

IBM Informix
Version 11.50

*IBM Informix High-Performance Loader
User's Guide*



IBM Informix
Version 11.50

*IBM Informix High-Performance Loader
User's Guide*



Note:

Before using this information and the product it supports, read the information in "Notices" on page L-1.

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 publication 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 IBM Corporation 1996, 2010.

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

Contents

Introduction	xi
In this introduction	xi
About this publication	xi
What's New in the High Performance Loader for Informix, Version 11.50	xi
Types of users	xii
Software dependencies	xii
Assumptions about your locale.	xii
Demonstration databases	xiii
Example Code Conventions.	xiii
Additional Documentation	xiv
Compliance with Industry Standards	xiv
Syntax Diagrams	xiv
How to Read a Command-Line Syntax Diagram	xv
Keywords and Punctuation	xvi
Identifiers and Names	xvii
How to Provide Documentation Feedback	xvii
Chapter 1. High-Performance Loader overview	1-1
Overview of HPL features	1-1
The HPL data-load process	1-2
The HPL data-unload process	1-3
HPL loading modes	1-4
The HPL express mode	1-4
The HPL deluxe mode	1-4
HPL components	1-4
onpload utility	1-4
ipload utility.	1-5
onpladm utility	1-5
onpload database	1-5
Relationships among the parts of the HPL	1-5
Environment variables needed for the HPL	1-6
DBONPLOAD environment variable.	1-7
PLCONFIG environment variable.	1-7
PLOAD_SHMBASE environment variable	1-8
PLOAD_LO_PATH environment variable	1-8
PLOAD_SHMAT environment variable	1-8
Architecture of the onpload utility	1-8
The onpload utility deluxe-mode process	1-9
The onpload utility express-mode load process	1-10
The onpload utility unload process	1-12
Chapter 2. Getting started through examples of using the ipload utility	2-1
Examples of loading and unloading jobs using the ipload utility	2-1
Preparing to use the ipload utility	2-2
Creating a file of data.	2-2
Creating a database	2-2
The ipload utility	2-2
Starting ipload	2-2
Choosing a project.	2-3
Checking the ipload utility default values	2-3
Load Job windows.	2-4
Load Job Select window	2-4
Load Job window	2-6
Device-Array windows	2-7
Device Array Selection window	2-7

Device-Array Definition window	2-7
Format windows	2-8
Format Views window	2-8
Record Formats window	2-9
Format-Definition window	2-11
Filter, Discard Records, and Logfile text boxes	2-12
Filter text box	2-12
Discard Records text box	2-12
Logfile text box	2-13
Map Views window	2-13
Creating a map by using Map Views window	2-13
Load Record Maps window	2-14
Map-Definition window	2-15
Load Options window	2-18
Running the ipload job	2-19
Active Job window	2-19
Verifying the transfer of data	2-20
Performing a level-0 backup	2-20
ipload utility Generate options	2-21
Using the information you created with the ipload example	2-21
Preparing the Unload-Job window	2-21
Performing the unload job	2-25

Chapter 3. Using the ipload utility windows 3-1

Use the ipload utility GUI or the onpladm command-line interface	3-1
Starting the ipload utility	3-1
The ipload utility GUI	3-2
HPL main window	3-2
Component-selection windows	3-3
Component-definition windows	3-5
Load job and unload job windows	3-6
Views windows	3-7
Selection-list windows	3-9
Message windows	3-10
Using the HPL ipload utility buttons	3-11
HPL ipload utility toolbar buttons	3-11
HPL ipload utility icon buttons	3-16
Buttons at the bottom of the HPL ipload utility display	3-18
Using the HPL online help	3-18
Using UNIX keyboard commands to move the cursor	3-19

Chapter 4. Defining HPL projects 4-1

HPL projects	4-1
Project organization	4-1
Select or create a project with the Projects window	4-3
Creating a new project	4-4
Selecting a project	4-4

Chapter 5. Configuring the High-Performance Loader 5-1

Configuring ipload	5-1
Selecting a database server	5-1
Using the Connect Server window	5-1
Creating the onpload database	5-2
Modifying the onpload default values	5-2
Defaults window	5-2
Changing the onpload defaults	5-4
Selecting a driver	5-4
Drivers window	5-4
Using the Drivers window	5-5
Modifying the machine description	5-6

Machines window	5-6
Using the Machines window	5-7
Chapter 6. Defining device arrays	6-1
Device arrays	6-1
Using multiple devices in a device array	6-1
Using the Device-Array Selection window	6-1
Using the Device-Array Definition window	6-2
Adding, editing, and removing devices	6-4
Chapter 7. Defining formats	7-1
Formats	7-1
Formats of supported data-file records	7-2
Fixed-length records	7-2
Creating a fixed format	7-2
Editing a format	7-6
Creating a fixed format that uses carriage returns	7-7
Creating a fixed format that includes BYTE or TEXT data.	7-8
Creating a fixed format that includes Ext type or simple LO data.	7-10
Delimited records.	7-12
Creating a delimited format	7-12
Creating a delimited format that includes BYTE or TEXT data.	7-13
Creating a delimited format with extended data types	7-14
COBOL records	7-15
Creating a COBOL format	7-16
Picture and usage descriptions	7-17
Other formats	7-18
Fast format	7-18
Fast job	7-18
Format options	7-18
Modifying fixed and COBOL formats	7-18
Modifying delimited-format options	7-19
Format Views window	7-22
Chapter 8. Defining queries	8-1
HPL queries	8-1
Creating a query	8-1
Using the Table button	8-3
Editing the WHERE clause	8-6
Editing a query	8-7
Exporting and importing queries	8-8
Importing a query	8-8
Exporting a query	8-9
Database Views window	8-10
Chapter 9. Defining maps	9-1
Load and unload maps	9-1
Using the map-definition window	9-2
Creating a load map	9-3
Unload maps	9-5
Creating an unload map	9-5
Unloading data using functions	9-7
Mapping options	9-8
Using the mapping options	9-8
Setting the mapping options	9-9
Editing options	9-10
Using the Delete button.	9-11
Using the Find button	9-11
Using the Specs button	9-12
Map Views window	9-13

Chapter 10. Defining filters	10-1
Filters	10-1
Using a filter	10-1
Creating a filter	10-2
Editing a filter	10-5
Filter views	10-6
Filters with code-set conversion (GLS)	10-7
Chapter 11. Unloading data from a database.	11-1
Unload jobs	11-1
Components of the unload job	11-1
Choosing the database server	11-2
Running multiple jobs	11-2
Unload Job windows	11-2
Creating an unload job	11-2
Running the unload job	11-5
Specifying to write to the end of the tape	11-6
Using the command-line information	11-6
Changing the unload options	11-7
Editing an unload job	11-7
Generate options for an unload job	11-8
Chapter 12. Loading data to a database table	12-1
Load jobs	12-1
Components of the load job	12-1
Choosing the database server	12-1
Running multiple jobs	12-2
Preparing user privileges and the violations table	12-2
Load Job windows	12-3
Creating a load job	12-4
Running the load job	12-6
Specifying to read to the end of the tape	12-6
Using the command-line information	12-7
Changing the load options	12-8
Editing a load job	12-9
Generate options for a load job	12-9
Chapter 13. The Generate options of the ipload utility	13-1
Overview of the ipload Generate options	13-1
Tasks that generate load or unload components	13-1
Generating from the Load Job window	13-1
Using the Autogenerate Load Components window	13-2
Generating from the Unload Job window	13-3
Using the Autogenerate Unload Components window	13-3
Generating from the Components menu	13-5
Generate window	13-5
Generating load and unload components	13-7
Using the No Conversion Job option	13-8
Chapter 14. HPL browsing options	14-1
Browsing options	14-1
Previewing data-file records	14-1
Reviewing records that the conversion rejected	14-3
Viewing the violations table	14-4
Viewing the status of a load job or unload job	14-5
Chapter 15. Managing the High-Performance Loader	15-1
Manage modes, errors, and performance	15-1
HPL modes	15-2

The HPL deluxe mode	15-2
The HPL express mode	15-2
How the express and deluxe modes work	15-3
HPL load and unload errors	15-5
Rejected records from the input file.	15-5
Constraint violations.	15-5
Viewing error records	15-6
HPL performance.	15-6
onpload configuration parameters	15-6
Express-mode limitations	15-7
onstat options for onpload.	15-7
Devices for the device array	15-8
HPL usage tasks	15-8
Settings for a no-conversion load or unload job.	15-9
An express-mode load with delimited ASCII	15-10
HPL performance hints	15-10
Limitation when using the Excalibur Text DataBlade Module indexes	15-12

Chapter 16. The onpload utility 16-1

Overview of the onpload utility	16-1
onpload filename size limitations on UNIX	16-1
Starting the onpload utility	16-2
Using the onpload utility	16-2
The onpload utility syntax.	16-2
Setting the onpload run mode with the -f option	16-4
Typing the onpload -f flags	16-5
Interpreting the onpload -d and -f options together	16-6
Modifying the size of onpload database parameters	16-6
Using the onpload -i option	16-9
Overriding the onpload database values	16-9
Loading data into collection data type columns	16-12

Chapter 17. The onpladm utility 17-1

Overview of the onpladm utility.	17-2
The onpladm utility features	17-2
Specification-file conventions	17-3
Error handling.	17-4
Defining onpladm utility jobs.	17-4
Creating onpladm jobs	17-4
Modifying a job by using a detailed specification file	17-13
Describing a job	17-14
Listing all jobs in a project	17-14
Running a job	17-15
Deleting a job	17-16
Defining device arrays.	17-17
Creating a device array	17-17
Modifying a device array.	17-18
Describing a device array.	17-18
Listing project device arrays.	17-18
Deleting a device array	17-19
Defining maps	17-19
Creating maps	17-19
Deleting a map	17-24
Describing a map	17-24
Modifying a map using a detailed specification file	17-25
Listing all maps in a project	17-25
Defining formats	17-26
Creating a format	17-26
Modifying a format using a specification file	17-28
Describing a format	17-28

