter items that together form a filter. Each filter is associated with a project and a format definition. Use the filter-definition window to create or modify a filter. For more information, see [“The Filter-Definition Window” on page 10-7](#).

Column	Type	Description
formid	SERIAL	Filter identifier (primary key)
projectid	INTEGER	Project with which this filter is associated (foreign key to the project table)
formatid	INTEGER	Format identifier of the format definition to which this filter applies (foreign key to the formats table)
name	CHAR(18)	The name of the filter
lockflag	CHAR(1)	Flag for locking mechanism used by ipload

Formatitem Table

The **formatitem** table defines the data-file records. Each field of a data file is described by an entry in this table. Use the Records Format window to prepare the record formats. [Figure A-1](#) lists the possible values for the **ftype** column. For more information, see [“A Completed Fixed-Format Definition Window with an Open Selection List” on page 7-6](#).

Column	Type	Description
formid	INTEGER	Record format identifier (foreign key to the formats table)
seq	INTEGER	Item sequence number for internal organization
fname	CHAR(18)	Name of record field

(1 of 2)

Column	Type	Description
ftype	INTEGER	A number that indicates the type of data in the field
bytes	INTEGER	Number of bytes in field
decimals	INTEGER	Number of decimal values to format when converting to ASCII
offset	INTEGER	Offset in record image where field starts
qual	INTEGER	Informix DATETIME /INTERVAL qualifier
picture	CHAR(15)	COBOL picture definition

(2 of 2)

Figure A-1
Possible Values for the ftype Column

ftype Value	Type of Data	ftype Value	Type of Data
1	Character (fixed and delimited)	23	Comp-4
2	Date	24	Comp-5
3	Short integer	25	Comp-6
4	Integer	26	Comp-X
5	Long Integer	27	Comp-N
6	Floating-point vale	28	Character (COBOL)
7	Double floating-point value	34	Blob Length
8	Unsigned short integer	35	Blob File
9	Unsigned integer	36	Blob HexASCII
10	Unsigned long integer	37	Blob Text
11	UNIX date		Int8
18	Packed Decimal		Ext Type String
19	Zoned decimal		Ext Type HexASCII

(1 of 2)

ftype Value	Type of Data	ftype Value	Type of Data
20	Comp-1		Ext Type Binary
21	Comp-2		Ext Type StringLength
22	Comp-3		Ext Type BinaryLength

(2 of 2)

Formats Table

The **formats** table defines the basic information for a record format. Use the Records Format window to modify this table. For more information, see [“The Record Formats Window” on page 7-5](#).

Column	Type	Description
formid	SERIAL	Unique format identifier (primary key)
projectid	INTEGER	Project to which the format is assigned (foreign key to the project table)
name	CHAR(18)	Name of format
type	CHAR(10)	Data-file format: Fixed, Delimited, COBOL
driver	CHAR(18)	Driver to use to access data records
machine	CHAR(18)	Machine name that defines binary-data parameters (foreign key to the machinename column of the machines table)
datatype	CHAR(18)	Character code set to use for conversion of data records
recordlength	INTEGER	Length in bytes of a fixed-format record
recordstrt	CHAR(15)	Record-start sequence for delimited format
recordstrty	CHAR(10)	Type of the record-start sequence: Hex, Octal, ASCII, or Decimal

(1 of 2)

Column	Type	Description
recordend	CHAR(15)	Record-end sequence for delimited format
recordendt	CHAR(10)	Type of the record-end sequence: Hex, Octal, ASCII, or Decimal
fieldsep	CHAR(15)	Field-separator sequence for delimited format
fieldsept	CHAR(10)	Type of the field-separator sequence: Hex, Octal, ASCII, or Decimal
fieldstrt	CHAR(15)	Field-start sequence for delimited format
fieldstrty	CHAR(10)	Type of the field-start sequence: Hex, Octal, ASCII, or Decimal
fieldend	CHAR(15)	Field-end sequence for delimited format
fieldendt	CHAR(10)	Record-end sequence separator type: Hex, Octal, ASCII, or Decimal
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses

(2 of 2)

Language Table

The **onpload** utility does not use the **language** table at this time.

Machines Table

The **machines** table defines the binary type sizes and byte order for different computers. The HPL uses this information when you transfer binary data. When **ipload** creates the **onpload** database, it inserts definitions for several different types of computers into this table. To transfer binary data to or from a computer that is not described in this table, you must create a new machine definition using the Machines window. For more information, see [“Machines Window” on page 5-12](#).

Column	Type	Description
machinename	CHAR(18)	Computer name or type (primary key)
byteorder	CHAR(3)	Binary byte ordering: LSB or MSB
shortsize	INTEGER	Size of a short integer
intsize	INTEGER	Size of an integer
longsize	INTEGER	Size of a long integer
floatsize	INTEGER	Size of a float value
doublesize	INTEGER	Size of a double value

Mapitem Table

The **mapitem** table defines the relationship between the columns of a database table and the record fields of a data file. The table stores pairs of column/record entries. The map options modify this table. For more information, see [“Load and Unload Maps” on page 9-3](#).

Column	Type	Description
formid	INTEGER	Specifies the map to which this record belongs (foreign key to the maps table)
seq	INTEGER	Unique identifier for the database-column/data file-record pair
colname	CHAR(18)	Name of database column
fname	CHAR(18)	Name of field in a data-file record

Mapoption Table

The **mapoption** table defines conversion options for the mapping pairs that are defined in **mapitem** table. Use the Mapping Options window to modify this table. For more information, see [“Using Mapping Options” on page 9-15](#).

Column	Type	Description
formid	INTEGER	Specifies the map to which this record belongs (foreign key to the maps table)
seq	INTEGER	The database-column and/or data file-record pair to which this option applies (foreign key to the mapitem table)
bytes	INTEGER	Maximum number of bytes to transfer from a field of a data file
minvalue	FLOAT	Minimum value allowed in field
maxvalue	FLOAT	Maximum value allowed in field
ccase	CHAR(18)	Case conversion option: None, Lower, Upper, Proper Noun
justify	CHAR(18)	String justification to perform: None, Left, Right, Center
fill	CHAR(1)	Fill character for string padding
picture	CHAR(55)	Picture mask to apply to target data
coloffset	INTEGER	Offset in column at which to start data transfer
recoffset	INTEGER	Offset in record field from which to start data extract
function	CHAR(55)	Custom function to call
looktable	CHAR(18)	Not in use
matchcol	CHAR(18)	Not in use
coldefault	CHAR(18)	Default value to set on column: ASCII HEX or ASCII binary

(1 of 2)

Column	Type	Description
inputcode	CHAR(18)	Format in which the BYTE or TEXT data is stored in the data file: ASCII HEX or ASCII binary
storecode	CHAR(18)	Format in which to store the BYTE or TEXT data
blobcolumn	CHAR(18)	The column that contains the name of the file where the BYTE or TEXT data is stored

(2 of 2)

If the values of **inputcode** and **storecode** are different, **onpload** converts the contents of the BYTE or TEXT data.

Maps Table

The **maps** table defines record-to-table mappings (for loads) and query-to-record mappings (for unloads). Use the map options to modify this table. For more information, see [“Load and Unload Maps” on page 9-3](#).

Column	Type	Description
projectid	INTEGER	Project to which this map is assigned (foreign key to the project table)
formid	SERIAL	Unique identifier for map (primary key)
name	CHAR(18)	Name of map
type	CHAR(6)	Specifies whether the map is a load or unload map; possible values include: <ul style="list-style-type: none">■ Record (load map)■ Query (unload map)
dbname	CHAR(30)	Name of load or unload database
qtable	CHAR(18)	Name of table to be loaded; used only for loads

(1 of 2)

Column	Type	Description
query	CHAR(18)	Name of query; used only for unloads
formatid	INTEGER	Identifier of the format that this map uses (foreign key to the format table)
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses

(2 of 2)

Note Table

The **note** table holds comments that you can store about the components that are used for loads and unloads. You can store notes about all of the **onpload** components: projects, devices, formats, maps, queries, filters, and load and unload jobs. For information about creating a note, see [“Notes Button” on page 3-24](#).

Column	Type	Description
type	CHAR(18)	Specifies the type of component to which this note is attached
formid	INTEGER	Corresponds to the formid of the component specified in the type column (The two columns together uniquely identify the component to which the note is attached.)
projectid	INTEGER	ID of project to which this note belongs (foreign key to the project table)
createdate	DATE	Date that the note was created
modifydate	DATE	Date that the note was last modified
note	TEXT	Text of the note

Project Table

The **project** table lists the projects in this **onpload** database. Use the Project window to modify this table. For more information, see [“Project Organization” on page 4-3](#).

Column	Type	Description
name	CHAR(18)	Name of object
projectid	SERIAL	Uniquely identifies the project (primary key)
dcreate	DATE	Date that the project was created

Query Table

The **query** table stores the queries that are used for unloading data from an Informix database. Use the query-definition window to modify this table. For more information, see [“Creating a Query” on page 8-4](#).

Column	Type	Description
formid	SERIAL	Unique number that identifies this query (primary key)
projectid	INTEGER	Number of the project that includes this query (foreign key to the projects table)
name	CHAR(18)	Name of the query
database	CHAR(30)	Name of database being queried
arrayname	CHAR(18)	Not in use
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses
sqlselect	TEXT	SQL statement of the query

Session Table

The **session** table controls the parameters that **onpload** uses to invoke a load or unload job.

Column	Type	Description
sessiontype	CHAR(1)	Describes the type of load or unload session: U = Job is driven by the user interface. N = Job expects a socket interface and is removed when the job is finished. S = Job is run from the command line.
automate	CHAR(1)	Flag for automatically creating maps and formats at runtime: Y = Create automatically. blank = Do not create.
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses
sessionid	SERIAL	Session identifier (primary key)
name	CHAR(20)	Name of the load or unload job. This name appears in the command line displayed in the Load Job Select or Unload Job Select window.
status	CHAR(1)	Job status: R = running C = connecting S = starting blank = Job is complete.
server	CHAR(40)	Override default server to load and unload
map	CHAR(18)	Name of the map that controls the load (foreign key to the name column of the maps table; the maps table specifies the format and, for unload jobs, the query)

(1 of 4)

Column	Type	Description
infile	CHAR(160)	Name of the device array (foreign key to the name column of the device table)
hostname	CHAR(40)	Name of the computer on which the onpload utility is running
dbname	CHAR(30)	Name of database to be loaded or unloaded
filter	CHAR(18)	Filter for screening import data (foreign key to the name column of the filters table)
recordfilter	CHAR(80)	File in which to store filtered records
suspensefile	CHAR(80)	File in which to store records that do not pass conversion
rejectfile	CHAR(80)	File in which to place records that the database server rejected
logfile	CHAR(80)	File in which to place session status messages
projectid	INTEGER	Project for maps and formats (foreign key to the project table)
headersize	INTEGER	Size in bytes of header information to strip from input
quiet	INTEGER	If true, suppresses status message output
tracelevel	INTEGER	Higher values result in more status messages
sourcetrace	INTEGER	If true, source and module line numbers are placed in status message outputs
multithread	INTEGER	Sets the maximum number of conversion threads that you can invoke on a device
blocksize	INTEGER	I/O blocksize for accessing device
filetype	INTEGER	Specifies the type of file: tape, array, pipe (only on UNIX)
number_records	INTEGER	Specifies the number of records to load
start_record	INTEGER	Specifies the number of the record at which to start loading

(2 of 4)

Column	Type	Description
maxerrors	INTEGER	Maximum number of errors to allow before aborting the load or unload
swapbytes	INTEGER	Specifies the number of bytes to swap (If <i>swapbytes</i> is 4, the first 4 bytes are swapped with the next 4 bytes. If blank, bytes are not swapped.)
runmode	INTEGER	<p>2 = Express-mode load, no conversion</p> <p>7 = Deluxe-mode load, without replication</p> <p>Deluxe-mode load, with conversion</p> <p>129 = Express-mode load, with conversion</p> <p>130 = Express-mode load, with conversion</p> <p>386 = and do not generate violations table</p> <p>Unload, no conversion</p> <p>2 = Unload, with conversion, dirty read</p> <p>129 = isolation level</p> <p>Unload, with conversion, committed</p> <p>130 = isolation level</p> <p>Unload, with conversion, cursor</p> <p>131 = stability isolation level</p> <p>Unload, with conversion, repeatable</p> <p>132 = read isolation level</p>
loadmode	INTEGER	<p>Type of job:</p> <p>1 = load</p> <p>2 = unload</p>
caseconvert	INTEGER	<p>Case conversion type. Convert to:</p> <p>U or u = uppercase</p> <p>L or l = lowercase</p> <p>P or p = propernames</p>

(3 of 4)



Column	Type	Description
commitinterval	INTEGER	Commit interval for committing a load transaction. (The value is specified in the Load Options window, page 12-15. The commit interval applies only to deluxe mode.)
socketport	INTEGER	Set by onpload to specify the port number of the connection
numtapes	INTEGER	Number of tapes to load

(4 of 4)

Tip: Deluxe-mode loads do not support the “no conversion” and “with conversion and do not generate violations table” options.

High-Performance Loader Configuration File

The default `$INFORMIXDIR/etc/plconfig.std` file on UNIX or `%\INFORMIXDIR\etc\plconfig.std` on Windows is the *high-performance loader configuration file*. The file is similar to the `ONCONFIG` file in the `etc` directory in `INFORMIXDIR`. The `plconfig.std` file sets various **onpload** buffer and system configuration parameters. You can modify the parameters to maximize resource utilization.

The `PLCONFIG` environment variable specifies an alternative name for the HPL configuration file. This file must reside in the `etc` directory in `INFORMIXDIR`. If you do not set the `PLCONFIG` environment variable, the default name of the file is `plconfig.std`.

Configuration Parameter Descriptions

The description of each parameter has one or more of the following fields (depending on their relevance):

<i>default value</i>	The value that appears in the plconfig.std file unless you explicitly change it
<i>units</i>	The units in which the parameter is expressed
<i>range of values</i>	The possible values for this parameter
<i>refer to</i>	Cross-reference to further discussion

File Conventions

Each parameter in the **plconfig.std** file in the **etc** directory in **INFORMIXDIR** is on a separate line. The file can also contain blank lines and comment lines that start with a **#** symbol. The syntax of a parameter line is as follows:

```
PARAMETER_NAME    parameter_value    # optional comment
```

Parameters and their values are case sensitive. The parameter names are always all uppercase letters. If the parameter-value entry is described with uppercase letters, you must use uppercase. You must put white space (tabs or spaces or both) between the parameter name, parameter value, and optional comment. Do not use any tabs or spaces within a parameter value.

AIOBUFFERS

<i>default value</i>	Maximum of (4,CONVERTTHREADS)
<i>recommended value</i>	Maximum of (4, 2*CONVERTTHREADS)
<i>range of values</i>	Integer value 4
<i>refer to</i>	“Loading and Unloading Data” on page 15-14

The AIOBUFFERS parameter sets the number of buffers used to transport data from converter threads to the AIO handler.

You must set the AIOBUFFERS parameter on Windows to a minimum of 8. ♦

Windows

AIOBUFSIZE

<i>default value</i>	64
<i>units</i>	Kilobytes
<i>range of values</i>	Minimum: 0.5 kilobyte (512 bytes) Maximum: depends on operating system resources.
<i>refer to</i>	“Loading and Unloading Data” on page 15-14

The AIOBUFSIZE parameter sets the size of the AIO memory buffers that transfer data to and from tapes and files. The HPL uses the AIO buffers to pass data between the converters and the I/O drivers.

The AIOBUFSIZE parameter is not the same as the tape-block size that you can set in the device arrays (see [page 6-8](#)). The tape-block size lets you control the size of the block that the device controller sends to the tape drive, while AIOBUFSIZE lets you control the size of internal buffers that pass data. If your computer has memory available, you can improve performance by increasing the AIOBUFSIZE parameter.