Listing all formats in a project	17-29
Deleting a format	17-29
Defining queries	17-30
Creating a query	17-30
Modifying a query	17-30
Describing a query	17-30
Listing all queries in a project	17-31
Deleting a query	17-31
Defining filters	17-32
Creating a filter	17-32
Modifying a filter	17-33
Describing a filter	17-33
Listing all filters in a project	17-33
Deleting a filter	17-34
Defining projects	17-34
Creating a new project	17-34
Running all jobs in a project	17-34
Listing all projects	17-35
Deleting a project	17-35
Defining machine types	17-36
Creating a machine type	17-36
Modifying a machine type	17-36
Describing a machine	17-37
Listing all existing machine types	17-37
Deleting a machine type	17-37
Defining database operations	17-38
Creating a database project	17-38
Configuring target-server attributes	17-40

Appendix A. onpload database A-1

Defaults table in the onpload database	A-1
Delimiters table in the onpload database	A-1
Device table in the onpload database	A-2
Driver table in the onpload database	A-2
Filteritem table in the onpload database	A-3
Filters table in the onpload database	A-3
Formatitem table in the onpload database	A-3
Formats table in the onpload database	A-6
Language table in the onpload database	A-6
Machines table in the onpload database	A-6
Mapitem table in the onpload database	A-7
Mapoption table in the onpload database	A-7
Maps table in the onpload database	A-8
Note table in the onpload database	A-9
Project table in the onpload database	A-9
Query table in the onpload database	A-10
Session table in the onpload database	A-10

Appendix B. High-Performance Loader configuration file. B-1

HPL configuration parameter descriptions	B-1
HPL configuration parameter file conventions	B-1
AIOBUFFERS configuration parameter	B-1
AIOBUFSIZE configuration parameter	B-2
CONVERTTHREADS configuration parameter	B-2
CONVERTVPS configuration parameter	B-3
HPLAPIVERSION configuration parameter	B-3
HPL_DYNAMIC_LIB_PATH configuration parameter	B-4
STRMBUFFERS configuration parameter	B-4
STRMBUFFERSIZE configuration parameter	B-4

Appendix C. Picture strings	C-1
Alphanumeric pictures	C-1
Numeric pictures	C-2
Date pictures	C-2
Appendix D. Match condition operators and characters	D-1
Operator descriptions and examples.	D-1
Appendix E. Custom-conversion functions	E-1
Custom conversion example	E-1
The onpload conversion process	E-1
API functions	E-3
Appendix F. onstat -j option	F-1
Using the onstat -j option	F-1
Appendix G. HPL log-file and pop-up messages	G-1
How HPL log-file messages are ordered	G-1
HPL log-file message categories	G-2
HPL log-file messages	G-2
HPL log-file pop-up messages	G-16
Appendix H. Custom drivers	H-1
Adding a custom driver to the onpload utility	H-1
Adding the driver name to the onpload database	H-1
Preparing the custom-driver code	H-1
Rebuilding the shared-library file	H-3
Connecting your code to onpload at runtime.	H-4
Driver initialization	H-5
Registering driver functions	H-5
An example of a custom driver	H-6
The plcstdrv.c file	H-6
Custom-driver code for MYDRIVER	H-6
Available driver methods	H-9
The PL_MTH_OPEN function	H-10
The PL_MTH_CLOSE function	H-10
Available API support functions	H-10
The pl_inherit_methods(driver, methodtable) function	H-10
The pl_set_method_function(methodtable, method, function) function	H-11
The pl_driver_inherit(method) function	H-11
The pl_get_recordlength() function	H-11
The pl_set_informix_conversion(flag) function	H-12
The pl_lock_globals() function	H-12
The pl_reset_inherit_chain(method) function	H-12
Appendix I. Running load and unload jobs on a Windows computer	I-1
Using the onpladm utility on Windows	I-1
Running the run job or run project commands	I-1
Running onpladm on UNIX with the database server running on Windows.	I-1
Preparing jobs with the ipload utility on Windows computers	I-3
Appendix J. Conversion and reversion scripts for HPL database migration	J-1
Upgrading the High-Performance Loader onpload database	J-1
Reverting from the current onpload database	J-1
Appendix K. Accessibility	K-1
Accessibility features for IBM Informix	K-1
Accessibility Features.	K-1
Keyboard Navigation.	K-1

Related Accessibility Information K-1
IBM and Accessibility K-1
Dotted Decimal Syntax Diagrams K-1

Notices L-1
Trademarks L-3

Index X-1

Introduction

In this introduction	xi
About this publication	xi
What's New in the High Performance Loader for Informix, Version 11.50	xi
Types of users	xii
Software dependencies	xii
Assumptions about your locale.	xii
Demonstration databases	xiii
Example Code Conventions.	xiii
Additional Documentation	xiv
Compliance with Industry Standards	xiv
Syntax Diagrams	xiv
How to Read a Command-Line Syntax Diagram	xv
Keywords and Punctuation	xvi
Identifiers and Names	xvii
How to Provide Documentation Feedback	xvii

In this introduction

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

About this publication

This publication describes how to use the IBM® Informix® 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 **ipload** utility, a graphical user interface that 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**, **onpload**, and **onpladm** utilities. See Appendix A, "onpload database," on page A-1 for information on the tables of the **onpload** database.

What's New in the High Performance Loader for Informix, Version 11.50

This publication includes information about new features and changes in existing functionality.

The following changes and enhancements are relevant to this publication. For a complete list of what's new in this release, see the release notes or the information center.

Table 1. What's New in IBM Informix High-Performance Loader User's Guide

Overview	Reference
<p>New editions and product names:</p> <p>Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products:</p> <ul style="list-style-type: none"> • IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS) • IBM OpenAdmin Tool for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS) • IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature 	<p>For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.</p>

Types of users

This publication is written for the following users:

- Database administrators
- Database server administrators

This publication 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

Software dependencies

This publication assumes that you are using Informix Version 11.50. If you use X-Windows to connect to a remote operating system, X-Windows must be installed on one of the operating systems listed on the following page: <http://www.ibm.com/software/data/informix/ids/requirements.html>.

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 publication 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 publications 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.

Example Code Conventions

Examples of SQL code occur throughout this publication. Except as 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 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. If you are using DB-Access, you must delimit multiple statements with semicolons.

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 documentation for your product.

Additional Documentation

Documentation about this release of IBM Informix products is available in various formats.

All of the product documentation (including release notes, machine notes, and documentation notes) is available from the information center on the Web at <http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp>. Alternatively, you can access or install the product documentation from the Quick Start CD that is shipped with the product.

Compliance with Industry Standards

IBM Informix products are compliant with various standards.

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 IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

The IBM Informix Geodetic DataBlade Module supports a subset of the data types from the Spatial Data Transfer Standard (SDTS)—Federal Information Processing Standard 173, as referenced by the document Content Standard for Geospatial Metadata, Federal Geographic Data Committee, June 8, 1994 (FGDC Metadata Standard).

IBM Informix Dynamic Server (IDS) Enterprise Edition, Version 11.50 is certified under the Common Criteria. For more information, see *Common Criteria Certification: Requirements for IBM Informix Dynamic Server*, which is available at <http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7690-00>.

Syntax Diagrams

Syntax diagrams use special components to describe the syntax for statements and commands.

Table 2. Syntax Diagram Components







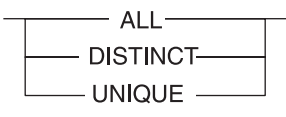
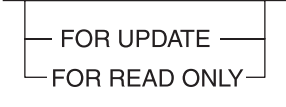
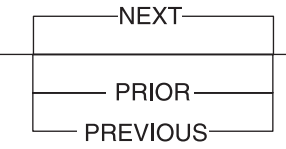
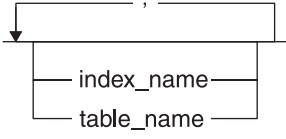

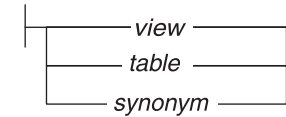
Component represented in PDF	Component represented in HTML	Meaning
	>>-----	Statement begins.
	----->	Statement continues on next line.
	>-----	Statement continues from previous line.
	-----<<	Statement ends.
	-----SELECT-----	Required item.

Table 2. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning
	<code>--+-----+-- '-----LOCAL-----'</code>	Optional item.
	<code>---+-----ALL-----+-- +--DISTINCT-----+ '---UNIQUE-----'</code>	Required item with choice. One and only one item must be present.
	<code>---+-----+-- +--FOR UPDATE-----+ '--FOR READ ONLY--'</code>	Optional items with choice are shown below the main line, one of which you might specify.
	<code>.---NEXT-----. ---+-----+-- +---PRIOR-----+ '---PREVIOUS-----'</code>	The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line will be used as the default.
	<code>.-----,-----. v ---+-----+-- +---index_name---+ '---table_name---'</code>	Optional items. Several items are allowed; a comma must precede each repetition.
	<code>>>- Table Reference -><</code>	Reference to a syntax segment.
Table Reference 	Table Reference <code> ---+-----view-----+-- +-----table-----+ '---synonym-----'</code>	Syntax segment.

How to Read a Command-Line Syntax Diagram

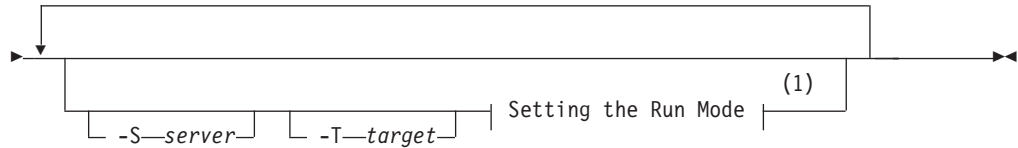
Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a No-Conversion Job

`>>- onpladm create job—job— [-p—project—] -n— -d—device— -D—database—>>`

`>>- -t—table—>>`

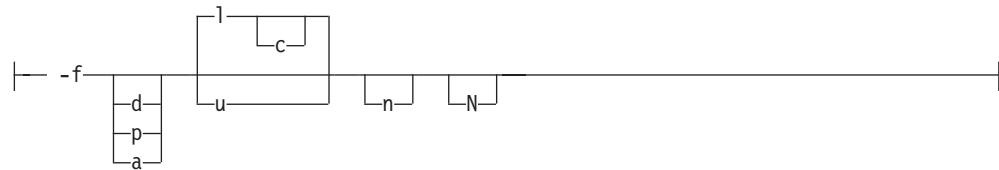


Notes:

1 See page Z-1

This diagram has a segment named "Setting the Run Mode," which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment in on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the Run Mode:



To see how to construct a command correctly, start at the top left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
 - **-n**
 - **-d** and the name of the device
 - **-D** and the name of the database
 - **-t** and the name of the table
4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
 - **-S** and the server name
 - **-T** and the target server name
 - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and Punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and Names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.

▶▶—SELECT—*column_name*—FROM—*table_name*—▶▶

When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

How to Provide Documentation Feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send e-mail to docinf@us.ibm.com.
- Go to the information center at <http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp> and open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.
- Add comments to topics directly in the Informix information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more! Find out more at <http://publib.boulder.ibm.com/infocenter/idshelp/v115/topic/com.ibm.start.doc/contributing.htm>.

Feedback from all methods is monitored by those who maintain the user documentation. The feedback methods are reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact IBM Technical Support. For instructions, see the IBM Informix Technical Support website at <http://www.ibm.com/planetwide/>.

We appreciate your suggestions.

Chapter 1. High-Performance Loader overview

Overview of HPL features	1-1
The HPL data-load process	1-2
The HPL data-unload process	1-3
HPL loading modes	1-4
The HPL express mode	1-4
The HPL deluxe mode	1-4
HPL components	1-4
onpload utility	1-4
ipload utility.	1-5
onpladm utility.	1-5
onpload database	1-5
Relationships among the parts of the HPL	1-5
Environment variables needed for the HPL	1-6
DBONPLOAD environment variable.	1-7
PLCONFIG environment variable.	1-7
PLOAD_SHMBASE environment variable	1-8
Avoiding shared-memory collision	1-8
Setting the PLOAD_SHMBASE environment variable	1-8
PLOAD_LO_PATH environment variable	1-8
PLOAD_SHMAT environment variable	1-8
Architecture of the onpload utility	1-8
The onpload utility deluxe-mode process	1-9
Threads that the onpload utility uses	1-9
Threads that the database server uses	1-10
The onpload utility express-mode load process	1-10
The onpload utility unload process	1-12

Overview of HPL features

The HPL is a database server tool 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 components:

- The **ipload** utility, which is a graphical user interface (GUI). This UNIX application helps you prepare load and unload jobs for both UNIX and Windows and includes the following functionality:
 - 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 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, which is a command-line version of the **ipload** utility.
- The **onpload** utility, which allows you to control HPL from a script.
- The **onpload** database, which maintains information about load and unload jobs that you have prepared.

You use the **ipload** utility to manage the **onpload** database on any database server on your network.

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 COBOL, ASCII, multibyte, delimited, or binary data.
- Can load and unload data of a different GLS locale from that of the database server.
- Provides synonym support for tables that are valid for the local database server. You can use synonyms for both the load and unload operations.
- Provides support for unloading data with a query that accesses a view in its SELECT statement.
- Supports loading of raw tables in express mode.

The HPL data-load process

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

UNIX Only

- Pipes (application-generated data)

End of UNIX Only

The HPL supports load and unload data greater than 2 gigabytes 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

Global Language Support

- Converting from the code set of one locale to the code set of another locale

End of Global Language Support

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,” on page 12-1, addresses the process of loading a file to a database.

The figure below summarizes the data-load process by showing how data moves from data files to table entries.

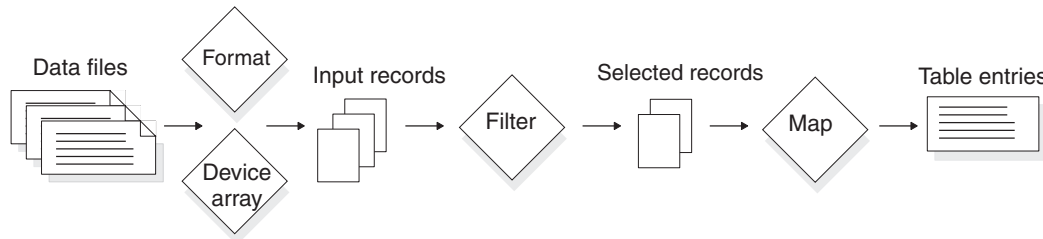


Figure 1-1. The data-load process

The HPL data-unload process

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

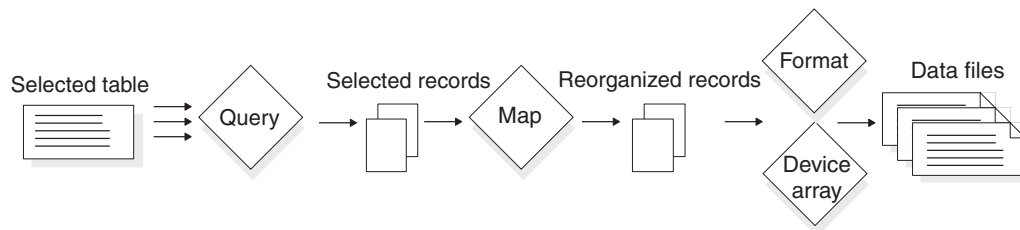


Figure 1-2. The data-unload process

The HPL uses the same components for an unload job as for a load job, with one exception. For an unload job, **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 job, see Chapter 11, “Unloading data from a database,” on page 11-1.

HPL 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,” on page 15-1.

The HPL express mode

HPL 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 re-enables indexes, evaluates and re-enables constraints, if possible, and re-enables triggers. (HPL does not evaluate triggers with respect to the loaded data.)

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

The HPL deluxe mode

HPL 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, the **ipload** utility, the **onpladm** command-line utility, 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

The **onpload** utility can load or unload data from files that are larger than 2 gigabytes and can generate **.log**, **.rej** and **.flt** files that are larger than 2 gigabytes.

The **onpload** database must be accessible from the machine on which the **onpload** utility is run. It does not need to be on the same machine.

The database that the **onpload** loads or unloads must be in a database server instance on the machine that is running the **onpload** (or **ipload** or **onpladm**) utility.

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

When you start **onpload** from **ipload**, the **onpload** and **ipload** utilities communicate using a socket connection. The **onpload** utility runs completely independently from **ipload** (see the information to the right of the vertical line in Figure 1-3).

For more information about **onpload**, see “Architecture of the onpload utility” on page 1-8, and Chapter 16, “The onpload utility,” on page 16-1.

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 2-2, “Starting the ipload utility” on page 3-1, and “Configuring ipload” on page 5-1.

You can use **ipload** to manage **onpload** databases on both UNIX and Windows. The **ipload** utility is not available on the Mac OS X

For more information about managing **onpload** databases on Windows, see Appendix I, “Running load and unload jobs on a Windows computer,” on page I-1.

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,” on page 17-1.

onpload database

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

- Contains information that the **onpload** utility requires to perform data loads and unloads
- Must be accessible from the machine where the **onpload** utility is run. It does not need to be on the same machine.

However; the database that the utility loads or unloads must be in a database server instance on the machine where the **onpload** utility (and hence the **ipload** or **onpladm** utility) is being run.

Relationships among the parts of the HPL

Figure 1-3 shows the relationships among the parts of the HPL. The **ipload** utility or **onpladm** command-line utility connects to the database server to populate the **onpload** database. The **ipload** utility or the **onpladm** utility controls the **onpload**

utility, which uses multithreaded architecture to make multiple connections to the database server and multiple I/O connections to tapes, files, or pipes.

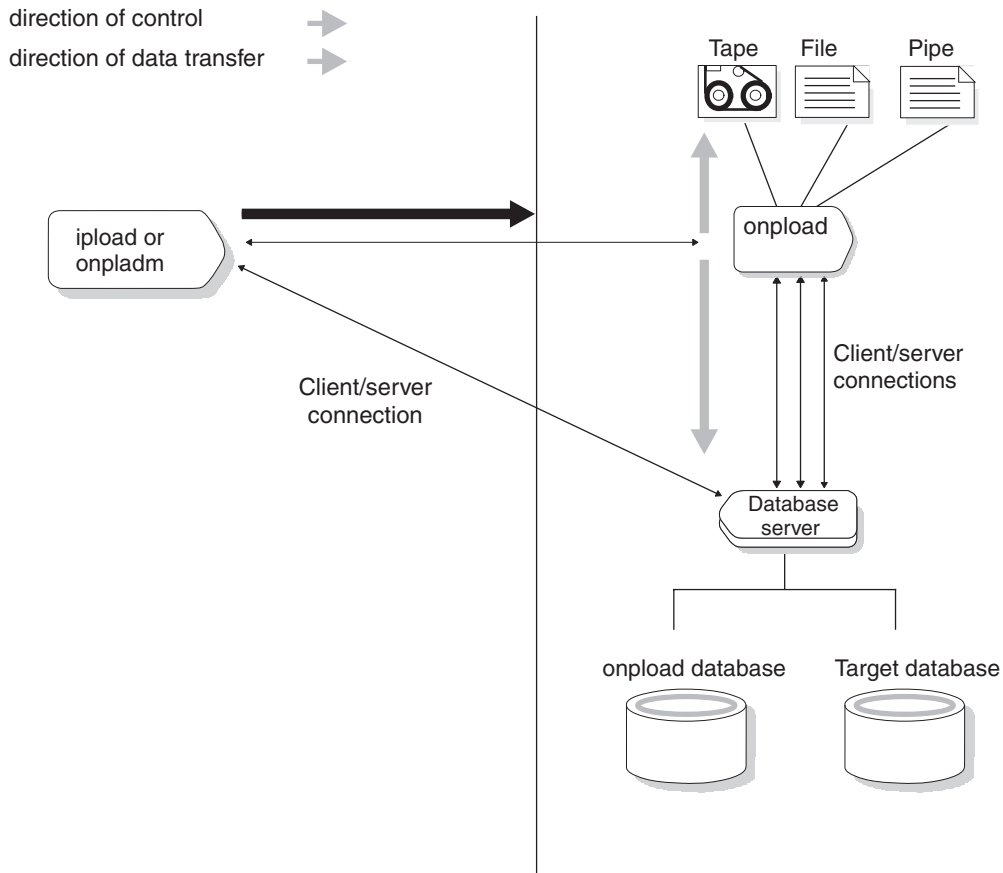


Figure 1-3. Relationships among the parts of the HPL

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

Environment variables needed for the HPL

The HPL is part of the database server, so you must start the database server before you use the HPL.