CONVERTTHREADS

default value 1

range of values Minimum: 1
Maximum: depends on computer configuration

refer to [“Loading and Unloading Data” on page 15-14](#)

The CONVERTTHREADS parameter sets the number of convert threads for each file I/O device. The convert threads run on the convert VPs.

If you are doing a convert-intensive job, increasing CONVERTTHREADS can improve performance on multiple-CPU computers. For convert-intensive jobs, set CONVERTTHREADS to 2 or 3 as a starting point for performance tuning. Except for computers with many CPUs, the useful maximum number of CONVERTTHREADS is almost always less than 10.

The total number of convert threads that **onpload** uses is as follows:

$$\text{CONVERTTHREADS} * \text{numdevices}$$

where *numdevices* is the number of devices in the current device array.

Having more than one converter per thread, in general, allows the conversion phase to run faster given that CPU resources are available. Conversion can be a CPU-intensive phase if complex conversions are being performed.

CONVERTVPS

<i>default value</i>	Single-processor computer: 1 Multiprocessor computer: 50 percent of physical CPUs
<i>range of values</i>	From 1 to the number of physical CPUs
<i>refer to</i>	“Loading and Unloading Data” on page 15-14

The CONVERTVPS parameter limits the maximum number of VPs used for convert threads. This parameter limits the number of VPs that the **onpload** client uses so that **onpload** does not monopolize system resources.

Setting CONVERTVPS too large can cause performance degradation. Do not set more converter VPs than there are physical CPUs. If the number of CONVERTVPS exceeds the number of physical CPUs, system resources are consumed with no performance benefit.

On single-CPU computers, increasing this parameter has a negative effect on performance.

HPLAPIVERSION

default value 0

range of values 0 or 1

0 = The custom conversion or driver function receives three arguments:

- The buffer into which the output should be placed
- The maximum length of the output buffer
- The value of the input field

1 = The custom conversion or driver function receives four arguments:

- The buffer into which the output should be placed
- The maximum length of the output buffer
- The value of the input field
- The length of the input value

refer to [“The onpload Conversion Process” on page E-2](#)

The HPLAPIVERSION configuration parameter specifies whether to use custom conversion or driver functions with three or four arguments. Using four arguments allows different lengths for data in the input and output buffers.

HPL_DYNAMIC_LIB_PATH

default value **\$INFORMIXDIR/lib/ipldd09a.so**

range of values Any valid directory, plus **ipldd09a.SOLIBSUFFIX**.
(**SOLIBSUFFIX** is the shared-library suffix for your operating system.)

For security reasons, you should keep all shared libraries used by the database server in directories under \$INFORMIXDIR.

refer to [“Rebuilding the Shared-Library File” on page H-5](#)

The **ipldd09a.SOLIBSUFFIX** shared-library file can contain custom-code files. You can add custom drivers or custom conversion functions to this file to extend the functionality of the HPL. For more information on custom drivers, see [Appendix H, “Custom Drivers.”](#) For more information on custom conversion functions, see [Appendix E, “Custom Conversion Functions.”](#)

Previous versions of the database server required the **ipldd09a.SOLIBSUFFIX** file to be installed in the **/usr/lib** directory on Solaris. Although you can set the value of HPL_DYNAMIC_LIB_PATH to **/usr/lib**, doing so creates a security risk.

STRMBUFFERS

<i>default value</i>	Maximum of (4,2*CONVERTTHREADS)
<i>recommended value</i>	Maximum of (4,2*CONVERTTHREADS)
<i>range of values</i>	Integer > 4
<i>refer to</i>	“Loading and Unloading Data” on page 15-14

The STRMBUFFERS parameter sets the number of server-stream buffers per device. The **onpload** utility sends data to the database server through a *server stream*. The server stream is a set of shared-memory buffers. The memory for the server-stream buffer is allocated from the memory allocated for the database server.

Each device has a separate server stream with STRMBUFFERS buffers. Thus the total number of stream buffers is as follows:

$$\text{STRMBUFFERS} * \text{numdevices}$$

where *numdevices* is the number of devices in the current array.

STRMBUFSIZE

<i>default value</i>	64
<i>units</i>	Kilobytes
<i>range of values</i>	Minimum: 2 * operating system page size Maximum: depends on operating system resources
<i>refer to</i>	“Loading and Unloading Data” on page 15-14

The STRMBUFSIZE parameter sets the size of a server-stream buffer. Larger buffers are more efficient because moving buffers around requires less overhead.

Picture Strings

The HPL uses two types of picture strings:

- COBOL picture strings
- Other picture strings

COBOL picture strings describe a data field in a file that a COBOL program generates. For a discussion of COBOL picture strings, see [“COBOL Records” on page 7-24](#). The other picture-string type reformats and masks character data. This appendix discusses the non-COBOL picture strings.

Picture strings allow you to insert constants, strip unwanted characters, and organize the position of character data. Picture strings have three basic types: alphanumeric, numeric, and date. Each type is handled uniquely. The picture-string type is determined by the control characters that you use to specify the picture.

You specify the picture string in the **Picture** text box in the Mapping Options window. For information about the Mapping Options window, see [“Mapping Options” on page 9-14](#).

Alphanumeric Pictures

Alphanumeric pictures control formatting of alphanumeric strings. An alphanumeric picture allows you to mix constant characters in the picture specification with the data being processed. You can also mask out unwanted character types.

When the HPL processes an alphanumeric picture, the picture string is scanned until a picture-control character is found. All noncontrol characters in the picture string are placed directly into the output string.

When a control character is found in the picture string, the input data is scanned until a character that matches the type of the picture-replacement character is found. This character is placed in the output string, and the process is repeated.

The alphanumeric picture-control characters are `x`, `a`, `A`, `9`, and `\`. A picture string that includes any of the preceding characters is, by definition, an alphanumeric picture string. All other characters in an alphanumeric picture string are treated as literals and inserted directly into the resulting output string.

The following table describes the behavior of the alphanumeric picture-control characters.

Character	Definition
X	Replaces the control character with any character from input data.
A	Replaces the control character with an alphanumeric character from input.
a	Replaces the control character with an alphabetic character from input.
9	Replaces the control character with a numeric character from input. Stuffs the string with leading 0 characters so that the length of the input string matches the length of the picture specification.
\	Causes the character that follows the backslash to be placed in the output. That is, the character that follows a backslash is not a control character.

The following table lists some examples of alphanumeric pictures.

Picture	Input Data	Output Data
XX-AJXXXX	12P45-q	12-PJ45-q
AA-\AJAAAA	12P45-q	12-AJP45q
aaaaaaaa	12P45-q	Pq
aa99999	123abc	ab00000

Numeric Pictures

Numeric pictures allow you to decode and reformat integer and decimal numeric values. A value is interpreted as a numeric value only if its picture string contains numeric picture-control characters.

The input data is first scanned for the number of digits to the left and right of the decimal point (if any), and for a negative sign that can either precede or follow the data. The picture string is then used to reformat the value. The numeric picture-control characters are 9, S, V, and Z.

The following table describes the behavior of the numeric picture-control characters.

Character	Definition
9	Replaces the control character with a numeric character.
S	Replaces the control character with a minus sign if the input value is negative.
V	Inserts a decimal point.
Z	Replaces the control character with a numeric character or a leading zero.

The following table lists some examples of numeric pictures.

Picture	Input Data	Output Data	Comment
9999999	123	0000123	Simple reformat
S999.99	123-	-123.00	Sign controlled on output
99V99	123	01.23	Implicit decimal point
99.99	103.455	103.45	Strip decimals

Date Pictures

When you load data, the date-format picture specifies how the HPL formats the input data before it writes the data into a database. When you extract data from a database, the date-format picture specifies how the HPL reformats the date before it writes the date to the output.

The date control characters are *M*, *D*, and *Y*. The following table provides definitions of these control characters.

Character	Definition
D	Day value
H	Hour value
M	Month value or minute value
S	Second value
Y	Year value

You can use Informix DATETIME strings, such as YYYY/MM/DD HH:MM:SS.

The following table shows some examples of date picture strings.

Picture	DBDATE Value	Input	Output
MM/DD/YY	YMD2/	12/20/91	91/12/20
MM/DD/YY	DMY2/	12/20/91	20/12/91
MMDDYY	DMY2/	122091	20/12/91
MM DD YYYY	DMY4/	12/20/1991	20/12/1991
MM/DD/YY	DMY2.	12/20/91	20.12.91
M/D/YY	DMY2/	02/01/91	2/1/91

Match Condition Operators and Characters

Operator Descriptions and Examples

This appendix describes the operators that are available when you match text and it provides an example of each operator.

Operator	Description
= <i>value</i>	<p>Matches if the character string in, or the value of, the data-record field equals the specified text or value. If you specify a character string, the characters must be delimited by quotes.</p> <p>For example, if you are matching on a field named City, the match condition = "Dallas" selects all records whose City field contains the entry Dallas.</p>
<i>value</i>	<p>Equals (=) is the default operator. Thus this case is equivalent to =<i>value</i>, except that the characters do <i>not</i> have to be delimited by quotes.</p> <p>For example, if you are matching on a field named City, the match condition Dallas selects all records whose City field contains the entry Dallas.</p>
> <i>value</i>	<p>Matches if the data record field is greater than the specified value.</p> <p>For example, if you are matching on a field named Income, the match condition > 50000 selects all records whose Income field contains an entry greater than 50,000.</p> <p>Character strings must be delimited by quotes (> "Jones").</p>
< <i>value</i>	<p>Matches if the data record field is less than the specified value.</p> <p>For example, if you are matching on a field named Income, the match condition < 50000 selects all records whose Income field contains an entry less than 50,000.</p> <p>Character strings must be delimited by quotes (< "Jones").</p>
>= <i>value</i>	<p>Matches if the data-record field is equal to or greater than the specified value.</p> <p>For example, if you are matching on a field named Income, the match condition >= 50000 selects all records whose Income field contains an entry 50,000 or greater.</p> <p>Character strings must be delimited by quotes (>= "Jones").</p>

(1 of 3)

Operator	Description
<code><= value</code>	<p>Matches if the data-record field is less than or equal to the specified value.</p> <p>For example, if you are matching on a field named Income, the match condition <code><= 50000</code> selects all records whose Income field contains an entry 50,000 or less.</p> <p>Character strings must be delimited by quotes (<code><= "Jones"</code>).</p>
<code><> value</code>	<p>Matches if the data-record field is not equal to the specified value. Character strings must be delimited by quotes.</p> <p>For example, if you are matching on a field named State, the match condition <code><> "TX"</code> selects all records whose State field contains an entry other than TX.</p>
<code>between value1 and value2</code>	<p>Matches if the data-record field is between the range specified in value 1 and value 2.</p> <p>For example, if you are matching on a field named Income, the match condition <code>between 50000 and 100000</code> selects all records whose Income field contains an entry between 50,000 and 100,000.</p> <p>Character strings must be delimited by quotes.</p>
<code>and</code>	<p>Constructs a comparison of two or more items. Matches only if the data record fields match <i>all</i> of the comparisons. Note that the comparisons can only be applied to one field.</p> <p>For example, if you are matching a field named Income, the match condition <code>> 5000 and <> 6000</code> selects all the records with income greater than 5000, but not a record of 6000.</p>
<code>or</code>	<p>Constructs a comparison of two or more items. Matches if the data record field(s) matches <i>any</i> of the comparisons.</p> <p>For example, if you are matching on a field named City, the match condition <code>"Dallas" or "Fort Worth"</code> selects all records whose City field contains either the entry Dallas or the entry Fort Worth.</p>

(2 of 3)

Operator	Description
NULL	<p>Matches when all characters are blank or when a character is binary zero (null).</p> <p>For example, you might want to discard any records that have all blanks for a name field.</p>
* (asterisk)	<p>Wildcard match of any number of characters in a string.</p> <p>For example, to match on a field that contains the city name and state, the match condition <code>Da11*</code> would select records with any of the following entries:</p> <ul style="list-style-type: none">■ Dallas-Forth Worth■ Dallas, TX■ Dallas TX
?	<p>Matches any single character in a string.</p> <p>For example, to match on a field that contains a last name, the match condition <code>Sm?t:h</code> would select records with any of the following entries:</p> <ul style="list-style-type: none">■ Smith■ Smyth

(3 of 3)

Custom Conversion Functions

Custom conversion functions allow you to add additional data conversion capability to the HPL. This feature lets **onpload** call a custom conversion function during the data-conversion process.

When you create a custom conversion function, you associate it with a particular mapping of input field to output field. To associate a custom function with a field, enter the name of the function in the **Function** text box of the Mapping Options window. For information about mapping options, see [“Mapping Options” on page 9-14](#).

Although the mapping options associate the custom conversion function with a particular field, the function can access all the input data fields and all the output data fields through a set of API functions provided with the **onpload** utility.

Custom Conversion Example

As an example, you might implement a custom conversion functions to do the following, expressed in pseudocode:

```
IF input field 1 satisfies condition A
THEN
  DO calculation X on input field 7
  OUTPUT data to output column 7
ELSE
  DO calculation Y on input field 6
  OUTPUT data to output column 5
```

The custom conversion function feature is available only on computers with operating systems that support dynamic linking.

The onpload Conversion Process

The **onpload** conversion process is identical for both import or export operations. The **onpload** utility:

- Extracts the source data from their native format.
- Examines the map.
- Applies the conversions called out in the map. Conversion order is implied by the ordering of the source-field names that are specified in the map.
- Calls any custom conversion function that is specified for a source field. The parameters **onpload** passes to the conversion function depends on the value of the HPLAPIVERSION parameter in the **plconfig** file:
 - If HPLAPIVERSION is set to 0 or it is not present in the **plconfig** file, then **onpload** passes the buffer into which the output should be placed, the maximum length of the output buffer, and the value of the input field.
 - If HPLAPIVERSION is set to 1, then **onpload** passes the buffer into which the output should be placed, the maximum length of the output buffer, the value of the input field, and the length of the input value.

For more information, see [“HPLAPIVERSION” on page B-7](#).

- If there is a custom conversion function, applies the value that the custom conversion function places in the function output buffer to the destination field that is associated with the source field in the map.
- Sends the results to the output generators.

The custom conversion function API uses ASCII strings as the canonical data type. The API functions present data as ASCII strings and expect data from the custom conversion functions to be presented as ASCII strings. The API functions convert source data of different types to ASCII strings, and also convert ASCII string data from custom conversion functions to destination data types.

Custom conversion functions are loaded into the **onpload** executable through a shared library.

To integrate your custom conversion functions into the onpload executable

1. Prepare the custom conversion function table.

The **onpload** utility uses the entries in a function table to translate custom-function string names that are specified in the load or unload map. You must supply the function table and the custom conversion functions.

To code the function table, use the following template for the file **plcstcnv.c**. You can copy this template from the **\$INFORMIXDIR/incl/hpl** directory. Add as many entries into the **functiontable** array as needed.

The **onpload** utility searches the **functiontable** array for the string name of the custom conversion function that the map specifies. The function pointer that is associated with the string name is retrieved and used as the custom conversion function. In the following template for the file **plcstcnv.c**, **ycf1** and **ycf2** are the strings that **ipload** uses to find the custom functions **your_conversion_func1** and **your_conversion_func2**, respectively. To add custom function string names to the **onpload** database, see [“Mapping Options” on page 9-14](#).

```
/*
 * plcstcnv.c
 */
#include "pldriver.h"

extern int your_conversion_func1();
extern int your_conversion_func2();

struct functable functiontable[] =
{
    {"ycf1", your_conversion_func1},
    {"ycf2", your_conversion_func2},
    {0, 0}
};
/* end of plcstcnv.c */
```

2. Prepare your conversion functions. Use the template in the following example to code your conversion functions:

```
/*
 * your_custom_conversion.c
 */

/*
 * The argument list must be adhered to.
 */
int your_conversion_func1(outbuffer, buflen, value)
    char *outbuffer; /* where to put your output */
    int  buflen;      /* max size of buffer in bytes*/
    char *value;      /* input value */
{
    /* your processing here */
}

int your_conversion_func2(outbuffer, buflen, value)
    char *outbuffer; /* where to put your output */
    int  buflen;      /* max size of buffer */
    char *value;      /* input value */
{
    /* your processing here */
}
/* end of your_custom_conversion.c */
```

3. Rebuild the **onpload** shared-library file **ipldd09a.SOLIBSUFFIX** (where **SOLIBSUFFIX** is the shared-library suffix for your platform). Follow the instructions in [“Rebuilding the Shared-Library File” on page H-5](#).