Important: Before you start the database server and use HPL, you must set these environment variables:

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

The **INFORMIXDIR**, **ONCONFIG**, and **INFORMIXSERVER** environment variables are documented in the *IBM Informix Guide to SQL: Reference*.

Some computers use the **LD_LIBRARY_PATH** environment variable for shared libraries. The name of this environment variable is platform-dependent. See your

operating system documentation for the name of the environment variable that specifies the search path for shared libraries. Then see the machine notes for information about **LD_LIBRARY_PATH**.

You can use the **IFX_ONPLOAD_AUTO_UPGRADE** environment variable with the **ipload** or **onpladm** utilities to automatically upgrade the **onpload** database the first time you invoke an HPL utility using the **ipload** or **onpladm** command after you migrate to a new database server version. You cannot use the **IFX_ONPLOAD_AUTO_UPGRADE** environment variable with the **onpload** utility.

In addition to the environment variables listed above, the following environment variables pertain to the HPL:

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

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,” on page B-1.

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 or 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. The **onpload** utility 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 *IBM Informix Administrator's Reference*.

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 smart large object (CLOB or BLOB) output files. If this environment variable is not set, the database server puts the output files in **/tmp**.

PLOAD_SHMAT environment variable

If the pload converter thread cannot attach to the passed address, you receive a message that asks you to set the **PLOAD_SHMAT** environment variable. When you set the **PLOAD_SHMAT** environment variable, the pload converter calculates the address using a global attached segment list that is maintained across pload virtual processors. The pload converter attaches at the next available address after the highest address on the list, ensuring that the converter always attaches to an unused shared memory segment.

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

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 *IBM Informix Administrator's Guide*. The following sections describe how **onpload** uses multithreading for deluxe loads, express loads, and unloads.

The onpload utility deluxe-mode process

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 it were loaded using SQL INSERT statements.

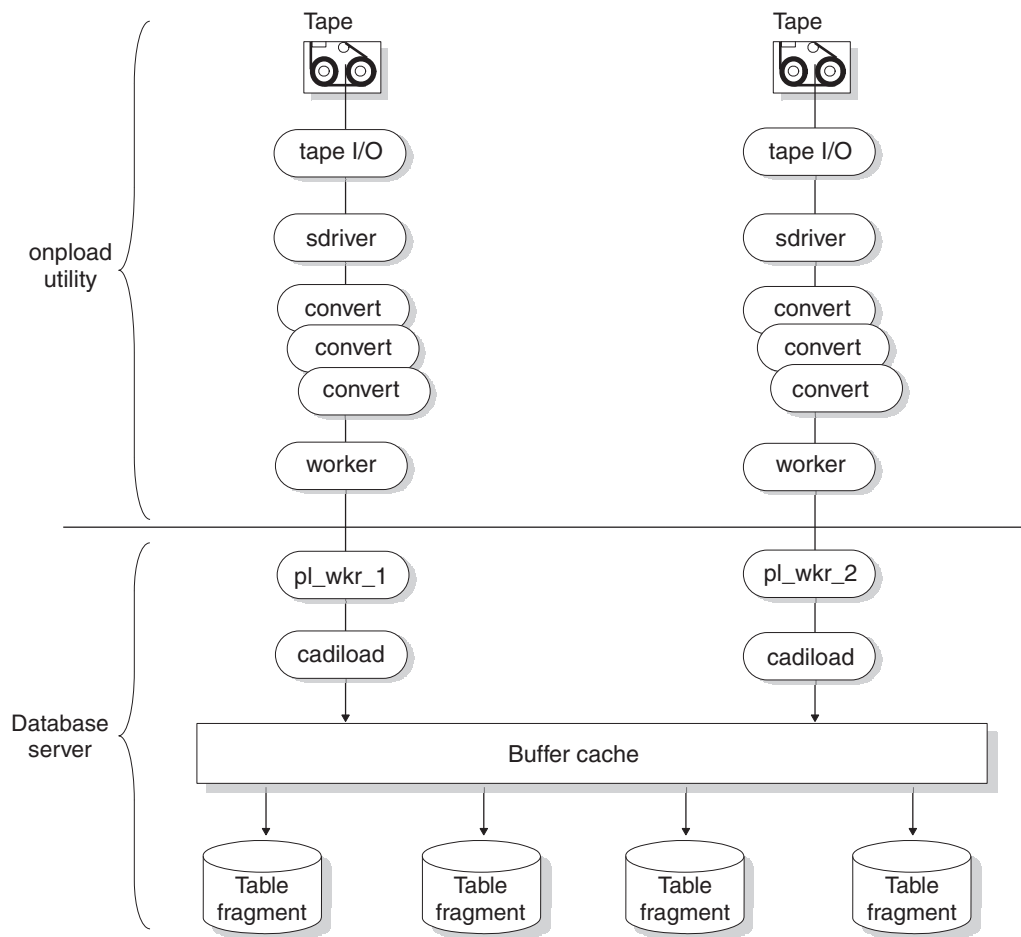


Figure 1-4. A deluxe-mode load

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.

UNIX Only

The **onpload** utility starts a similar thread for piped output.

End of UNIX Only

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 during an INSERT statement. The SQL optimizer governs the method that is used for inserting the data.

The onpload utility express-mode load process

Figure 1-5 on page 1-11 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 1-9. 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.

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.

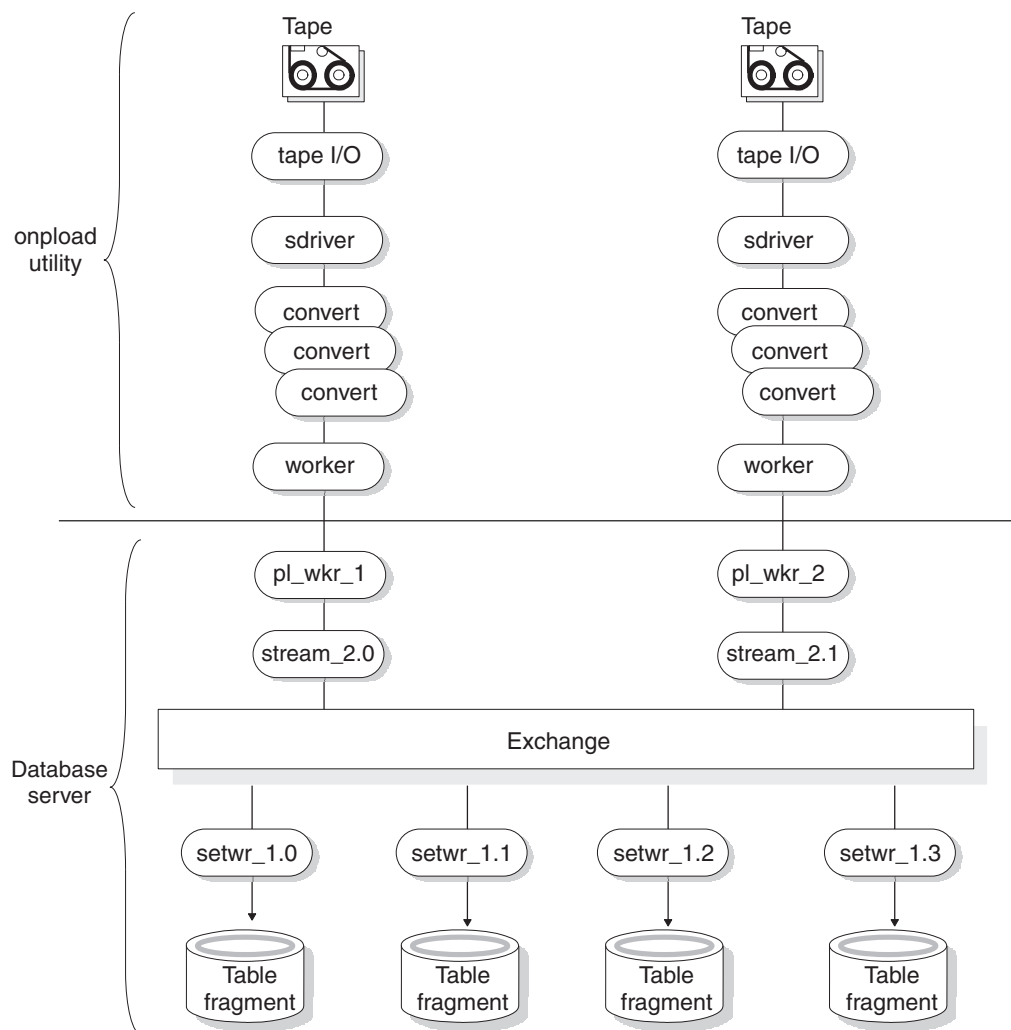


Figure 1-5. An express-mode load

The onpload utility unload process

Figure 1-6 on page 1-12 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 1-9 and “Threads that the database server uses” on page 1-10, except that the threads are unloading the data instead of loading it.

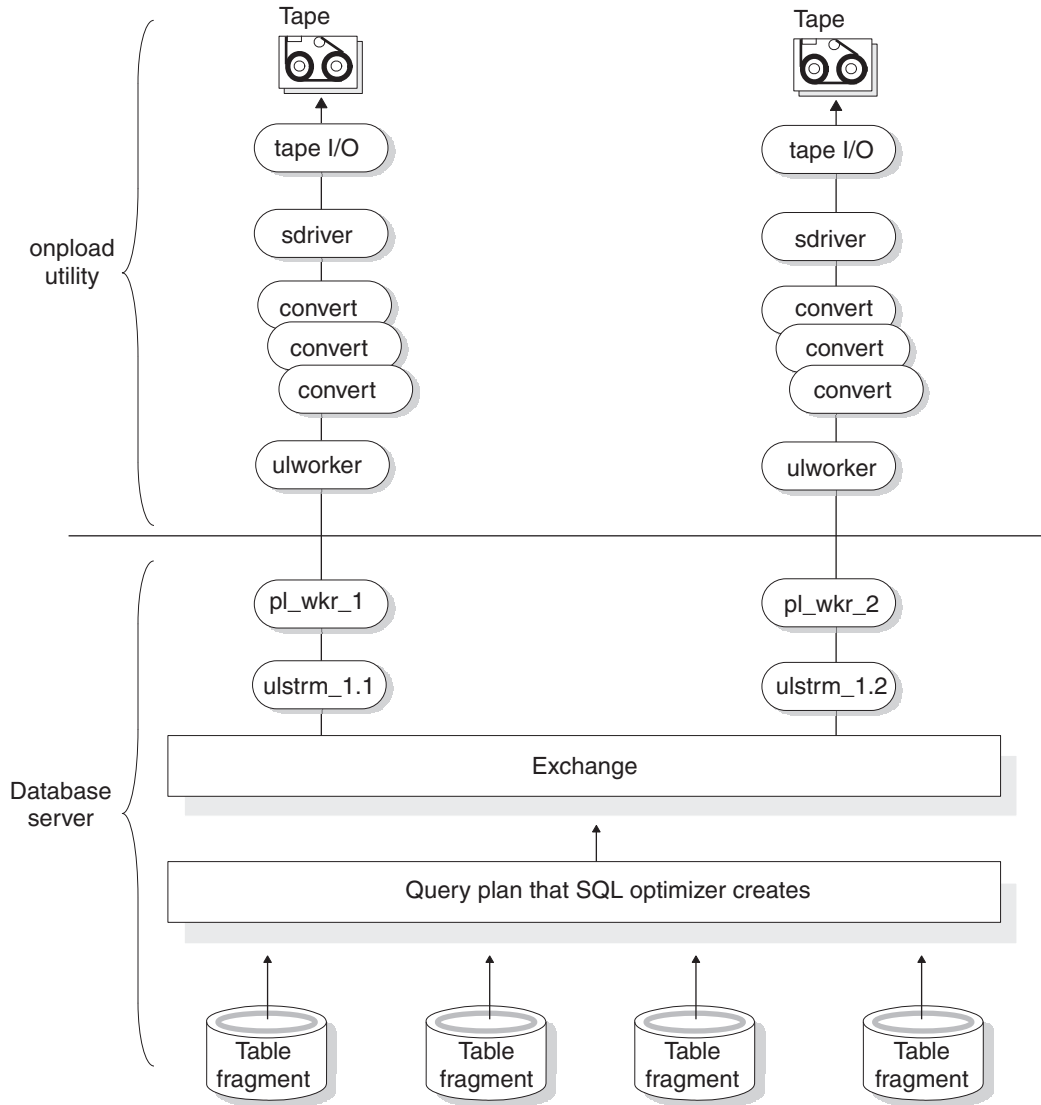


Figure 1-6. The unload procedure

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 like a query plan running 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 1-12.

Chapter 2. Getting started through examples of using the ipload utility

Examples of loading and unloading jobs using the ipload utility	2-1
Preparing to use the ipload utility	2-2
Creating a file of data.	2-2
Creating a database	2-2
The ipload utility	2-2
Starting ipload	2-2
Choosing a project.	2-3
Checking the ipload utility default values	2-3
Looking at the Defaults window	2-3
Looking at the Machines window.	2-4
Load Job windows.	2-4
Load Job Select window	2-4
Load Job window	2-6
Device-Array windows	2-7
Device Array Selection window	2-7
Device-Array Definition window	2-7
Format windows	2-8
Format Views window	2-8
Record Formats window.	2-9
Format-Definition window.	2-11
Filter, Discard Records, and Logfile text boxes	2-12
Filter text box	2-12
Discard Records text box	2-12
Logfile text box	2-13
Map Views window	2-13
Creating a map by using Map Views window	2-13
Load Record Maps window	2-14
Map-Definition window	2-15
Load Options window	2-18
Running the ipload job	2-19
Active Job window	2-19
Verifying the transfer of data	2-20
Performing a level-0 backup	2-20
ipload utility Generate options	2-21
Using the information you created with the ipload example	2-21
Preparing the Unload-Job window	2-21
Performing the unload job.	2-25

Examples of loading and unloading jobs using the ipload utility

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 (for a load and an unload job) that use the **ipload** graphical user interface (GUI).

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.

The **onpladm** command-line interface is equivalent to the **ipload** utility. For information on using **onpladm** instead of using **ipload**, see the Chapter 17, “The onpladm utility,” on page 17-1 chapter in this guide.

Preparing to use the ipload utility

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 a Windows computer,” on page I-1.

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,” on page A-1, describes the database tables.

When **ipload** starts, the HPL main window appears, as Figure 2-1 shows.

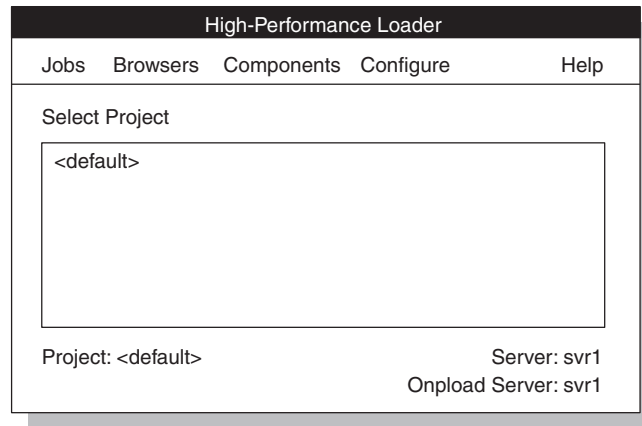


Figure 2-1. The HPL Main Window

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

The **onpladm** command line utility provides the same functionality as the **ipload** graphical user interface. For information on using **onpladm** instead of using **ipload**, see the Chapter 17, “The onpladm utility,” on page 17-1 chapter in this guide.

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 HPL projects,” on page 4-1.

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 ipload utility default values

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,” on page 5-1, 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 default values” on page 5-2.

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 5-6. 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 2-6, 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.

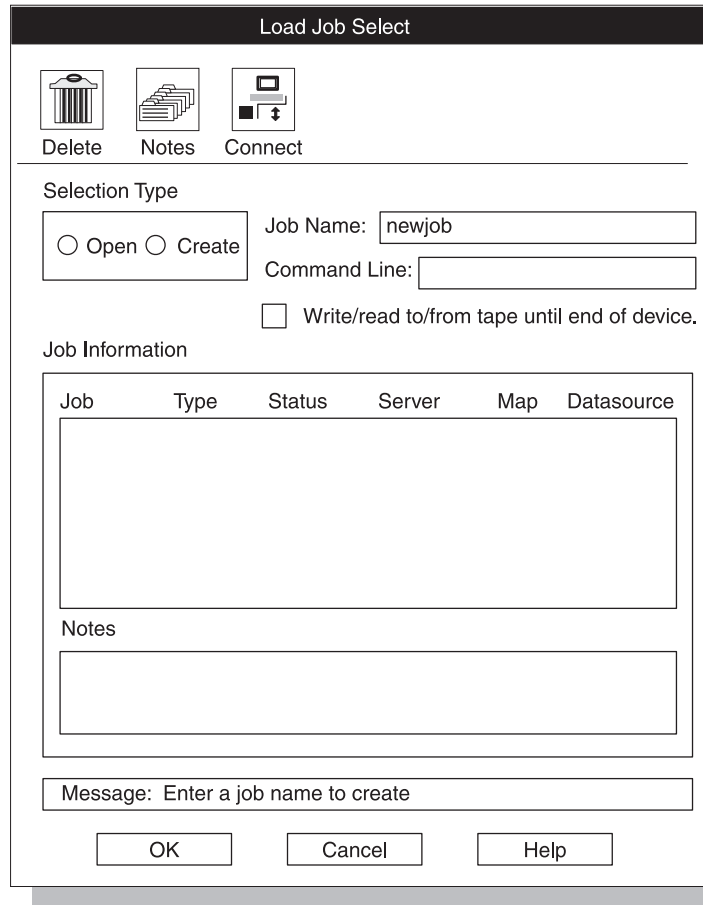


Figure 2-2. The Load Job Select Window

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.