The **onpload** utility uses the same library for both custom conversion functions and custom drivers. When you rebuild the library, if there are custom drivers, you must link the custom driver code as well as the custom conversion functions.

4. Install the shared library in the appropriate path for your platform. For example, on Solaris the shared library should be installed in **\$INFORMIXDIR/lib** or any configurable path that is specified by the **HPL_DYNAMIC_LIB_PATH** configuration parameter.

API Functions

Depending on the value of the HPLAPIVERSION parameter in the **plconfig** file, **onpload** expects your custom conversion function to have one of the prototypes listed below.

If the HPLAPIVERSION parameter is set to 0 or HPLAPIVERSION is not present in the **plconfig** file, use the following prototype:

```
/*
 * input::  char* outbuffer:  where to put your output.
 *          int  buflen:     size you have for your output.
 *          char* value:     the input value to work on.
 * return:: 0               to indicate ok.
 *          non-zero        to discard entire record.
 */

int your_func(outbuffer, buflen, value)
    char *outbuffer;
    int  buflen;
    char *value;
{
    /* your processing here */
}
```

If the HPLAPIVERSION parameter is set to 1, use the following prototype:

```
/*
 * input::  char* outbuffer:  where to put your output.
 *          int  buflen:     size you have for your output.
 *          char* value:     the input value to work on.
 *          int  vallen:     size of the input value.
 * return:: 0               to indicate ok.
 *          non-zero        to discard entire record.
 */

int your_func(outbuffer, buflen, value, vallen)
    char *outbuffer;
    int  buflen;
    char *value;
    int  vallen;
{
    /* your processing here */
}
```

To discard an entire record, return a nonzero value. Otherwise, return a zero value.

The following functions support your access to data in the source and destination buffers.

DBXget_source_value(fldname,buffer,buflen)

This routine retrieves the source value that is associated with **fldname** and copies the value to the specified buffer.

Arguments	I/O	Description
char *fldname	Input	Name of source field, as defined in ipload map
char *buffer	Input	Address where fldname value is placed
int buflen	Input	Buffer size in bytes

DBXget_dest_value(fldname,buffer,buflen)

This routine retrieves the destination value that is associated with **fldname** and copies the value to the specified buffer.

Arguments	I/O	Description
char *fldname	Input	Name of destination field, as defined in ipload map
char *buffer	Input	Address where fldname value is placed
int buflen	Input	Buffer size in bytes

DBXput_dest_value(fldname,buffer)

If a previous conversion has not set the destination value, this routine sets the destination value that is passed to the buffer. The **ipload** utility automatically clips the data value if it is too long.

Arguments	I/O	Description
char *fldname	Input	Name of destination field, as defined in ipload map
char *buffer	Input	Address where fldname value is placed

DBXget_dest_length(fldname)

This routine returns the maximum length of the data buffer that is associated with **fldname**.

Arguments	I/O	Description
char *fldname	Input	Name of destination field, as defined in ipload map

onstat -j Option

The **-j** option of the **onstat** utility provides special information about the status of an **onpload** job. The **-j** option provides an interactive mode that is analogous to **onstat -i**. For information about **onstat -i** and how to use the interactive mode, refer to the *Administrator's Reference*.

Using the onstat -j Option

When **onpload** starts, it writes a series of messages to **stdout** or to a log file. The following lines show a typical **onpload** log file:

```
Mon Jul 24 16:11:30 1995

SHMBASE          0x4400000
CLIENTNUM        0x49010000
Session ID 1

Load Database    -> cnv001
Load Table       -> cnv001a
Load File        -> testrec.dat
Record Mapping   -> cnv001a

Database Load Completed -- Processed 50 Records
Records Inserted-> 50
Detected Errors--> 0
Engine Rejected--> 0

Mon Jul 24 16:11:37 1995
```

The two lines that start with SHMBASE and CLIENTNUM provide the information that you need to locate shared memory for an instance of **onpload**. The **oninit** process has similar values stored in the **\$ONCONFIG** file. When you use **onstat** to gather information about the **oninit** process, **onstat** uses information from **\$INFORMIXDIR/etc/\$ONCONFIG** to locate shared memory. When you use **onstat** to gather information about **onpload**, you must give **onstat** the name of a file that contains SHMBASE and CLIENTNUM information.

Typically the file that contains the SHMBASE and CLIENTNUM information is the log file. For example, if the **onpload** log file is **/tmp/cnv001a.log**, you can enter the following command:

```
onstat -j /tmp/cnv001a.log
```

The previous command causes **onstat** to attach to **onpload** shared memory and to enter interactive mode. You can then enter ? or any other bogus request to see a usage message displayed. An example follows:

```
onstat> ?
Interactive Mode: One command per line, and - are optional.
-rz      repeat option every n seconds (default: 5) and
        zero profile counts
MT COMMANDS:
all      Print all MT information
ath      Print all threads
wai      Print waiting threads
act      Print active threads
rea      Print ready threads
sle      Print all sleeping threads
spi      print spin locks with long spins
sch      print VP scheduler statistics
lmx      Print all locked mutexes
wmx      Print all mutexes with waiters
con      Print conditions with waiters
stk <tid> Dump the stack of a specified thread
glo      Print MT global information
mem <pool name|session id> print pool statistics.
seg      Print memory segment statistics.
rbm      print block map for resident segment
nbm      print block map for non-resident segments
afr <pool name|session id> Print allocated poolfragments.
ffr <pool name|session id> Print free pool fragments.
ufr <pool name|session id> Print pool usage breakdown
ioy      Print disk IO statistics by vp
iof      Print disk IO statistics by chunk/file
ioq      Print disk IO statistics by queue
iog      Print AIO global information
iob      Print big buffer usage by IO VP class
sts      Print max and current stack sizes
qst      print queue statistics
wst      print thread wait statistics
jal      Print all Pload information
jct      Print Pload control table
jpa      Print Pload program arguments
jta      Print Pload thread array
jmq      Print Pload message queues, jms for summary only
onstat>
```

Most of the options are the same as those that you use to gather information about Dynamic Server, with the following exceptions:

<code>jal</code>	Print all Pload information
<code>jct</code>	Print Pload control table
<code>jpa</code>	Print Pload program arguments
<code>jta</code>	Print Pload thread array
<code>jmq</code>	Print Pload message queues, <code>jms</code> for summary only

These options apply only to **onpload**. You can use **onstat -j** to check the status of a thread, locate the VP and its PID, and then attach a debugger to a particular thread. The options for **onstat** that do not apply to **onpload** are not available (for example, `-g ses`).

HPL Log-File and Pop-Up Messages

This appendix provides explanatory notes and corrective actions for unnumbered messages that print in the HPL log file. The appendix also includes information specific to messages that are returned to standard output or appear in a pop-up dialog box (depending on the way you invoked **onpload**).

For more information on error messages and corrective actions, use the **finderr** (UNIX) or **Informix Error Messages** (Windows) utility.

Several HPL error messages refer to the **errno.h** file, which is located in the following directories:

- **/usr/include/errno.h** ♦
- **errno.h**, **windsock.h**, and **winsock2.h** in the **include** subdirectory for Microsoft Visual C++. ♦

A few of the messages included here might require you to contact Technical Support. Such messages are rarely, if ever, seen at customer locations.

For information on how to view the log file and some guidance on how and when you might want to read it, see [“Viewing the Log File” on page 14-9](#).

UNIX

Windows

How the Messages Are Ordered

The HPL log-file messages appear in this appendix in alphabetical order, sorted with the following additional rules:

- The time stamp that precedes each message is ignored.
- Letter case in alphabetization is ignored.
- File, record, database server, and table names are ignored.
- Error numbers are ignored.
- Spaces are ignored.
- Quotation marks are ignored.
- The word *the* is ignored if it is the first word in the message.

A cause and suggested corrective action for a message or group of messages follows the message text.

A section that lists pop-up messages (or messages that are returned to standard error) appears after the log-file message sections. Messages in this section are arranged according to the same rules that apply to log-file messages.

Message Categories

Four general categories of messages can be defined, although some messages fall into more than one category:

- Routine information
- Assertion-failed messages
- Administrative action needed
- Fatal error detected

The assertion-failed messages reflect their traditional use by technical staff to assist in troubleshooting and diagnostics. The information that they report often falls into the category of *unexpected events* that might or might not develop into problems caught by other error codes. Moreover, the messages are terse and often extremely technical. They might report on one or two isolated statistics and not provide an overall picture.

When technical staff investigate a problem, this information can suggest to them possible research paths. However, you might find that the information has little or no application when it is taken out of this context, or when processing proceeds normally.

Log-File Messages

Blob conversion error occurred on record *record_num*.

Cause: The SQLBYTE blob data could not be converted to HEXASCII, or the SQLTEXT blob has invalid character data (characters not in the code set).

Action: Remove the invalid characters from the input data.

Cannot access database table *table_name*: SQL error *error_num*.

Cause: The target database table cannot be accessed.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot allocate shared memory.

Cause: A memory allocation error occurred. Probably the system is out of virtual shared memory.

Action: Run **onpload** again when fewer users are on the system.

For UNIX, increase the amount of available shared memory with the UNIX kernel configuration.

For Windows, reduce the number of applications running concurrently.

Cannot allocate TLI memory for *operating_system* structure.

Cause: System memory cannot be allocated for communications. This situation should only happen if all system resources are consumed.

Action: Note the circumstances and contact Technical Support.

UNIX

Cannot bind socket connection: errno=operating-system_error_num.

Cause: A TCP socket cannot be opened.

Action: See your **errno.h** file.

Cannot bind TLI connection: t_errno=t_error_num.

Cause: An error occurred when **onpload** attempted to open a TLI connection.

Action: Check that TLI services are installed on the operating system. See your **tiuser.h** file.

Cannot configure driver *driver_name*.

Cause: You may be specifying a driver incorrectly. If the Driver Class specification is not **Fixed**, **Cobol**, or **Delimited**, either the **onpload** custom-driver shared library is not in the path-name, or the custom-driver shared library is not installed correctly.

Action: For information on building a shared library, see [“Rebuilding the Shared-Library File” on page H-5](#). Make sure your driver is configured correctly for **Fixed**, **Cobol**, or **Delimited**.

Cannot connect to message server: Socket error=UNIX_error_num.

Cause: This message is generated by **ipload** when it cannot connect to the **onpload** socket service.

Action: See **/usr/include/errno.h**.

Cannot connect to message server: TLI error=*t_error_num*, TLI event=*t_event_num*, errno=*error_num*.

Cause: An error occurred when **onpload** attempted to open a TLI connection.

Action: Check that TLI services are installed on the operating system. See */usr/include/tiuser.h* (*t_error_num*). ♦

Cannot connect to server *server_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The target database server cannot be opened.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot connect worker to server data stream.

Cause: A possible permissions problem exists for **onpload** or **oninit**.

Action: Note the circumstances and contact Technical Support.

Cannot disable *table_name* object constraints: SQL error *error_num*, ISAM error *error_num*.

Cause: The constraint objects are disabled during the load and re-enabled after the load. An error occurred when **onpload** attempted to disable the constraint objects.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot disable primary-key constraint. Child-table references exist.

Cause: You attempted to use express mode to load a table that has child-table records that refer to it. Express mode does not support this condition. (The **onpload** utility cannot disable the primary key constraint when child-table records refer to the load table.)

Action: Perform the load in deluxe mode or remove the constraint in question.

Cannot express load to logged table on HDR server *server_name*.

Cause: You attempted to use express mode to load an HDR replicated table. Express mode does not support this condition.

Action: Perform the load in deluxe mode.

Cannot filter indexes for table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The index objects are set to filtering mode during the load and re-enabled after the load. An error occurred when **onpload** attempted to set the indexes objects to filtering mode.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot find the shared library path in the *plconfig* file. Using the shared library from the default location *library_location*.

Cause: The **ipldd09a.so** shared library path is not set in the **plconfig** file.

Action: If the default location is not correct, set the correct shared library path in the **plconfig** file using the **HPL_DYNAMIC_LIB_PATH** configuration parameter.

Cannot find the user-defined function *user_func_name* in the shared library: error *error_num*.

Cause: The **onpload** process could not find the required user-defined function in the shared library.

Action: Restart the **onpload** load or unload with the correct shared library or function name in the **pload** job definition.

Cannot get systable info for table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: Cannot access the **systable** table to get dictionary information for the indicated table.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot initialize shared library handling.

Cause: The **pload** utility cannot start because it failed to initialize the shared library-handling functionality.

Action: Ensure that the computer has the proper resources and that the shared library has been built properly.

Cannot load code-set conversion file from -> *file_name* to -> *file_name*.

Cause: The data type for the load file is different than the data type for the database server. The code set does not exist in the **\$INFORMIXDIR/gls/cvx** or **%INFORMIXDIR%\gls\cvx** directory where *x* is the version number of the **cv** subdirectory.

Action: Check that the file exists. Check the file for permissions.

Cannot load mapping definitions.

Cause: A memory-allocation error or database-integrity error occurred when **onpload** accessed the **onpload** database.

Action: Use oncheck to check the **maps**, **mapitem**, **mapoption**, **formats**, and **formatitem** tables for consistency. If the tables are consistent, a referential integrity problem between the map and the format the map references might exist. If the problems persists, contact Technical Support.

Cannot load the shared library *library_location*.

Shared library load failed with error message *error_message*.

Cause: The **onpload** utility could not load the shared library from the *library_location* path and failed with *error_message*.

Action: For more information, use the **finderr** or **Informix Error Messages** utility. Correct the problem and reload the shared library.

Cannot locate delimiter in data file.

Cause: No delimiter is found when **onpload** scans for an end-of-record delimiter in the load data.

Action: Check that the end-of-record delimiter specification is correct, or that you have the correct data file. Note differences in the end-of-line characters between UNIX and Windows.

Cannot open.

Cause: An internal error occurred when **onpload** attempted to open the load or unload file.

Action: Note the circumstances and contact Technical Support.

Cannot open blob file *file_name*. blob not loaded.

Cause: The record references a filename that should contain a blob, but the file cannot be located.

Action: Check that the BYTE or TEXT file exists.

Cannot open simple large object file *file_name*. Simple large object not loaded.

Cause: The record references a filename that should contain a simple large object, but the file cannot be located.

Action: Check that the simple-large-object file exists.

Cannot open database *database_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The target database cannot be opened.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot open file *file_name*: error number *operating-system_error_num*.

Cause: The file cannot be opened.

Action: See your **errno.h** file.

Cannot open TCP connection for *server_name*: errno *operating-system_error_num*.

Cause: A TCP socket cannot be opened.

Action: See your **errno.h** file.

Cannot perform express-mode load on table with pseudo rowid.

Cause: The load table is fragmented by row ID. Express mode does not support this condition.

Action: Perform the load in deluxe mode.

Cannot perform express-mode load with `rowsize=row_length > page_size`.

Cause: The table-row size exceeds page size. Express mode does not support this condition.

Action: Perform the load in deluxe mode.

Cannot read file `file_name`: AIO error code `operating-system_error_num`.

Cause: The load file cannot be accessed. This error might result from operating-system limitations; the **onpload** utility cannot load successfully from a file (on disk) that is longer than 2 gigabytes.

Action: See your **errno.h** file.

Cannot re-enable all objects: `num_violations` violations detected. Check for violations in violations table `table_name` and diagnostics table `table_name`.