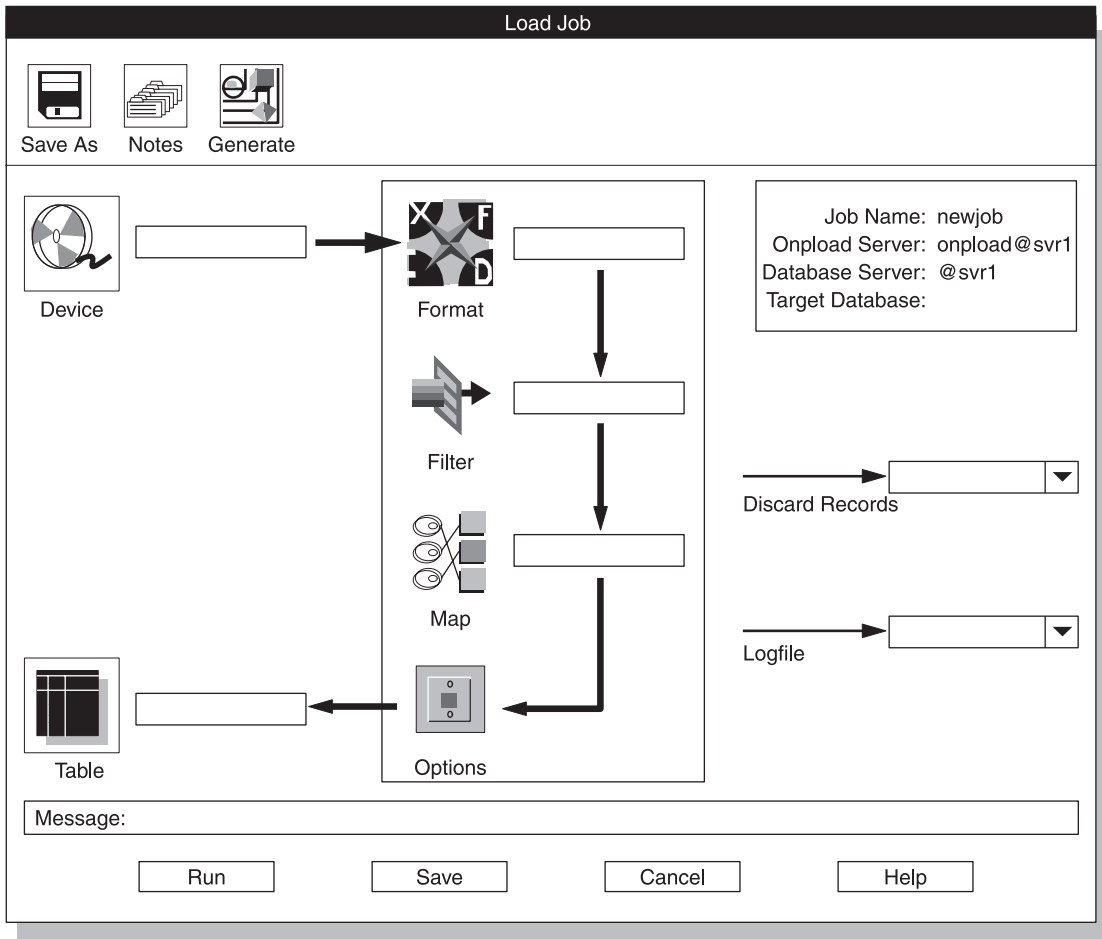


Figure 2-3. The Load Job Window

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 2-2.

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.

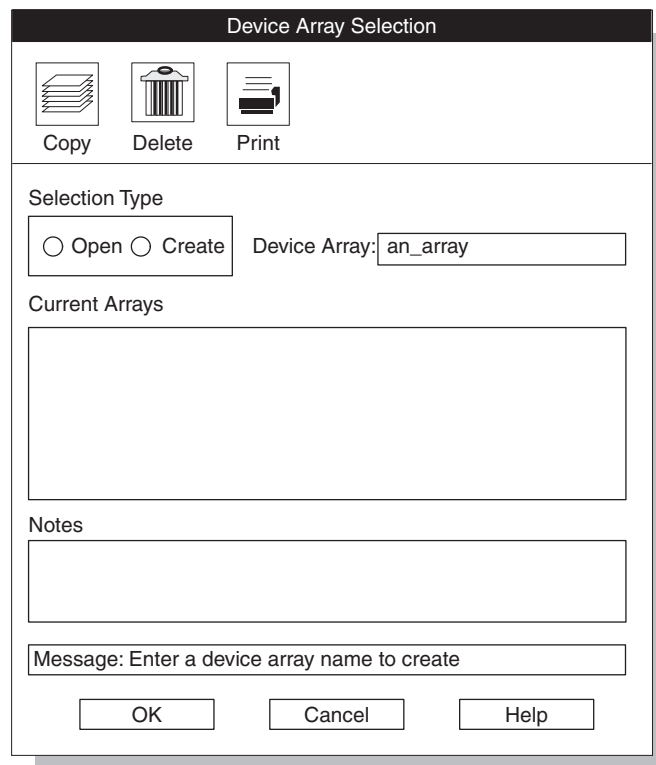


Figure 2-4. The Device Array Selection window

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.

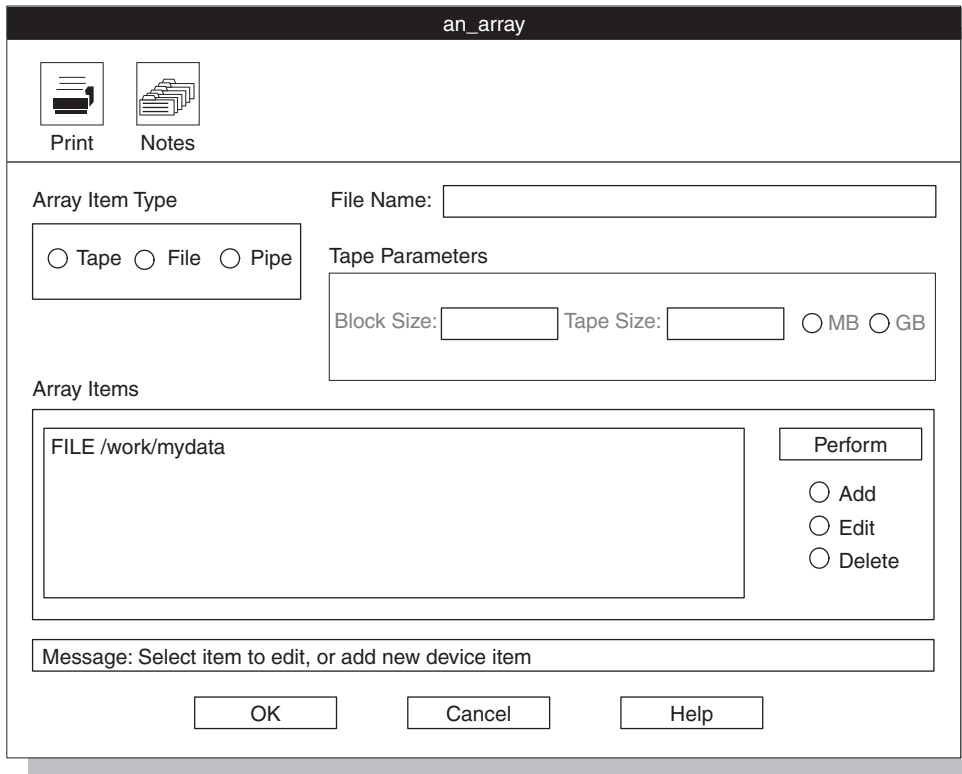


Figure 2-5. The Device-Array Definition window with one array item

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 path name 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 2-2. 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.

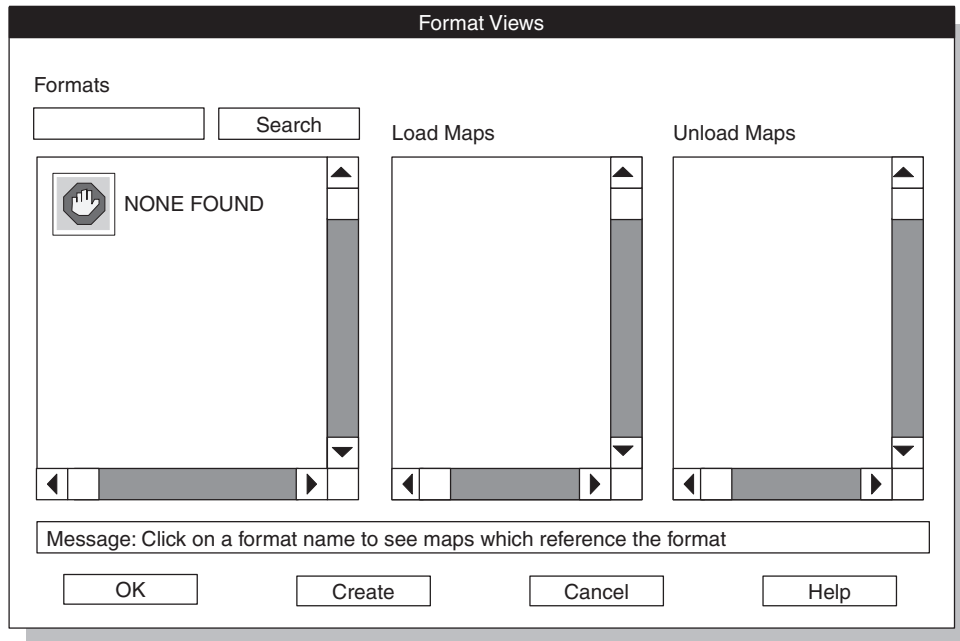


Figure 2-6. The Format Views window

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.