Cause: Data loaded by **onpload** violates the object constraints specified for the table. The records that violate the object-constraints have been placed in the **violations** table, and the reason code for each violation is listed in the **diagnostics** table.

Action: Review the information in the violations and **diagnostics** tables.

Cannot re-order query statement to align simple large objects or Ext Types.

Cause: The unload query does not contain a FROM clause.

Action: Rewrite the query so that it contains a FROM clause.

Cannot re-order query statement to align blobs.

Cause: The unload query does not contain a FROM clause.

Action: Rewrite the query so that it contains a FROM clause.

Cannot set mode of *table_name* objects from *current_mode* to *final_mode* mode: SQL error *error_num*, ISAM error *error_num*.

Cause: The constraints objects are disabled during the load and re-enabled after the load. An error occurred when **onpload** attempted to reset constraint objects back to their original state.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot start violations table for *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: An error occurred when **onpload** attempted to set up the violations table for the load table.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot stop violations table for *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: If a violations table exists on the load table, violations can be turned off during the load. An error occurred when **onpload** attempted to turn off violations detection.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot unload to multiple devices when the given query cannot be executed in parallel.

Cause: The server determined that the query cannot be executed in parallel.

Action: Remove non-parallel aspects of the query, such as non-parallel UDRs, or unload to a single device.

Cannot write file *file_name*: AIO error code *operating-system_error_num*.

Cause: The unload file cannot be accessed.

Action: See your **errno.h** file.

Code-set conversion overflow.

Cause: The code-set conversion caused the number of bytes in the BYTE and TEXT data to expand or contract when **onpload** unloaded the data into a fixed-format record. The **onpload** utility cannot update the BYTE and TEXT data tag in the record that specifies the length of the BYTE and TEXT data at this stage.

Action: To unload this data, use a delimited format.

Conversion of onpload database failed due to error *error_num*.

Cause: The **onpload** tried to convert the old database when **onpload** ran for the first time on the new database server. This conversion fails because of the error referenced in the error message.

Action: For more information, use the **finderr** or **Informix Error Messages** utility. Resolve this before you rerun **onpload**.

UNIX

Conversion of onpload database failed due to error *error_num*, run as user *informix*.

Cause: Database conversion fails because the current user running **onpload** does not have sufficient privileges to convert the **onpload** database.

Action: Run the **onpload** job as user **informix** once.

Custom conversion function *function_name* not found in shared library.

Cause: The custom function specified in a map option was not located in **ipldd09a.so**. The shared library extension is platform specific; for example, the **.so** extension is specific for Solaris and is probably different on other platforms.

Action: For information on how to configure the custom function library, see [Appendix E](#). ♦

Discarded *num_bytes* null bytes from end of tape device *device_name*.

Cause: The tape data is not blocked in a multiple of the record size, so that the last block of data contained bytes that are discarded. This situation occurs on devices with stream cartridges that allow writing to the device only in whole blocks.

Action: If necessary, manually enter the discarded data.

Environment variable *variable_name* expansion would overflow string.

Cause: A mapping option specifies an environment variable as the default value, but expansion of the environment variable requires more space than allocated to the column.

Action: Use a shorter default value, or expand the length of the column.

Error accepting socket connection: *errno=operating-system_error_num*.

Cause: A TCP socket cannot be accessed.

Action: See your **errno.h** file.

Error accessing *file_name*.

Cause: An error occurred when **onpload** attempted to open the load or unload file.

Action: Check that the file exists. Check the file for permissions.

Error accessing format: SQL error *error_num*, ISAM error *error_num*.

Cause: An integrity problem exists in the **onpload** database. The format for the map does not exist, or a problem exists with the **format** or **formatitem** table.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing map *map_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The requested map for the load or unload does not exist, or a problem exists with the **onpload** database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing sysmaster: SQL error *error_num*, ISAM error *error_num*.

Cause: An access error occurred on the sysmaster database on the target server where **onpload** attempted to perform the load or unload job.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The target database table cannot be accessed.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error:AIO buffer size *buffer_size* is less than required minimum size *size*.

Cause: AIO buffer size is less than required size.

Action: Increase the specified buffer size in the PLCONFIG file.

Error *error_num* closing current database.

Cause: A server error occurred when **onpload** closed the **onpload** or target database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *operating-system_error_num* closing file *file_name*.

Cause: An error occurred when **onpload** closed the load or unload file.

Action: See your **errno.h** file

Error *error_num* converting record field *field_name* to column *column_name*.

Cause: A conversion error occurred when **onpload** attempted to convert the record data to the database column type.

Action: For more information, use the **finderr** or **Informix Error Messages** utility. If the load map indicates that the data field is mapped to the correct column, check that the supplied data is valid.

Error declaring cursor: could not get table info.

Cause: Cannot access information about the load table.

Action: Check the validity of the table in the target database.

Error declaring cursor: SQL error *error_num*, ISAM error *error_num*.

Cause: The **onpload** utility is unable to use the autogenerated formats and maps to create entries in a table in the **onpload** database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error describing unload query *query_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The unload query cannot be processed.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* initializing backend connection.

Cause: An internal error occurred in **onpload**. Probably the server went down.

Action: Note the circumstances and contact Technical Support.

Error inserting into table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The **onpload** utility is unable to use the autogenerated formats and maps to create entries in a table in the **onpload** database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

UNIX

Error listening for socket connection: `t_errno=t_error_num`
`errno=operating-system_error_num`.

Cause: An error occurred listening on a Socket connection.

Action: See your **errno.h** file.

Error listening for TLI connection: `t_errno=t_error_num`
`errno=UNIX_error_num`.

Cause: An error occurred listening on a TLI connection.

Action: See **/usr/include/tiuser.h** (`t_error_num`). ♦

Error `error_num` on record `record_num` converting column
`column_name` to record field `field_name`.

Cause: A conversion error occurred when **onpload** attempted to convert the column data to the record field type.

Action: For more information, use the **finderr** or **Informix Error Messages** utility. Check the load map to verify that the column is mapped to the correct record field.

Error occurred on record `%d` reading pipe `%s`.

Cause: A conversion error occurred in a record.

Action: No action is required.

Error on close of server load session: SQL error `error_num`, ISAM error `error_num`.

Cause: An internal error occurred in **onpload**. Probably the server went down.

Action: Note the circumstances and contact Technical Support.

Error opening cursor: SQL Error *error_num*.

Cause: An error occurred when **onpload** attempted to set up an insert cursor on the load table.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error preparing query: SQL error *error_num*.

Cause: The unload query cannot be processed.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error preparing statement *statement_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: An internal error occurred when **onpload** attempted to access the **onpload** database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error preparing unload query *query_name*: SQL error *error_num*, ISAM error *error_num*.

Cause: The unload query cannot be processed.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* reading message queue.

Cause: This critical initialization error probably means that the operating kernel does not have enough shared memory or semaphores configured, or that the allocated shared memory was removed.

Action: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

Error *operating-system_error_num* reading SOCKET connection.

Cause: An error occurred reading a socket connection.

Action: See your **errno.h** file.

Error *operating-system_error_num* reading TLI connection.

Cause: An error occurred reading a TLI connection.

Action: See your **errno.h** file.

Error *error_num* setting isolation level.

Cause: An access error occurred when **onpload** attempted to set the isolation level for an unload job.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* writing message on message queue.

Cause: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured, or that the allocated shared memory has been removed.

Action: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

Error *operating-system_error_num* writing SOCKET connection.

Cause: An error occurred writing a socket connection.

Action: See your **errno.h** file.

Error *operating-system_error_num* writing TLI connection.

Cause: An error occurred writing a TLI connection.

Action: See your **errno.h** file.

Error:Stream buffer size *buffer_size* is less than required minimum size *size*.

Cause: Stream buffer size is less than required size.

Action: Increase the specified buffer size in the PLCONFIG file.

Exhausted all attempts to allocate shared-memory key.

Cause: All the shared-memory keys in the key range tried by **onpload** are currently allocated.

Action: Wait until another **onpload** session finishes. If the problem persists, contact Technical Support.

Fatal error: cannot execute *pipe_name*.

Cause: An attempt to execute the PIPE type device in the device array failed.

Action: Make sure the PIPE entry in the device array is a valid, executable program. Pipes are only supported on UNIX.

Fatal error creating server load session: error *error_num*.

Cause: Cannot start the load session with the server.

Action: Note the circumstances and contact Technical Support.

Fatal error getting stream buffer from server.

Cause: An internal error occurred in **onpload**. Probably the server went down.

Action: Note the circumstances and contact Technical Support.

Fatal error in server row processing: SQL error *error_num*, ISAM error *error_num*.

Cause: An internal communication problem exists between the server and **onpload**.

Action: Note the circumstances and contact Technical Support.

File type device file *file_name* is not a regular (disk) file.

Cause: The device array specifies that the file is a disk file, but it is not.

Action: Change the type of the file in the device-array definition, or make sure that the file is a disk file.

Got Interrupt: Shutting down.

Cause: An internal error occurred, or a user sent an interrupt to **onpload**.

Action: If a user did not generate this interrupt, contact Technical Support.

Internal error: cannot initialize AIO library.

Cause: This critical initialization error probably means that the UNIX kernel does not have enough shared memory or semaphores configured.

Action: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

Internal error: cannot send message.

Cause: An internal error occurred in **onpload**. The most likely cause is a lack of shared memory.

Action: Note the circumstances and contact Technical Support.

Internal error: *error_num*. Contact Tech Support.

Cause: A critical internal error occurred.

Action: Note the circumstances and contact Technical Support.

Internal error: invalid message type *error_num*.

Cause: A critical internal error occurred.

Action: Note the circumstances and contact Technical Support.

Internal error *error_num* reading queue.

Cause: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

Action: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

Invalid code-set character: cannot convert.

Cause: The data being loaded or unloaded has invalid character data.

Action: Make sure that you specified the correct data type on the format definition.

Invalid HEXASCII blob representation on *field_name*, record *record_num*.

Cause: The BYTE or TEXT field being loaded was classed as HEXASCII, but the data contains a non-HEXASCII character.

Action: Fix the data.

Invalid HEXASCII simple large object representation in *fieldname*, record *record_num*.

Cause: The simple large object data field being loaded was classed as HEXASCII, but the data contains a non-HEXASCII character.

Action: Fix the data.

Invalid project name *project_name* entered.

Cause: Incorrect project name was specified for **onpload**.

Action: Check the given project name and restart **onpload**.

Invalid session ID *id_number*.

Cause: The command line specified an invalid session ID for the job to run. An entry for the entered session ID must exist in the **session** table of the **onpload** database in order to run the job.

Action: Make sure the session ID on the command line matches the correct session ID in the **session** table.

Invalid Tape Header Expecting -> *tape_name*.

Cause: Incorrect tape was mounted.

Action: Mount the correct tape.

Map `map_name` type is not of the required type.

Cause: Incorrect map was specified to **onpload**. You must use a load map for a load job and an unload map for an unload job.

Action: Verify that you are using the correct map type.

Method not supported by current driver.

Cause: An internal error occurred in **onpload**.

Action: Note the circumstances and contact Technical Support.

MT cannot bind to vpid.

Cause: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

Action: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

MT internal failure.

Cause: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

Action: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

MT failure putting CPU online.

Cause: This critical initialization error probably means that the UNIX kernel does not have enough shared memory or semaphores configured.

Action: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

No insert permission on table *table_name*.

Cause: You cannot load the indicated table because the DBA has not granted permission for you to do so.

Action: Make sure that you have insert permissions on the table.

No mapping to blob field *field_name*.

Cause: The record format specifies a BYTE and TEXT field type, but no column from the query is mapped to the record field.

Action: Map a column to the field, or remove the field from the record format.

No mapping to simple large object field *field_name*.

Cause: The record format specifies a simple large object or Ext Type field type, but no column from the query is mapped to the record field.

Action: Map a column to the field, or remove the field from the record format.

onpload must run on the host *host_name* that contains the target database.

Cause: User tried to run **onpload** on a host computer other than the one that has the target database.

Action: Run **onpload** on the host specified in the error message.

UNIX

onpload terminated by signal.

Cause: Either an internal error occurred or a user sent **onpload** a termination signal.

Action: If the signal is not SIGKILL, SIGTERM, or SIGQUIT, note the circumstances and contact Technical Support. ♦

Pipe type device file `file_name` is not a regular file.

Cause: The device array specifies that the file is a pipe (executable program) file, but it is not.

Action: Change the type of the file in the device-array definition, or make sure that the file is an executable disk file. Pipes are only supported on UNIX.

Pload cannot reorder queries having expressions/aggregates and blobs in the same select list. Please reorder the select list in the query in the following order 1. non-blob columns 2. blob columns in the format order.

Cause: The unload query contains expressions and/or aggregates and BYTE or TEXT data columns in the select list. The **onpload** requires that the BYTE or TEXT data columns are in the end of the select list to avoid staging a BYTE or TEXT data column during unloading. But **onpload** is unable to re-order the select list because of expressions, aggregates, or both.

Action: Re-order the select list in the unload query to have all non-BYTE or TEXT columns first and BYTE or TEXT columns in the same order as they are specified in the output format.

Pload cannot reorder queries having expressions/aggregates and blobs/udts in the same select list. Please reorder the select list in the query in the following order 1. non-blob, non-udt columns 2. inrow udts in the case of fixed format 3. other blob/udt columns

Cause: **onpload** requires that the special columns (BLOB and UDT) appear at the end of the select list. **onpload** will reorder simple select statements but is unable to re-order the select list because of expressions, aggregates, or both.

Action: Re-order select columns manually as explained in error message. Alternatively, you can remove aggregates and expressions from the select list by selecting the columns into a temporary table and then unloading them from that table.

Query contains unmapped blob *column_name*: cannot proceed.

Cause: The unload query is extracting a BYTE or TEXT column that is not mapped to the record field.

Action: Modify the unload query so that it does not reference the BYTE or TEXT column, or map it to a field in the record format.

Query contains unmapped simple large object column *column_name*: cannot proceed.

Cause: The unload query is extracting a simple large object or Ext Type column that is not mapped to the record field.

Action: Modify the unload query so that it does not reference the simple large object or Ext Type column, or map it to a field in the record format.

Query for unload is not a select query.

Cause: The unload query does not contain a SELECT statement.

Action: Modify the query so that it contains a SELECT statement.

Record is too long to process: *recnum record_num, length record_length, bufsize buffer_size*.

Cause: The record size exceeds the size of the **onpload** buffers (AIOBUFSIZE). This error can occur when a delimited record contains simple-large-objects or Ext Types, and a format specification for a field is missing, which causes a simple large object or a Ext Type to be treated as a regular field.

Action: Increase the size of AIOBUFSIZE for this record, or check that the format specification for the field matches the input file.

Record is too long to process: recnum *record_num*, length *record_length*, bufsize *buffer_size*.

Cause: The record size exceeds the size of the **onpload** buffers (AIOBUFSIZE). This error can occur when a delimited record contains BYTE or TEXT data, and a format specification for a field is missing, which causes BYTE or TEXT data to be treated as a regular field.

Action: Increase the size of AIOBUFSIZE for this record, or check that the format specification for the field matches the input file.

Server interface error; expected *num_input* but got *num_received* instead.

Cause: An **onpload**/server interface error occurred.

Action: Note the circumstances and contact Technical Support.

SQL error *error_num* executing statement *statement_name*.

Cause: An internal error occurred when **onpload** accessed the **onpload** database.

Action: For more information, use the **finderr** or **Informix Error Messages** utility.

Simple large object conversion error occurred on record *record_num*.

Cause: The SQLBYTE simple large object or Ext Type data could not be converted to HEXASCII, or the SQLTEXT simple large object or Ext Type has invalid character data (characters not in the code set).

Action: Remove the invalid characters from the input data.

Start record *record_num* is greater than number of records
total_num read from input *file_name*.

Cause: A start record was specified for the load, but fewer records are in the input file than the indicated number of records to skip.

Action: Specify the start-record number again.

Successfully loaded the shared library *library_location*.

Cause: The shared library was loaded successfully.

Action: Check the path to verify that the correct library is being used. If the path is incorrect, edit the HPL_DYNAMIC_LIB_PATH configuration parameter in the **plconfig** file to supply the correct path.

Table *table_name* will be read-only until level-0 archive.

Cause: After an express-mode load, a level-0 archive is needed to make the table available for update.

Action: Perform a level-0 archive.

Tables with BLOBS cannot be loaded in High Performance Mode.

Cause: You attempted to use express mode to load a table that contains BYTE or TEXT data. Express mode does not support this condition.

Action: Perform the load in deluxe mode.

Tables with simple large objects or Ext Types cannot be loaded in express mode.

Cause: You attempted to use express mode to load a table that contains simple large object or Ext Type data. Express mode does not support this condition.

Action: Perform the load in deluxe mode.

Tables with BLOBS cannot be processed with no conversion (-fn).

Cause: You attempted a no-conversion load on a table with BYTE or TEXT columns. This action is not allowed.

Action: Remove the no-conversion specification, and run the job again.

Tables with simple large objects or Ext type cannot be processed with no conversion (-fn).

Cause: You attempted a no-conversion load on a table with simple large object or Ext Type columns. This action is not allowed.

Action: Remove the no-conversion specification, and run the job again.

Tape header is larger than I/O buffer: tape *header_length*, I/O *buffer_size*.

Cause: A tape header size is too large to fit into a memory buffer.

Action: Increase AIOBUFSIZE in PLCONFIG to at least the value specified for tape I/O.

Tape type device file *file_name* is not a character-special or block-special file.

Cause: The device array specifies that the file is a tape device, but it is not.

Action: Change the type of the file in the device-array definition, or make sure that the file is a tape device.

There is no mapping to column *column_name*, which cannot accept null values.

Cause: The specified column has a NOT NULL constraint, but in the definition of the load map, no field is mapped to the column.

Action: Correct the load map or drop the NOT NULL constraint.

Unable to load locale categories for locale *locale_name*: error *error_num*.

Cause: The GLS locale specified in CLIENT_LOCALE or DB_LOCALE cannot be loaded, or if these variables are not set, the GLS file cannot be loaded.

Action: Check the \$INFORMIXDIR/gls or %INFORMIXDIR%\gls directory to ensure that the locale files are present.

Unload query select item for the *query_item* expression needs to be assigned a name.

Cause: A SELECT statement contains a column name that might not be unique.

Action: Modify the SELECT statement to contain a name for each column expression. For example:

```
SELECT Max(I) Mcol FROM table x
```

Write/read to/from tape until end of device.

Cause: The **onpload** command line option **-Z** enabled write to and read from the tape until the end of the device.

Action: No action is required.

Write to device (tape or pipe) *device_name* failed; no space left on device. AIO error *error_num*.

Cause: The write to tape is failing on a tape device due to lack of space.

Action: Increase the space on the tape device or replace the device and restart the load or unload job.

Pop-Up Messages

Cannot attach to server shared memory.

Cause: If the server is on, a permissions problem exists.

Action: On UNIX, check that the following permissions and ownership of **onpload** are set:

```
-rwsr-sr-x 1 informix informix
```

On Windows, check the permissions of the user running **onpload**.

UNIX

Cannot create shared-memory message queue: error *error_num*.

Cause: A critical initialization error occurred. Probably the UNIX kernel does not have enough shared memory or semaphores configured.

Action: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

Cannot create shared-memory pool: *errno UNIX_error_num*.

Cause: The operating system shared-memory system cannot be accessed.

Action: See your **errno.h** file.

Cannot initialize multithreaded library.

Cause: A critical initialization error occurred. Probably the UNIX kernel does not have enough shared memory or semaphores configured.

Action: Increase shared memory or semaphores. If the condition persists, contact Technical Support. ♦

UNIX

Cannot initialize shared memory: *errno operating-system_error_num*.

Cause: The operating system shared-memory system cannot be accessed.

Action: See your **errno.h** file.

Cannot load X resource.

Cause: The **ipload** utility attempted to display a full-color splash screen image but another process was already using the resources needed for color display.

Action: Use the **ipload -n** option, which does not display a splash screen. ♦

Cannot open.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Cause: An internal error occurred when **onpload** attempted to open the load or unload file.

Action: Press R to try to access the load or unload file again. Press C to skip the file indicated and continue to process the rest of the files. Press Q to stop the job.

Cannot open log file *log_file_name*.

Cause: The log file for the job cannot be opened.

Action: See your **errno.h** file.

Cannot start I/O.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Cause: An internal error occurred when **onpload** attempted to open the load or unload file.

Action: Press R to try to access the load or unload file again. Press C to skip the file indicated and continue to process the rest of the files. Press Q to stop the job.

Fatal error: shared memory will conflict with server.

Cause: The shared-memory segment allocated to **onpload** is located below the shared memory segment of the server, and the size needed to run the job would cause the **onpload** shared memory to overlap the shared memory of the server.

Action: Reduce the size and number of buffers allocated to **onpload** on `$INFORMIXDIR/etc/plconfig` or `%INFORMIXDIR%\etc\plconfig`, or increase the start address for the shared memory location of the server.

Incorrect database version. Make sure that it is upgraded properly.

Cause: Upgrade of **onpload** database failed.

Action: Look in `$INFORMIXDIR/etc/conpload.out` for information on conversion errors.

Press 'r' when ready, 'c' to shutdown device or 'q' to quit.

Cause: The tape device is full.

Action: Mount a new tape device and press `r` to continue the load or unload process. Press `c` to stop the load or unload process on the current drive. Press `q` to stop the load or unload process.

Set the shared library path as an absolute path in the `plconfig` file.

Cause: The full absolute path of the `ipldd09a.so` shared library is not set.

Action: Set the `HPL_DYNAMIC_LIB_PATH` configuration parameter to an absolute path in the `plconfig` file and restart the job.

Tables with blobs cannot be loaded in High-Performance Mode.

Cause: Express mode cannot load tables that contain blobs.

Action: Use Deluxe mode.

Write error.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Cause: An internal error occurred when **onpload** attempted to open the load or unload file.

Action: Press R to try to access the load or unload file again. Press C to skip the file indicated and continue to process the rest of the files. Press Q to stop the job.

Custom Drivers

If your operating system supports dynamic linking of libraries, you can use a *custom driver* to extend the functionality of the HPL to support different file types or access mechanisms. For example, you could implement a custom interface to load data from a structured file, a high-speed communications link, or another program that generates data to be stored in the database.

The **onpload** utility accesses the custom code through the driver name that you assign to the record-format definition. When **onpload** references a record format, the driver that the record format specifies is examined. If the driver name does not match one of the standard drivers (Fixed, COBOL, Delimited), **onpload** looks into the custom-driver function table to find the custom driver.

The custom-driver code reads data into buffers during a load job and writes out buffers during an unload job. By following the coding procedure discussed in this appendix, you can use the parallel I/O facilities of the HPL to manipulate data buffers, or you can use a custom driver to replace the HPL I/O facilities with your own I/O functionality.

Adding a Custom Driver to the onpload Utility

To add a custom driver to **onpload**, you must perform the following tasks:

- Add the custom driver to the **onpload** database.
- Prepare the code for the custom driver.
For instructions, see [“Preparing the Custom-Driver Code” on page H-3](#).
- Build the shared-library file for the custom driver and custom-conversion functions and install the file in the appropriate directory.
For instructions, see [“Rebuilding the Shared-Library File” on page H-5](#).

Adding the Driver Name to the onpload Database

You must add the name of the custom driver to the **onpload** database so that you can select it when you prepare a load or unload job.

To add a driver name to the onpload database

1. Choose a name for your custom driver.
You can select your own name. This section uses the name *your_custom_driver*.
2. Add the name to the **onpload** database.
Choose **Configure→Driver** to add the driver name to the database.
For more information, see [“Using the Drivers Window” on page 5-10](#).
3. If you prepare multiple custom drivers, you must choose a name for each driver and add it to the **onpload** database.

Preparing the Custom-Driver Code

A driver is implemented as a set of functions, referred to as *methods*. The methods enable **onpload** to open, close, read, and write data files. You can create a custom driver that adds more complex functionality to data-file handling of **onpload**.

A custom driver consists of one or more functions that replace the capability of an existing driver method. The custom driver needs to provide all of the methods for a driver, such as OPEN, READ, WRITE, and CLOSE.

To add to the capability of an existing driver method, the custom driver function calls the existing driver method from the custom function before or after any custom processing, as appropriate.

To replace an existing driver method, the custom function provides all processing that is necessary for that function. The custom driver function does not call the existing standard driver functions.

To prepare the custom-driver code, you must prepare the following two files. You can store the files in any convenient directory.

- The *your_custom_driver.c* file contains the functions that provide your user-specific driver functionality. You must provide a function for each driver.
- The *plcstdrv.c* file tells **onpload** where to find the custom-driver functions.

To prepare the file that provides the driver functionality

1. Create a file for the driver functions (for example, *your_custom_driver.c*).
2. Prepare the driver code.

This appendix includes an example of driver files, [“Driver Example” on page H-11](#). Use this example as a template for building your driver code.

The driver methods and API functions that you can use are described in [“Available Driver Methods” on page H-16](#) and [“Available API Support Functions” on page H-17](#).

To prepare the `plcstdrv.c` file

1. Use the following code as a template to create the `plcstdrv.c` file. You can copy a template for `plcstdrv.c` from the `$INFORMIXDIR/incl/hpl` directory.

```
/* ****  
 * Start of plcstdrv.c  
 */  
  
/* plcstdrv.h is in $INFORMIXDIR/incl/hpl */  
#include "plcstdrv.h"  
  
/* Your driver configuration function */  
int your_driver_config_function();  
  
(*pl_get_user_method(driver, method)) ()  
char *driver;  
int method;  
{  
    /*  
     * your_driver_name is the name of your driver  
     */  
    if (strcmp(driver, "your_driver_name") == 0)  
    {  
        /*  
         * If onpload is trying to configure the driver,  
         * return the function that will handle the  
         * initialization.  
         */  
        if (method == PL_MTH_CONFIGURE)  
            return(your_driver_config_function);  
    }  
    /*  
     * YYYY is the name of another driver  
     * This is how additional custom drivers are  
     * configured  
     */  
    if (strcmp(driver, "YYYY") == 0)  
    {  
        if (method == PL_MTH_CONFIGURE)  
            return(YYYY_driver_config_function);  
    }  
}  
/* **** end of plcstdrv.c *****/
```

2. Replace *your_driver_name* with the name of the driver that you chose in step 1 of “[Adding the Driver Name to the onpload Database](#)” on page H-2.
3. Replace *your_driver_config_function* with the function name of the driver-configuration function that you coded in *your_custom_driver.c*.
4. To add multiple custom drivers, repeat the main **if** statement for each driver.

Rebuilding the Shared-Library File

If you use custom drivers or custom conversion functions, you must rebuild the **ipldd09a.SOLIBSUFFIX** shared-library file with your custom-code files. (**SOLIBSUFFIX** is the shared-library suffix for your operating system.)

After you rebuild **ipldd09a.SOLIBSUFFIX**, you must install it in the **\$INFORMIXDIR/lib** directory. To store the shared library in a different location, such as **/usr/lib**, set the **HPL_DYNAMIC_LIB_PATH** configuration parameter to the appropriate path in the **plconfig** file.



Important: For security reasons, you should keep all shared libraries used by the database server in directories under **\$INFORMIXDIR**.

To build the shared-library file

1. Use the following code as a template to prepare a makefile. You can copy a template for the makefile from the `$INFORMIXDIR/incl/hpl` directory.

```
#####  
# Makefile for building onpload shared library  
#  
  
LD = ld  
  
# link flag for building dynamic library, may be different  
# for your platform, see the man page for ld, the link  
# editor.  
LDSOFLAGS = -G  
  
# where to find plcstdrv.h  
INCL = $INFORMIXDIR/incl/hpl  
  
# SOLIBSUFFIX is the shared library suffix for your  
# platform. For Solaris it is "so"  
SOLIBSUFFIX = so  
  
PLCDLIBSO = ipldd09a.$(SOLIBSUFFIX)  
  
.c.o:  
    $(CC) $(CFLAGS) -I$(INCL) -c $<  
  
#  
# plcstdrv.c contains the custom driver table, required  
# plcstcnv.c contains the custom function table, required  
# your_custom_driver.c contains your custom driver  
#   routines, optional  
# your_custom_conversion.c contains your custom  
#   conversion functions, optional  
#  
  
SOOBS = plcsrdrv.o plcsrcnv.o your_custom_driver.o  
your_custom_function.o  
  
solib: $(SOOBS)  
    $(LD) -o $(PLCDLIBSO) $(LDSOFLAGS) $(SOOBS)  
  
##### end of makefile #####
```


2. Include the following files in the makefile.

File	Description
<code>plcstdrv.c</code>	Prepares the custom driver tables
<code>plcstcnv.c</code>	Prepares the custom-conversion function tables
<code>your_custom_driver.c</code>	Contains your user-specific driver functions
<code>your_custom_functions.c</code>	Contains your user-specific conversion functions

[Appendix E, “Custom Conversion Functions”](#) describes the `plcstcnv.c` and the `your_custom_functions.c` files.

3. Run **make** using the makefile.

The makefile builds the shared-library file `ipldd09a.SOLIBSUFFIX`, where `SOLIBSUFFIX` is the shared-library suffix for your operating system.

The HPL uses the same library for both custom-conversion functions and custom drivers, so when you rebuild the library, you must also link the custom-conversion code.

4. Install the shared library in the appropriate path for your operating system.

For example, on Solaris you must install the shared library in the `$INFORMIXDIR/lib` directory, or in the directory specified by the `HPLDYNAMICLIB` configuration variable in the `PLCONFIG` file. You can move `ipldd09a.SOLIBSUFFIX` into the shared-library directory, or you can use a link. For administrative purposes, a link might be clearer.

5. Set the owner and group of the shared-library files to **informix**. Set the permission bits to 0755 (octal).

Tip: When the libraries are updated, the letter before the decimal (here, the letter *a*) changes to indicate that the library has changed. If you do not find `ipldd09a`, look for `ipldd09b` or `ipldd09c`.



Connecting Your Code to onpload at Runtime

When **onpload** determines that a custom driver is required to read or write data for a given record format, it calls the function **pl_get_user_method()**.

The **pl_get_user_method()** function returns the function that the loader should call to perform initial driver configuration before any I/O activity is started. The function that you specify should be a function that you are supplying in your driver. This function should not do any other initialization. The example in the previous section illustrates the coding technique for this initial connection of your driver to **onpload**.

Driver Initialization

The **onpload** utility calls the configure function that you returned in **pl_get_user_method()**, expecting this function to configure all driver methods that are to be customized. The configure function must call the **pl_inherit_methods()** function, specifying the class of driver that is appropriate for the data being processed.

A driver class can be one of following classes.

Driver Class	Description
Fixed	Fixed drivers process data on the assumption that the data is organized as constant length records of the same format. Fields in the record will consistently appear at the same offset in each record.
Delimited	Delimited drivers assume that the record and field boundaries are defined by markers (called delimiters) in the data.
COBOL	The COBOL driver treats data as constant length records in the same manner as the Fixed driver. The distinguishing factor of the COBOL driver is its support for conversion of the ANSI COBOL data types.

After the **pl_inherit_methods()** function is called, you can add additional functions that are called to support open, close, read, and write requirements of the driver. Call **pl_set_method_function()** to tie your driver functions into the **onpload** execution.

Registering Driver Functions

The `pl_set_method_function()` registers a passed function to the passed method ID. For a description of the method IDs (defined in `$INFORMIXDIR/incl/plcustom/pldriver.h`) applicable to your driver implementation, refer to the [“Available API Support Functions” on page H-17](#). The following table shows the available methods.