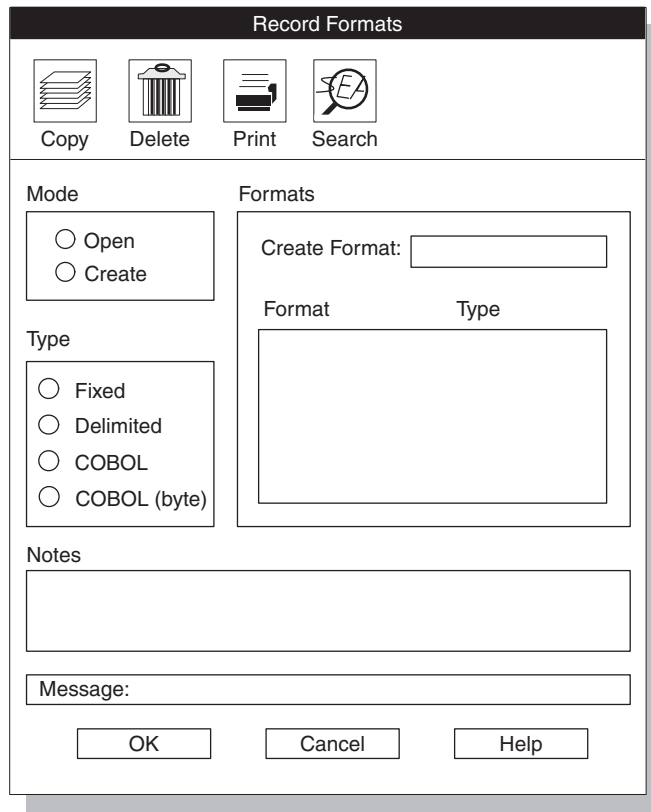


Figure 2-7. The Record Formats window

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,” on page 7-1.

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.

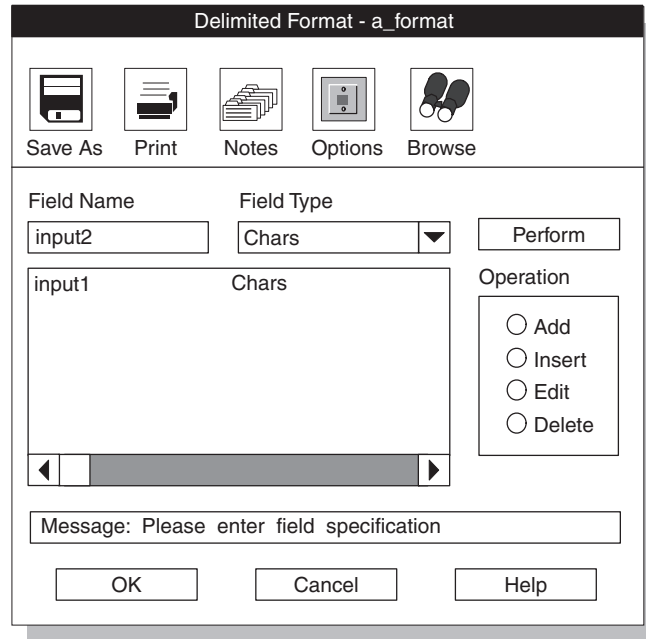


Figure 2-8. The Format-Definition window

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," on page 7-1.

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.

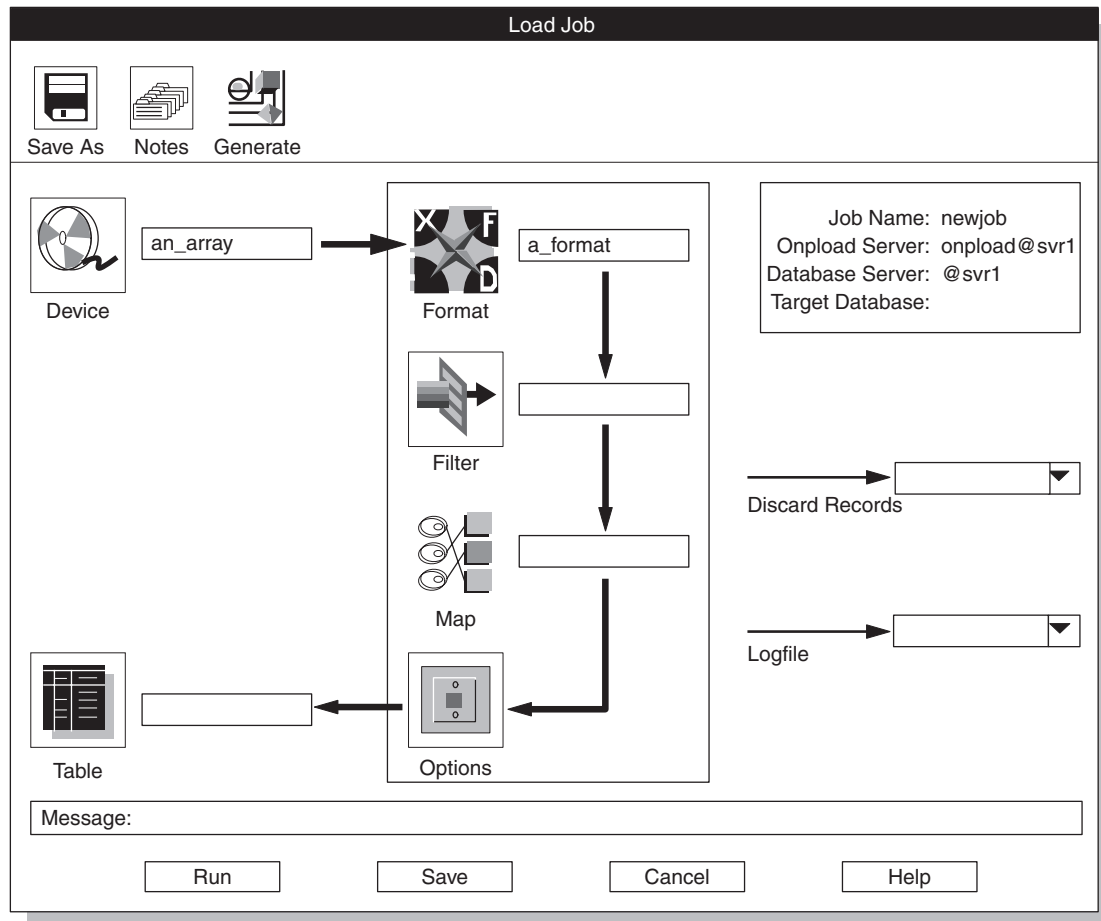


Figure 2-9. Partially Completed Load Job window

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," on page 10-1, 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 14-3 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 14-5 describes how to view the log file.

Map Views window

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

Creating a map by using 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.

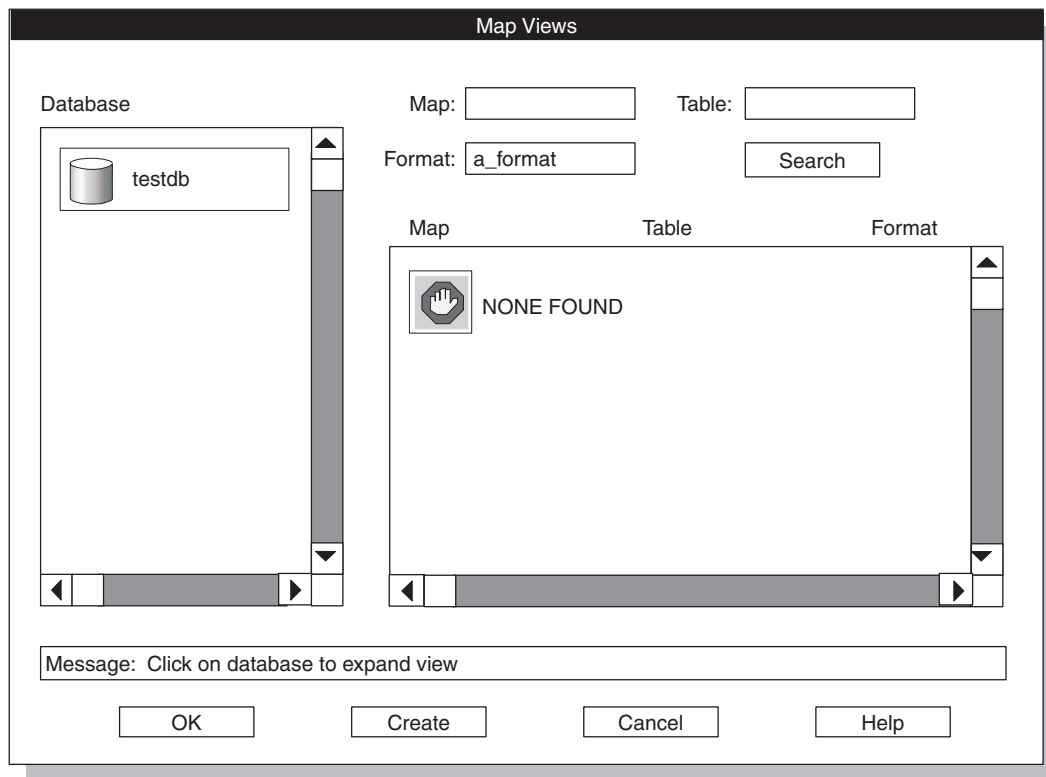


Figure 2-10. The Map Views window

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.