Method	Description
PL_MTH_OPEN	The function called to open the file of interest for the load/unload
PL_MTH_CLOSE	The function called to close the file at the end of the load/unload
PL_MTH_RAWREAD	The function called to get the next block of data from the load file
PL_MTH_RAWWRITE	The function called to write a block of data that is passed to it

You do not need to register a function for any of the methods IDs (although presumably you register at least one, or there is no point in writing the driver).

Use the **pl_driver_inherit()** function to get the standard function for the passed method. For example, to find and execute the function currently registered for reading data from the load input, you would code as follows:

```
int my_rawread_function(bufptr, bufsize, bytesread)
char *bufptr;
int    bufsize;
int    *bytesread;
{
    int (*funcptr)();
    int rtn;

    funcptr = pl_driver_inherit(PL_MTH_RAWREAD);
    rtn = (*funcptr)(bufptr, bufsize, &bytesread);
    if (rtn)
        return(rtn); /* error */
    /*
     * Now you have a buffer of data in bufptr, of
     * size bytesread. So you can process data in
     * this buffer here before it is converted
     */
    return(rtn);
}
```

For performance reasons, system calls are strongly discouraged. System calls cause the VP on which the driver thread is running to be blocked for the duration of the system call. This blockage prevents the VP from doing other work, causing **onpload** to run less efficiently.

Driver Example

The following section shows an example of custom drivers. The example shows how to code a custom driver that completely takes over the open, close, read, and write responsibility.

Driver Example

The custom driver in this example takes over the open, close, read, and write responsibility. The example illustrates the form of a driver and the necessary initialization, registration, and mechanism of the driver. The coding of user-specific functionality is not represented.

The plcstdrv.c File

Assume that you chose **MYDRIVER** as the driver name and that you added this name to the **onpload** database with **ipload**. The **plcstdrv.c** file is as follows:

```
#include "plcstdrv.h"

int          DrConfig();

(*pl_get_user_method(driver, method)) ()
char         *driver;
int          method;
{
    if (strcmp(driver, "customdrv") == 0)
    {
        if (method == PL_MTH_CONFIGURE)
            return (DrConfig);
    }

    return (0);
}
```

Custom-Driver Code for MYDRIVER

The following driver code supports MYDRIVER:

```
#include <plcstdrv.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/fcntl.h>

extern char *malloc();
extern      errno;

static int DrOpen();
static int DrRead();
static int DrWrite();
static int DrClose();

#define CTRLDELIM 0x01          /* fake delimiter (CTRL-A)      */
#define REALDELIM 0x7c         /* real delimiter (|)           */
#define ESCAPESIGN 0x5c        /* escape sign (\)              */
#define ENDRECORD 0x0a         /* end of record (\n)           */

int      fd;

/*-----
 * DrConfig()
 *
 * Input   : (char *)          driver name
 *           (void *)          method table
 *
 * Return  : PL_RTN_OK
 *
 * Schema  : Fills in the driver table
 *-----*/

int
DrConfig(driver, methodtable)
char      *driver;
void      *methodtable;
{
    pl_inherit_methods("Delimited", methodtable);
    pl_set_method_function(methodtable, PL_MTH_OPEN, DrOpen);
    pl_set_method_function(methodtable, PL_MTH_RAWREAD, DrRead);
    pl_set_method_function(methodtable, PL_MTH_RAWWRITE, DrWrite);
    pl_set_method_function(methodtable, PL_MTH_CLOSE, DrClose);
    pl_lock_globals();

    return PL_RTN_OK;
}
```

```

/*-----
 * DrOpen()
 *
 * Input  : (devicearray *) dev    device array structure
 *
 * Return : PL_RTN_FAIL           error
 *          PL_RTN_OK             open succeeded
 *
 * Schema : Open the specific file for that driver thread
 *          Note that the custom driver thread is bound to its
 *          own CPU VP, therefore it is safe to have globals like fd
 *-----*/

static int
DrOpen(dev)
devicearray *dev;
{
    fd = open(dev->filename, O_RDONLY);
    if (fd < 0)
    {
        return PL_RTN_FAIL;
    }

    return PL_RTN_OK;
}

/*-----
 * DrRead()
 *
 * Input  : (char *) bf           output buffer to write record to
 *          (int)   size          size of output buffer
 *          (int *) count         number of bytes written to output buffer
 *
 * Return : PL_RTN_FAIL           error
 *          PL_RTN_OK             returning buffer
 *          PL_RTN_EOF            returning the last buffer, no more data
 *
 * Schema : Reads from input and fill up data buffer provided by the
 *          caller. Here, the caller expect a record where the delimiter
 *          is |. Our custom driver changes all CTRL-A into | and
 *          escapes the already existing | from input.
 *-----*/

static int
DrRead(bf, size, count)
char *bf;
int size;
int *count;
{
    int rtn; /* return value */
    int n; /* bytes read in */
    static char *bftemp = 0; /* temp buffer */
    char *p; /* pointer to temp buff */
    char *start; /* start of output buffer */
    static off_t currseek = 0; /* current seek in input */
    int escaped = 0; /* did we escape last character */

```

```
start = bf;

if (bftemp == 0)
{
    if ( (bftemp = malloc(size)) == 0 )
    {
        return PL_RTN_FAIL;
    }
}

/*
 * read data in
 */
errno = 0;
do
{
    n = read(fd, bftemp, size);
} while (n == -1 && errno == 4);

rtn = (n < 0) ? PL_RTN_FAIL : (n == size) ? PL_RTN_OK : PL_RTN_EOF;

currseek += n;

p = bftemp;

/*
 * format output buffer
 */
while (size)
{
    if (*p == REALDELIM)
    {
        *bf = ESCAPESIGN;
        escaped = 1;
    }
    else if (*p == CTRLDELIM)
    {
        *bf = REALDELIM;
    }
    else
    {
        *bf = *p;
    }

    size--;
    bf++;

    if (escaped && size)
    {
        *bf++ = *p;
        escaped = 0;
        size--;
    }

    p++;

    if ((int) (p - bftemp) == n)
        break;
}
```



```

    if (escaped)
    {
        p--;
        rtn = PL_RTN_OK;
    }

    if ((int) (p - bftemp) != n)
    {
        currseek -= (off_t) (n - (p - bftemp));
        lseek(fd, currseek, SEEK_SET);
    }

    if (rtn == PL_RTN_EOF)
    {
        *bf = 0;
    }

    *count = (int) (bf - start);

    return rtn;
}

static int
DrWrite(bf, size)
char      *bf;
int       size;
{
    return PL_RTN_OK;
}

static int
DrClose(device)
devicearray *device;
{
    close(fd);

    return PL_RTN_OK;
}

```

Available Driver Methods

The following section describes the methods that you can use to build custom drivers. The methods defined in this section use the return values that are described in the following table.

Return Value	Interpretation
PL_RTN_OK	Method executed successfully
PL_RTN_FAIL	Method did not succeed. Stop processing.
PL_RTN_EOF	Method reached the end of the file

PL_MTH_OPEN

Call this function to open the file of interest for the load or unload.

Function Information	Description of Arguments
Input arguments: devicearray *device;	Device/file pathname to open
Output arguments: None	
Return values: PL_RTN_OK, PL_RTN_FAIL	

PL_MTH_CLOSE

Call this function to close the file at the end of the load or unload.

Function Information	Description of Arguments
Input arguments: None	
Output arguments: None	
Return values: PL_RTN_OK, PL_RTN_FAIL	

Available API Support Functions

This section describes the API support functions that you can use with custom drivers. The methods defined in this section use the return values that are described in the following table.

Return Value	Interpretation
PL_RTN_OK	Function succeeded.
PL_RTN_FAIL n	Function failed.

pl_inherit_methods(driver, methodtable)

This function loads the passed method function table with the functions currently configured for the passed driver.

Function Information	Description of Arguments
Function type:	int
Input arguments:	char *driver void *methodtable
	Name of driver to inherit Pointer to method table
Return values:	PL_RTN_OK, PL_RTN_FAIL

The **onpload** utility is supplied with three standard drivers described in [“Driver Initialization” on page H-8](#).

pl_set_method_function(methodtable, method, function)

pl_set_method_function(methodtable, method, function)

This function inserts the passed function into the method chain.

Function Information		Description of Arguments
Function type:	int	
Input arguments:	void *methodtable	Method table passed to PL_MTH_CONFIGURE
	int method	Method ID
	int (*funcptr)()	Function to insert into method chain
Return values:	PL_RTN_OK, PL_RTN_FAIL	

pl_driver_inherit(method)

This function returns a pointer to the function that currently supports the passed method and advances the method chain so that the next request returns the next function in the method list.

Function Information		Description of Arguments
Function type:	int	
Input argument:	int method	Method ID
Return values:	PL_RTN_OK, PL_RTN_FAIL	

When you inherit a driver and use the **pl_set_method_function** to override a method, you can assume all processing functionality for method.

However, if you want to add to the existing processing functionality, **pl_driver_inherit** returns the function that was installed in the function table prior to the call to **pl_set_method_function**.

Example

You are processing a file in which the context of the data determines the record fields present. You want to analyze the records and reformat the data into a delimited format that the **onpload** data converter recognizes.

In the driver setup, specify that your function should inherit the Delimited driver and add your custom function to the **PL_MTH_READREC** method.

When your function is called, get the inherited function with **pl_driver_inherit()**. Invoke this function, which returns a pointer to the start of a record and its length.

Apply your changes to the data in the buffer that is returned by the call to the inherited function.

pl_get_recordlength()

This function returns the length of the active record format. If the format is for a delimited record type, the value returned is 0.

Function Information		Description of Arguments
Function type:	int	
Return values	length of record	Function succeeded
	PL_RTN_FAIL	Function failed

pl_set_informix_conversion(flag)

When this function is set to 1, it disables data conversion during the load. The data must be formatted in Informix final-form format.

When this function is set to 0, data conversion is enabled.S

Function Information		Description of Arguments
Function type:	int	
Input argument:	int flag	1 = disable, 0 = enable
Return values:	PL_RTN_OK, PL_RTN_FAIL	

pl_lock_globals()

This function ensures global data integrity when you supply a custom driver that uses globally defined variables.

Function Information		Description of Arguments
Function type:	int	
Return values:	PL_RTN_OK, PL_RTN_FAIL	

pl_reset_inherit_chain(method)

This function resets the function inheritance chain to the start of the list. You should need it only if you are implementing recursive processing.

Function Information		Description of Arguments
Function type:	void	
Input arguments:	int method	Method ID
Return values:	none	

Running Load and Unload Jobs on Windows Computers

You can run load and unload jobs on Windows computers using the following methods:

- Prepare and run jobs using the **onpladm** utility on a Windows computer
- Prepare jobs using the **ipload** utility on a UNIX computer, then run them on a database server on a Windows computer

Each of these methods is described below.

Using the onpladm Utility on Windows

You can run all **onpladm** utility commands from the Windows command line. For more information, see [Chapter 17, “The onpladm Utility.”](#)

Running the Run Job or Run Project Commands

To execute the **run job** or **run project** commands with the **onpladm** utility, enter a user name and password in the **Host Information** tab of the **SETNET32** utility of the IBM Informix Client Software Developer's Kit. For more information on the **SETNET32** utility, see the *IBM Informix Client Products Installation Guide*.

If you do not enter the username and user password, you receive the following error:

```
Cannot execute stored procedure start_onpload SQL ERROR -668
```

If you receive this error, the online log file contains the following message:

```
System() command "$INFORMIXDIR/bin/onpload -H hostname -S  
servername -rl -fb" in SPL routine cannot be executed because user  
"username" did not connect with a password.
```

To execute other commands with the **onpladm** utility, you do not have to enter a user name and password.

Running onpladm on UNIX with Dynamic Server Running on Windows

You can execute the Run Job or Run Project commands on the **onpladm** utility on a UNIX computer and create HPL objects in a database server that runs on a Windows computer.

To execute Run Job or Run Project commands on UNIX with a database server on Windows:

1. Make the UNIX computer a trusted host on your Windows computer.
2. Verify that the UNIX computer and Windows computers can connect to each other.

3. Install a database server on the Windows computer.
Verify that the database server is running.
4. Install a database server on the UNIX computer.
Ensure that the database servers on the UNIX and Windows computers are the same version. You do not need to start the database server on the UNIX computer to run the **onpladm** utility.
5. Set the **INFORMIXSERVER** and **INFORMIXSQLHOSTS** environment variables on your UNIX workstation as follows:

```
INFORMIXSERVER WINservername
INFORMIXSQLHOSTS full.sqlhosts.pathname
```

WINservername Name of the database server on the Windows computer

full.sqlhosts.pathname Complete path of the **sqlhosts** file on the UNIX computer (for instance, **\$INFORMIXDIR/etc/sqlhosts.wn**)

Use the **-S** or **-T** **onpladm** utility option to override the **INFORMIXSERVER** environment variable setting.

6. Add the following line to the **\$INFORMIXSQLHOSTS** file on the UNIX workstation:

```
WINservername ontlitcp WINname servicename
```

WINservername Name of the database server on the Windows computer

WINname Name of the Windows computer

servicename Service that the Windows database server uses

7. Use the **onpladm** utility commands to create, configure, delete, describe, list, and modify HPL objects on your Windows database server.
For more information, see [Chapter 17, “The onpladm Utility.”](#)
8. Prepare load and unload jobs with the **onpladm** utility.
For more information, see [“Creating Jobs” on page 17-10.](#)

9. Type the following line into the **.netrc** file in your home directory on a UNIX computer:

```
machine WIN_machinename login username password user_password
```

WIN_machinename Name of the Windows computer

username Your user name

user_password Your user password

Your user name and user password must be valid on the Windows computer.

Preparing Jobs for Windows with the *ipload* Utility

You cannot prepare load and unload jobs for the High-Performance Loader (HPL) on Windows computers. However, you can use the **ipload** utility on UNIX to prepare the load and unload jobs and then use the **onpload** command on Windows to run those jobs.

To use the *ipload* utility on UNIX to prepare jobs for a database server on Windows

1. Make the UNIX computer a trusted host on your Windows computer.
2. Make sure that the UNIX computer can connect to the Windows computer.
3. Install a database server on Windows and make sure that it is running.
4. Install a database server on UNIX. You need not start the database server to run the **ipload** utility.
5. Make sure that the database server installed on UNIX is the same version as the Informix database server on Windows.
6. If the database server on UNIX is running, do *not* modify the environment variables. If the database server on UNIX is *not* running, set the environment variables (from your UNIX workstation) as the following example shows:

```
INFORMIXSERVER WINservername
INFORMIXSQLHOSTS full.sqlhosts.pathname
```

WINservername The name of the database server on Windows

full.sqlhosts.pathname The full pathname of the **sqlhosts** file on your UNIX workstation (for example, **\$INFORMIXDIR/etc/sqlhosts.wn**)

- 7. Add the following line to the \$INFORMIXSQLHOSTS file on the UNIX workstation:

WINservername ontlitcp WINname servicename

<i>WINservername</i>	Name of the database server on the Windows computer
<i>WINname</i>	Name of the Windows computer
<i>servicename</i>	Service that the Windows database server uses

- 8. On UNIX, run **ipload** as the DBA or as user **informix**.
If the database server on UNIX is *not* running (and you set the environment variables correctly), **ipload** immediately connects to the Windows database server. If the database server on UNIX *is* running, **ipload** initially connects to that database server. To connect to the database server on Windows, choose **Configure Server**.
The Connect Server dialog shows two columns: Onload Server and Target Server. Select the database server on Windows in both columns and click **OK**. The **ipload** utility connects to the database server on Windows.
- 9. Create, edit, view, and run HPL jobs on the database server on Windows. After you prepare the load and unload jobs using **ipload**, you can also run the jobs using the **onpload** command on your Windows computer.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

AIX; DB2; DB2 Universal Database; Distributed Relational Database Architecture; NUMA-Q; OS/2, OS/390, and OS/400; IBM Informix[®]; C-ISAM[®]; Foundation.2000[™]; IBM Informix[®] 4GL; IBM Informix[®] DataBlade[®] Module; Client SDK[™]; Cloudscape[™]; Cloudsync[™]; IBM Informix[®] Connect; IBM Informix[®] Driver for JDBC; Dynamic Connect[™]; IBM Informix[®] Dynamic Scalable Architecture[™] (DSA); IBM Informix[®] Dynamic Server[™]; IBM Informix[®] Enterprise Gateway Manager (Enterprise Gateway Manager); IBM Informix[®] Extended Parallel Server[™]; i.Financial Services[™]; J/Foundation[™]; MaxConnect[™]; Object Translator[™]; Red Brick Decision Server[™]; IBM Informix[®] SE; IBM Informix[®] SQL; InformiXML[™]; RedBack[®]; SystemBuilder[™]; U2[™]; UniData[®]; UniVerse[®]; wintegrate[®] are trademarks or registered trademarks of International Business Machines Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Index

A

Active Job window 2-30, 11-8, 11-9, 12-10, 13-9, 13-10, 13-15
 AIO error code 27 12-11
 AIOBUFFERS parameter
 affecting onpload processes 15-11
 description of B-3
 example 15-15
 AIOBUFSIZE parameter
 affecting onpload process 15-11
 description of B-4
 example 15-15
 Alter table schema 15-14
 ALTER TABLE statement
 format to use 13-11
 ANSI compliance
 icon Intro-9
 level Intro-16
 ASCII. *See* Code set.
 Assigning records to devices 6-3
 Autogenerate Unload Components window 2-35, 13-5, 13-7

B

Binary data, format of 5-11
 Binary type, of computer A-10
 BLOB data type. *See* Extended data types.
 Simple large objects
 See also Express-mode limitations.
 Blobs. *See* Simple large objects.
 Block size of tape 16-10, 17-11, 17-63
 Boldface type Intro-7

Browse option
 description of 14-3
 description of fields 14-5
 log file 14-10
 Browsers menu
 description of 3-5
 Logfile option 14-10
 Record option 14-4
 Violations option 14-8
 Buffer size
 I/O, with onpload 16-10
 server stream buffer 16-10
 Bulk loader. *See* High-Performance Loader.
 BYTE data type. *See* Simple large objects.
 Byte
 number to transfer 9-17
 order of binary information 5-7, 5-12
 order, specification of A-10
 size of variables 5-12
 Bytes, size of data type 7-10

C

cadiload threads 1-17
 Carriage returns, in fixed format 7-12
 Case conversion 9-16, 16-13, A-19
 Changing unload job options 12-14
 CHAR data type 15-17
 Character
 case conversion 16-13
 invalid entries 3-8
 set, to modify 7-29, 7-30

- See also* Code set.
 - See also* Case conversion.
 - CLOB data type. *See* Extended data types.
 - COBOL (byte) format
 - setting with onpladm 17-13, 17-32, 17-65
 - used with generate 13-12
 - COBOL format 7-25
 - creating 7-25
 - definition window 7-24
 - records 7-24
 - setting with onpladm 17-13, 17-32, 17-65
 - used with generate 13-12
 - Code set
 - defaults table A-2
 - GLS 5-7, 7-28, 7-30, 10-12
 - ISO 8859-1 Intro-4
 - of data file 5-7
 - of database 5-7
 - setting with onpladm 17-40, 17-41, 17-42, 17-66
 - to modify 7-29, 7-30
 - with delimited formats 7-30
 - with Fixed and COBOL formats 7-28
 - See also* Character.
 - Code-set
 - conversion 10-12
 - Code, sample, conventions for Intro-12
 - Column
 - characteristics of 9-21
 - default values 9-17
 - drop, add, modify 15-14
 - offset, in mapping options 9-17
 - Column Selection window 8-8, 8-9
 - Command line. *See* onpload command.
 - Command-line conventions
 - elements of Intro-10
 - Comment icons Intro-8
 - Commit interval
 - in onpload database A-20
 - load job 12-14
 - Communication configuration file. *See* Configuration parameter.
 - Comparison of express and deluxe modes 15-8
 - Compliance
 - icons Intro-9
 - with industry standards Intro-16
 - Components menu
 - description of 3-5
 - devices 6-5
 - filter 10-6
 - formats 7-5
 - generate 13-14
 - maps 9-7, 9-11
 - query 8-4, 8-12
 - Computer
 - configuration, reorganize 15-13
 - description, modifying 5-11
 - Configuration file
 - conventions B-2
 - HPL 16-10, 16-13, B-1
 - onpload 1-12
 - Configuration parameter
 - AIOBUFFERS B-3
 - AIOBUFSIZE B-4
 - and thread control 15-11
 - CONVERTTTHREADS B-5
 - CONVERTVPS B-6
 - descriptions B-2
 - HPLAPIVERSION B-7
 - HPL_DYNAMIC_LIB_PATH B-8, H-5
 - STRMBUFFERS B-9
 - STRMBUFSIZE B-10
 - See also* each parameter listed under its own name.
 - Configure menu 3-6
 - Configuring ipload 5-3
 - Confirm delete window 3-24
 - Confirmation window
 - delete 3-24
 - file overwrite 8-17
 - Connect Server window 5-4
 - Constraint
 - checking 15-5
 - violations 15-9
 - Contact information Intro-16
 - Conventions,
 - documentation Intro-7
 - Conversion functions, custom E-1
 - Converter threads 1-16, 15-17, 16-13
 - CONVERTTTHREADS parameter
 - affecting onpload process 15-11
 - description of B-5
 - example 15-15
 - CONVERTVPS parameter
 - affecting onpload process 15-11
 - description of B-6
 - example 15-15
 - Copy Data window
 - illustration 3-22
 - using 3-22
 - Copy existing format 3-22
 - cron job 11-4, 12-4
 - Cursor stability isolation level A-19
 - Custom conversion functions E-1
 - Custom driver
 - adding 5-10
 - creating H-1
 - Custom file handling software 5-8
 - Custom input to pipes 5-8
-
- D**
- Data conversion, with
 - onpload 16-7
 - Data file
 - formats supported by ipload 7-3
 - structure of 7-3
 - Data load. *See* Load job.
 - Data masking 9-18
 - Data source, for onpload 16-6, 16-7
 - Data types
 - COBOL 7-24
 - fixed format 7-8
 - values in onpload database A-7
 - Database
 - create for example 2-4
 - creating a project with onpladm 17-62
 - onpload. *See* onpload database.
 - unloading records 11-5
 - Database code set. *See* Code set.
 - Database name, override in
 - onpload 16-13
 - Database server
 - limitations 12-4

- listing attributes with
 - onpladm 17-67
- name. *See* dbservername.
- selecting 5-3
- setting attributes with
 - onpladm 17-66
- target server 11-4, 12-4
- Database Views window 8-18
- Data, unload using onpload 16-7
- DATE data type 15-17
- DB-Access utility Intro-5
- DBDELIMITER environment
 - variable 7-31
- DBONPLOAD environment
 - variable 1-12
 - description of 1-12
 - mentioned 5-5
- DB_LOCALE environment
 - variable 5-7
- Debugging information A-18
- DECIMAL data type 15-17
- Decimals 7-10
- Default
 - changing the values 5-7
 - column values 9-17
 - data code set 5-7
 - HPL database name 5-5
 - machine type 5-7
 - name of log file 14-9
 - name of rejects file 14-9
 - project name 4-3
 - server name 5-6
 - settings for onpload 5-5
 - used as server name 5-6
 - values in onpload database A-2
 - values, for example 2-6
- Default locale Intro-4
- defaults table, in onpload
 - database A-2
- Defaults window 3-18, 5-6
- Define format
 - delimited records 7-19, 7-25
 - editing a format 7-10
 - fixed-length records 7-5
 - modifying a format 7-28
- Define mapping options 9-15
- definition window 7-25
- Delete existing format 3-23

- Delimited format
 - creating 7-19, 7-25
 - in formats table A-8
 - modifying options 7-30
 - setting with onpladm 17-13, 17-31, 17-65
 - using simple large objects 7-21
- Delimited Format window 2-16, 7-21, 7-23
- Delimited record, definition of 7-19
- Delimiter characters
 - changing 7-30
 - description 7-19
- Delimiter Options window 7-31
- delimiters table, in onpload
 - database A-3
- Deluxe mode
 - choosing from load options 12-14
 - compare to express 15-8
 - description 1-7
 - description of 15-4
 - INSERT statement 15-4
 - mentioned 15-3
 - without replication 15-4
- Demonstration databases Intro-5
- Dependencies, software Intro-4
- Device array
 - definition of 6-3
 - device types 6-3, 6-6
 - editing 6-9
 - elements of, in onpload
 - database A-3
 - example 2-10
 - improving performance 15-13
- onpladm
 - creating 17-26
 - deleting 17-29
 - describing 17-27
 - listing all in a project 17-28
 - modifying 17-27
 - using with 17-12, 17-19
 - speed of 15-13
 - tape parameters 6-8
 - use with onpload 16-6
- Device Array Selection
 - window 2-11, 3-7, 6-4
- device table of onpload
 - database A-3

- Device-array definition
 - window 2-12, 3-10, 6-6, 6-8
- Dirty read isolation level 11-12
- Documentation notes Intro-14
- Documentation, types of Intro-13
 - documentation notes Intro-14
 - machine notes Intro-14
 - release notes Intro-14
- Driver
 - class, format type of 5-10
 - creating custom driver H-1
 - modifying 7-29
 - name of custom driver 5-10
- Drivers window
 - using 5-10, 5-13

E

- EBCDIC data, generating 13-12
- Editing a format 7-10
- Environment variable
 - DBDELIMITER 7-31
 - DBONPLOAD 1-12
 - DB_LOCALE 5-7
 - INFORMIXDIR 1-11
 - INFORMIXSERVER 1-11
 - LD_LIBRARY_PATH 1-11
 - ONCONFIG 1-11
 - PLCONFIG 1-12
 - PLOAD_SHMBASE 1-13
- en_us.8859-1 locale Intro-4
- Error code 27 12-11
- Errors
 - constraint violations 15-10
 - maximum number
 - allowed 11-12, 12-14, 16-10
 - onpladm utility 17-9
 - See also* Log file.
- Exporting a query 8-13
- Express mode
 - choosing from load options 12-14
 - compare to deluxe 15-8
 - described 1-7
 - foreign key constraints 15-7
 - level-0 backup 12-11
 - load example 15-15
 - mentioned 15-3
 - page size limitation 15-5

sequence of events 15-6

Extended data types

BLOB 1-14, 15-5

CLOB 1-14, 15-5

F

Fast format 7-27

Fast Job Startup window 13-15

Fast job, definition of 7-27

Feature icons Intro-9

Features of this product,
new Intro-6

Field

set minimum and/or

maximum 9-17

set offset 9-17

Figure

extracting data from a table 8-3

foreign-key constraints 15-7

load and unload modes 15-3

mapping options symbol 9-16

maps found by a search 9-25

use of a map 9-3

using OK and Cancel 3-31

view indicator 9-19

File

COBOL 7-24

configuration for HPL B-1

default onpload

configuration 1-12

.flt 15-9

import/export queries 8-13

ipldd09a.so B-8, E-4, H-5

onpload.std 1-12

pathname for I/O 16-5

plconfig 16-13

sqlhosts. *See* sqlhosts file.

File descriptor, COBOL 7-26

Fill character, mapping

options 9-17

Filter

conversion of code set 10-12

creating 10-6

defining 10-5

description of 10-3

editing 10-9, 10-10

example 10-4

match conditions 10-8

mode, with constraints 15-10

onploadm

creating 17-50

deleting 17-54

describing 17-52

listing all in project 17-52

modifying 17-51

onpload database A-5

rejected records 15-9

Filter Views window 10-11

filteritem table, in onpload

database A-5

filters table, in onpload

database A-6

Filters window 10-6

Find button 9-19

Find Node window 9-20

finderr utility Intro-15

Fixed binary format

format type group 13-12

setting with onploadm 17-13,
17-32, 17-65

Fixed format

data types 7-8

definition of 7-4

in formats table A-8

setting with onploadm 17-13,

17-31, 17-65

using carriage returns 7-12

using simple large objects 7-13,
7-16

Fixed Format definition

window 7-6, 7-12, 7-14, 7-17

Fixed Format edit window 7-6

Fixed Format Options

window 7-29

Fixed internal format

mentioned 7-27

setting with onploadm 17-13,

17-31, 17-65

used in generate 13-11

fixed-length 7-3

FLOAT data type 15-17

.flt file 15-9

-fn 7-27

Foreign key constraints 15-7

Format 7-3

COBOL 7-24, 13-12

copy 3-22

create 7-5

definition of 7-3

delete 3-23

delimited 7-3

fast job 7-27

fast, definition 7-27

fixed internal 13-11

onploadm

creating 17-39

deleting 17-46

describing 17-44

listing all in project 17-44

modifying 17-43

performance 15-17

steps for editing 7-10, 7-11

testing 14-3

types supported by HPL 7-3

types used by generate 13-11

Format Views window 2-14, 3-16,
3-17, 7-32

formatitem table, in onpload
database A-6

formats table, in onpload
database A-8

Fragmenter threads 1-17

Function

custom conversion E-1

user-defined in mapping
options 9-18

G

Generate

assumptions 13-13

description of 13-3

EBCDIC data 13-12

format types 13-11

from unload job 13-6

no-conversion job 13-14

types of tasks 13-3

Generate window 13-10, 13-14

Global Language Support

(GLS) Intro-4, 5-7, 7-28, 7-30,
10-12

GLS code set. *See* Code set.

GLS. *See* Global Language Support
(GLS).

H

Help Intro-13
 menu description 3-6
 using online help 3-31
High-Performance Loader
 configuration file B-1
 managing 15-3
 modes 1-7, 15-3
 usage models 15-13
HPL configuration file 16-10, 16-13
HPL main window 2-5
HPLAPIVERSION configuration
 parameter B-7
HPL. *See* High-Performance
 Loader.
HPL_DYNAMIC_LIB_PATH
 configuration parameter B-8,
 H-5

I

Icons
 compliance Intro-9
 feature Intro-9
 Important Intro-8
 platform Intro-9
 product Intro-9
 Tip Intro-8
 Warning Intro-8
Important paragraphs, icon
 for Intro-8
Importing a query 8-13
Import/Export File Selection
 window 8-14
Industry standards, compliance
 with Intro-16
Informix internal format. *See*
 Internal format.
\$INFORMIXDIR/etc/sqlhosts. *See*
 sqlhosts file.
INFORMIXDIR environment
 variable 1-11
INFORMIXDIR/bin
 directory Intro-5
INFORMIXSERVER environment
 variable 1-11
Input, starting record 16-11

INSERT statement 15-4
INT data type 15-17
Internal format
 limitations 7-27
 use with generate 13-14
Invalid characters in entry
 fields 3-8
ipldd09a.so file B-8, E-4, H-5
ipload utility
 command 2-5, 3-3
 configuration 5-3
 purpose of 1-8
ISO 8859-1 code set Intro-4
Isolation levels
 committed 11-12
 cursor stability 11-12
 in unload option 11-12
 repeatable read 11-12
 setting with onpladm 17-16
I/O
 buffer size 16-10
 number of tapes to load 16-11
 tape block size 16-10

J

Jobs
 conversion 17-10
 no-conversion 17-10
Jobs menu, description of 3-5
Justification of data in mapping
 options 9-16

L

language table, in onpload
 database A-9
LD_LIBRARY_PATH environment
 variable 1-11
Least significant byte 5-12
Level-0 backup in express
 mode 2-31, 15-6
Limitations, database server 12-4
Load and unload session
 maximum errors 16-10
Load data
 with onpladm 17-12, 17-19
 with onpload 16-7

Load job

 browsing options 14-3
 changing options 12-15
 commit interval 12-14
 components 12-3
 creating 12-7, 12-16
 description of 12-3
 device-array speed 15-13
 example 2-7
 from the command line 16-3
 generate violations records 12-14
 log file 14-9
 maximum errors 12-14
 mode options 12-14
 multiple jobs 11-4, 12-4
 number of records 12-14
 onpladm
 conversion 17-10
 creating for every table 17-62
 creating no-conversion 17-17
 deleting 17-25
 describing 17-22
 listing all in project 17-23
 modifying 17-21
 running 17-24
 running on Windows I-2
 onpload database A-17
 preview records 14-3
 run example 2-29, 2-37
 running 12-10, 12-16
 server considerations 11-4, 12-4
 starting record 12-14
 status log 12-10
 tapes, number of 12-14
 using cron 11-4, 12-4
Load Job Select window
 command line information 12-13
 illustration 2-8, 12-8
Load Job window 2-9, 2-18, 2-27,
 3-13, 12-9
Load log, examining 14-9
Load map
 how to create 9-6
Load Options window 2-28, 12-15
Load Record Maps window 2-22
Locale Intro-4
 default Intro-4
 en_us.8859-1 Intro-4

Log file
 created by ipload 12-7
 for load job 14-9
 messages 12-11, G-2
 sample entry 14-11
 setting 11-8
 Lowercase conversion 9-16, 16-13
 LSB 5-12

M

Machine notes Intro-14
 Machine type
 default 5-7
 modifying 5-11, 7-29
 onpladm
 creating 17-58
 deleting 17-61
 describing 17-60
 listing all 17-61
 modifying 17-59
 machines table, in onpload
 database A-10
 Machines window
 illustration 5-12
 using 5-13
 Managing the HPL 15-3
 Map
 columns and fields of same
 name 9-6, 9-12
 definition of 9-3
 for blobs in separate files 7-15,
 7-18
 for in-line blobs 7-15, 7-18
 onpladm
 creating 17-29
 deleting 17-37
 describing 17-38
 listing all in project 17-39
 modifying 17-38
 Map Views window 2-21, 9-23, 9-24
 Map-definition window 2-24, 2-25,
 2-26, 9-4, 9-9, 9-12, 9-14, 9-19
 Map-edit window
 description of 9-5
 purpose 9-5
 using the find button 9-19
 mapitem table, in onpload
 database A-11
 mapoption table, in onpload
 database A-12
 Mapping options
 bytes to transfer 9-17
 case conversion 9-16
 column offset 9-17
 default column value 9-17
 defining 9-15
 field minimum and/or
 maximum 9-17
 field offset 9-17
 fill character 9-17
 function, user-defined 9-18
 justification 9-16
 picture format 9-18
 steps to define 9-15
 summary 9-14
 symbol 9-16
 Mapping Options window 9-15
 maps table, in onpload
 database A-13
 Masking data 9-18
 Match conditions
 definition of 10-3
 in a filter 10-8
 of WHERE clause 8-12
 Maximum number of errors 11-12
 Message log
 categories of messages G-3
 Message log file
 pathname in onpload 16-14
 Message window 3-19
 Mode options, load job 12-14
 Modify format
 COBOL 7-28
 delimited 7-30
 fixed 7-28
 MONEY data type 15-17
 Most significant byte 5-12
 MSB 5-12
 Multiple load or unload jobs 11-4,
 12-4

N

New features of this
 product Intro-6
 No-conversion job
 changing computer
 configuration 15-13
 definition of 7-27
 load example 15-15
 option 13-14
 restrictions 7-27
 run fast job 13-15
 running the job 13-14
 using onpladm 17-10
 using onpload 16-7
 Non-printable field delimiter
 values A-3
 NOT NULL violation 15-9
 Notes window 3-25
 null UDT data 7-9
 Number of conversion
 threads 16-13
 Number of records in a load
 job 12-14
 Number of records to process
 assigned in onpload 16-11
 Number of tapes to load 16-11

O

ONCONFIG environment
 variable 1-11
 ONCONFIG file parameters. *See*
 Configuration parameter.
 Online help Intro-13
 how to use 3-31
 menu 3-6
 Online manuals Intro-13
 onpladm command
 -B option 17-11
 -F option 17-7
 -f option 17-12
 -M option 17-11, 17-63
 -n option 17-11
 -P option 17-63
 -R option 17-22
 -S option 17-7, 17-11

setting format type 17-13, 17-31, 17-64
 setting run mode 17-12
 specification file syntax 17-7
 -T option 17-11
 -Z option 17-24
 -z option 17-13, 17-31, 17-64
 onpload utility
 creating no-conversion jobs 17-17
 creating onpload database 17-5
 database project, creating 17-62
 defining jobs 17-10
 described 17-5
 device arrays, defining 17-26
 error handling 17-9
 features of 17-5
 filters, defining 17-50
 formats, defining 17-39
 machine types, defining 17-58
 maps, defining 17-29
 projects, defining 17-55
 queries, defining 17-46
 specification file
 conventions 17-6
 target server attributes,
 listing 17-67
 target server attributes,
 setting 17-66
 using on Windows I-2
 onpload
 configuration file 1-12
 load data 16-7
 starting utility 1-9
 onpload command
 -d option 16-8
 description 1-8
 generated for Load Job 12-13
 generated for Unload Job 11-11
 -i option 16-11
 -M option 15-17
 starting 16-3
 syntax 16-4
 usage 16-3
 -Z option 16-5
 onpload database
 connection to 5-3
 creating 5-5
 creating with onploadm 17-5
 creation of 2-5

default settings 5-5
 description 1-9
 multiple 1-12
 select server 5-3
 table
 defaults A-2
 delimiters A-3
 device A-3
 filteritem A-5
 filters A-6
 formatitem A-6
 formats A-8
 language A-9
 machines A-10
 mapitem A-11
 mapoption A-12
 maps A-13
 progress A-15
 query A-16
 session A-17
 table descriptions A-1
 onpload utility, architecture 1-14
 onpload utility. *See* onpload
 command.
 onpload.std file 1-12
 onstat utility 15-12
 Options
 load job 12-14
 unload job 11-12
 Options symbol 9-16
 Organization of a Record that
 Includes In-Line TEXT
 Data 7-16

P

Page size in express mode 15-5
 Parallel loader. *See* High-
 Performance Loader.
 Performance
 converter threads 15-17
 hints 15-16
 improving 15-10
 VPs 15-17
 Permissions. *See* Privileges.
 Picture description, COBOL 7-26
 Picture format, in mapping
 options 9-18

Pipe
 device arrays 5-8
 in a device array 6-7
 starting command 6-3
 use with onploadm 17-12, 17-19
 use with onpload 16-7
 Platform icons Intro-9
 plconfig configuration file
 HPL value override 16-10, 16-13
 See also Configuration file;
 Configuration parameters.
 PLCONFIG environment
 variable 1-12
 PLCONFIG file
 override I/O buffer size 16-10,
 16-13
 PLOAD_SHMBASE
 avoiding shared memory
 collision 1-13
 environment variable 1-13
 pl_wkr threads 1-17
 Print button 3-27
 Privileges 12-5
 Problems during a load job 11-10,
 12-11
 Product icons Intro-9
 Project
 creating a new project 4-7
 default name 4-3
 onploadm
 creating 17-55
 creating for all tables 17-62
 deleting 17-57
 listing all 17-57
 running all jobs 17-56
 Project name, in onpload 16-5,
 17-45, 17-55
 project table, in onpload
 database A-15
 Projects window 4-6
 Proper-name case conversion 16-13
 Proper-name conversion 9-16

Q

Query
 description of 8-3
 export to a file 8-13, 8-16

for unload map 9-10
import from a file 8-13
onploadm
 creating 17-46
 deleting 17-50
 describing 17-48
 listing all in project 17-48
 modifying 17-47
steps for defining 8-4
 using the Table button 8-7
query table, in onpload
 database A-16
Query window 8-5
Query-definition window 8-6, 8-10,
 8-11
Quiet, suppress output A-18

R

Raw load and unload 7-27, 13-14
Reading to end of device 12-12,
 16-5, 17-24
Record Browser window 14-4, 14-5
Record filter 16-13
Record Formats window 2-15, 3-26,
 7-5
Record map, assigned by
 onpload 16-5
Records, number to process 16-11
Rejected records 15-9
 conversion errors 15-9
 filter conditions 15-9
 reviewing 14-6
Release notes Intro-14
Reorganize computer
 configuration 15-13
Repeatable read isolation
 level 11-12

S

Sample-code conventions Intro-12
Schema, of database table 13-3
Screen-illustration
 conventions Intro-12
sdriver threads 1-16
SELECT clause, preparing 8-6
Selection type 3-8

Server name, default 5-6
Server. *See* Database server.
session table, in onpload
 database A-17
SET CONSTRAINTS statement
 DISABLED 15-7
 ON 15-10
setrw threads 1-17
Simple large objects
 as inline data 7-13, 7-16
 BYTE data type 15-5
 in delimited records 7-21
 in fixed format 7-13, 7-16
 in separate files 7-15, 7-18
 TEXT and BYTE 15-5
Single CPU, performance B-6
Smart large objects 15-5
SMFLOAT data type 15-17
Software dependencies Intro-4
Specification file for onploadm
 conventions 17-6
 conversion job, creating 17-13
 device arrays, creating 17-26
 device arrays, modifying 17-27
 filters, creating 17-50
 filters, modifying 17-51
 formats, creating 17-39
 formats, modifying 17-43
 jobs, modifying 17-21
 machine types, creating 17-58
 machine types, deleting 17-61
 machine types, modifying 17-59
 maps, creating 17-32
 maps, modifying 17-38
 no-conversion jobs,
 creating 17-19
 queries, creating 17-46
 queries, modifying 17-47
 syntax 17-7
Specifications window 9-21
Specs button 3-21, 9-21
Splash screen 2-5, 3-3
SQL code Intro-12
SQL query. *See* Query.
SQL statement, use in HPL 8-3
sqlhosts file 5-4
Start record for input 16-11
Start record, for load job 12-14
Statistics. *See* onstat utility.

Status log. *See* Log file.
Steps
 to add a custom driver name 5-10
 to change load job options 12-15
 to create a device array 6-5
 to create a fixed-length format 7-5
 to create a load job 12-8, 12-16
 to create a load map 9-7
 to create an unload map 9-11
 to define a filter 10-6
 to define a query 8-4
 to define fixed-length file 7-5
 to define mapping options 9-15
 to edit a device array 6-9
 to edit a filter 10-9, 10-10
 to edit a format 7-10, 7-11
 to generate components
 menu 13-12
 to modify delimited formats 7-31
 to modify format options 7-29
 to review rejected records 14-7
 to review source records for
 load 14-4
 to select a database server 5-4
 to specify onpload defaults 5-7
 to use Map Views window 9-23
 to use Specifications
 window 9-21
 to use the Fast Job window 13-15
 to view load log 14-10
 to view violations table 14-8
stores_demo database Intro-5
Stream threads 1-17
STRMBUFFERS parameter
 affecting onpload process 15-11
 description of B-9
 example 15-15
STRMBUFSIZE parameter
 affecting onpload process 15-11
 description of B-10
 example 15-15
Structured query language. *See* SQL
 statement.
superstores Intro-5
superstores_demo database Intro-5
Suppress message output A-18
Swap bytes 16-10, A-19
Symbol, mapping options 9-16
Synonyms 1-4, 8-7, 9-8

Syntax, onpload utility 16-3
 System requirements
 database Intro-4
 software Intro-4

T

Table
 create for example 2-4
 descriptions A-1
 Tape I/O threads 1-16
 Tape parameters, specifying 6-8
 Tape size
 setting 6-8
 Tapes
 block size 16-10
 number of 12-14, A-20
 number to load 16-11
 reading to end 12-12, 16-5, 17-24
 writing to end 11-10, 16-5, 17-24
 Target server 5-3, 11-4, 12-4
 Testing formats 14-3
 TEXT data type 15-5
 Thread
 cadiload 1-17
 convert 1-16
 fragmenter 1-17
 pl_wkr 1-17
 sdriver 1-16
 setrw 1-17
 stream 1-17
 tape I/O 1-16
 ulstrm 1-22
 unload-stream 1-22
 worker 1-16
 Tip icons Intro-8
 Trace level A-18
 Transfer bytes, in mapping
 options 9-17

U

ulstrm threads 1-22
 Unload data
 using onpload 16-7, 17-12, 17-19
 Unload job
 changing the options 11-12
 components of 11-3

 creating 11-5
 definition of 11-3
 example 2-32
 from the command line 16-3
 generate option 13-6
 log file 14-9
 multiple jobs 11-4, 12-4
 null UDT data 11-8
 onpladm
 conversion 17-10
 creating for every table 17-62
 creating no-conversion 17-17
 deleting 17-25
 describing 17-22
 listing all in project 17-23
 modifying 17-21
 running 17-24
 running on Windows I-2
 onpload database A-17
 options 11-12, 12-14
 options, how to change 12-14
 using cron 11-4
 Unload Job Select window
 command line information 11-11
 illustration 11-6
 Unload Job window 2-34, 2-36,
 11-7, 13-8
 Unload map
 definition of 9-10
 how to create 9-10
 steps to create 9-11
 Unload Options window 11-13
 Unload Record Maps window 9-11
 Uppercase conversion 9-16, 16-13
 Usage description, COBOL 7-26
 Usage models for HPL 15-13
 Users, types of Intro-4
 Utility, onpladm. *See* onpladm
 utility.
 Utility, onpload. *See* onpload
 command.

V

VARCHAR data type 15-17
 Variable, binary size of 5-12
 View icon, description 3-28
 View indicator, figure 9-19

Views 1-4, 8-7
 Violations table
 generate from load job 12-14
 viewing 14-7
 Violations Table Browser
 window 14-8, 14-9
 Violations, description of 15-9
 VPs, performance 15-17

W

Warning icons Intro-8
 WHERE clause
 match conditions 8-12
 preparation 8-11
 Whitespace in configuration
 file B-2
 Window
 Active Job 2-30, 11-9
 Autogenerate Unload
 Components 2-35, 13-5, 13-7
 Browse Logfile 14-10
 COBOL Format definition 7-24,
 7-25
 Column Selection 8-8, 8-9
 confirm delete 3-24
 Confirm file-overwrite 8-17
 Connect Server 5-4
 Copy Data 3-22
 Database Views 8-18
 Defaults 3-18, 5-6
 Delimited Format 2-16
 Delimited Format definition 7-21,
 7-23
 Delimiter Options 7-31
 Device Array Selection 2-11, 3-7,
 6-4
 device-array definition 2-12, 3-10,
 6-6, 6-8
 Fast Job Startup 13-15
 Filter Views 10-11
 Filters 10-6
 Find Node 9-20
 Fixed Format 7-6
 Fixed Format definition 7-12,
 7-14, 7-17
 Fixed Format definition
 window 7-6

- Fixed Format Options 7-29
- format definition 2-16
- Format Views 2-14, 3-16, 3-17, 7-32
- Generate 13-10, 13-14
- HPL main window 2-5
- Import/Export File Selection 8-14
- Load Job 2-9, 2-18, 2-27, 3-13, 12-9
- Load Job Select 2-8, 12-8, 12-13
- Load Options 2-28, 12-15
- Load Record Maps 2-22
- Machines 5-12
- map definition 2-24, 2-25, 2-26, 9-4, 9-9, 9-12, 9-14, 9-19
- Map Views 2-21, 9-23, 9-24
- Mapping Options 9-15
- Message 3-19
- notes 3-25
- Projects 4-6
- Query 8-5
- query definition 8-6, 8-11
- Query-definition 8-10
- Record Browser 14-4, 14-5
- Record Format 2-15, 3-26, 7-5
- Specifications 9-21
- Unload Job 2-34, 2-36, 11-7, 13-8
- Unload Job Select 11-6
- Unload Options 11-13
- Unload Record Maps 9-11
- Violations Table Browser 14-8, 14-9
- Windows, using onpladm utility I-2
- Worker threads 1-16
- Writing to end of device 11-10, 16-5, 17-24

X

- X/Open compliance level Intro-16