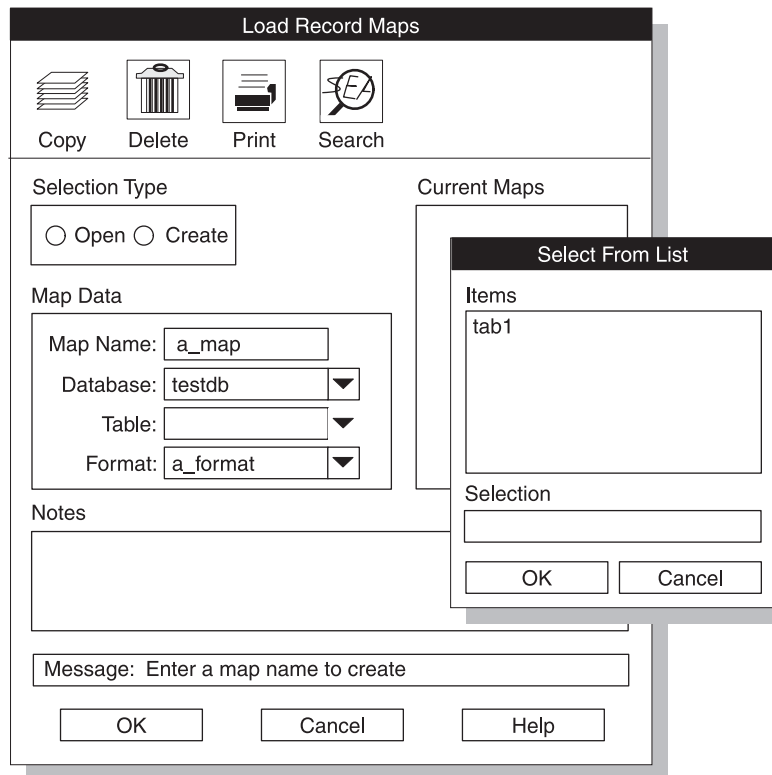


Figure 2-11. A partially completed Load Record Maps window with an open selection list

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 2-10.

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.

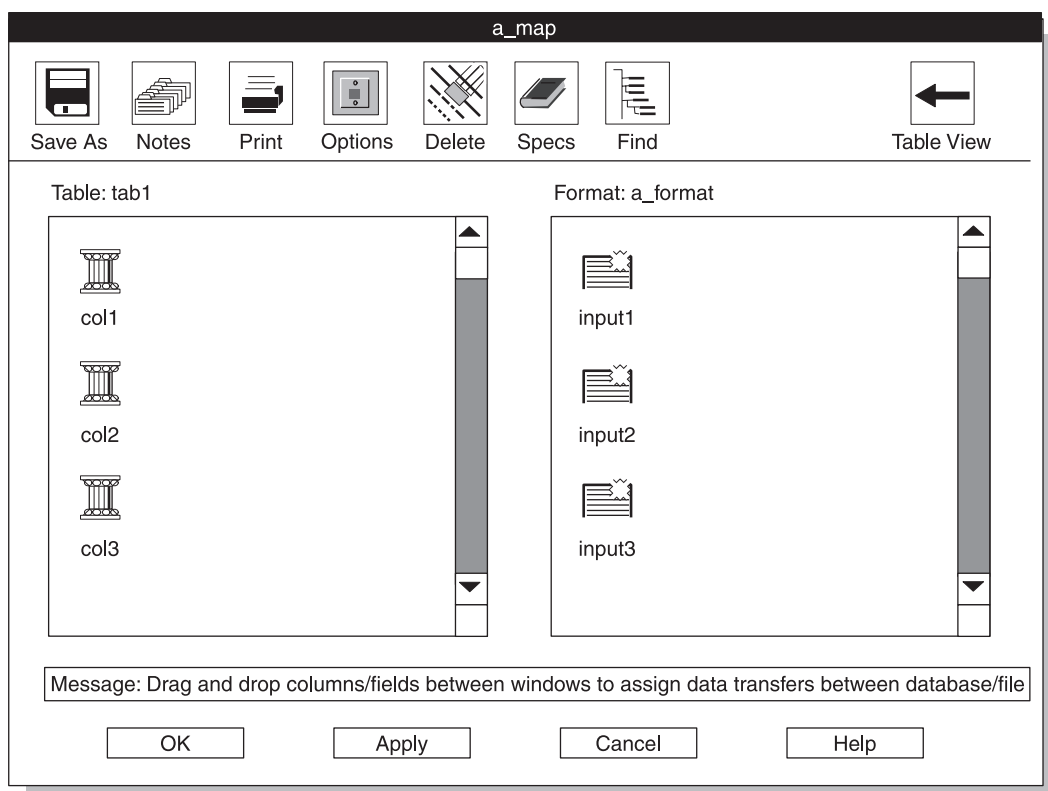


Figure 2-12. The Map-Definition window

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.

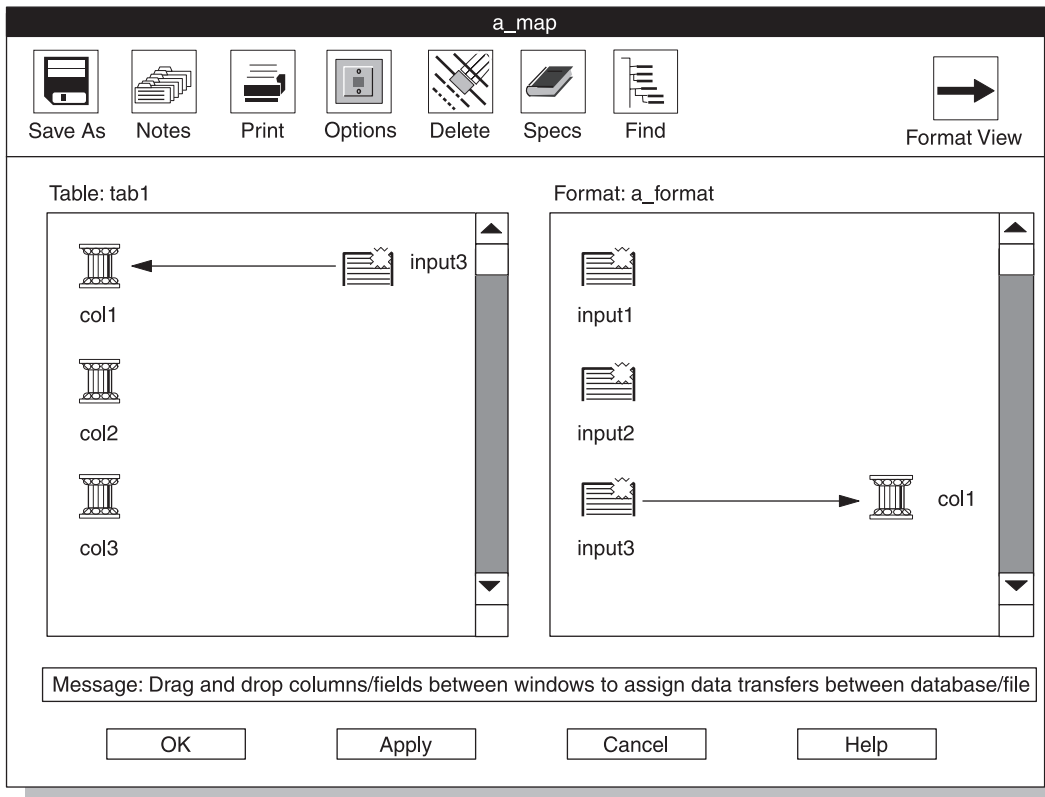


Figure 2-13. The Map-Definition window with one association completed

4. Connect **col2** to **input2**.
5. Connect **col3** to **input1**.

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

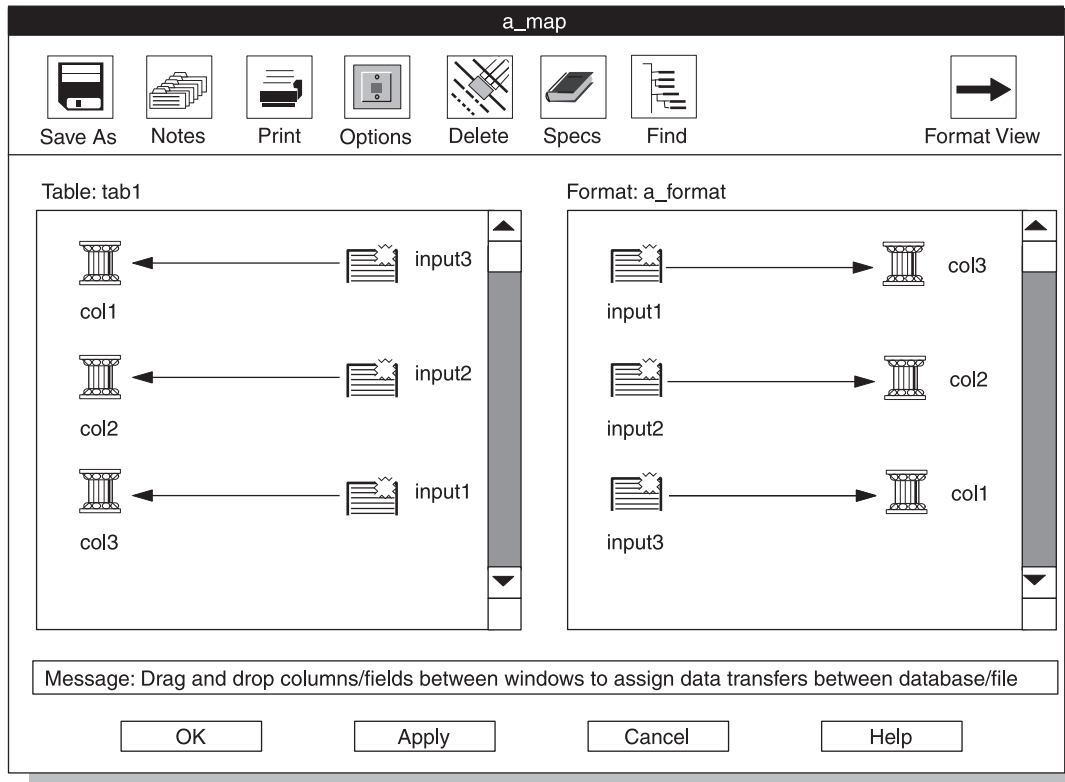


Figure 2-14. The Map-Definition window with all associations completed

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.

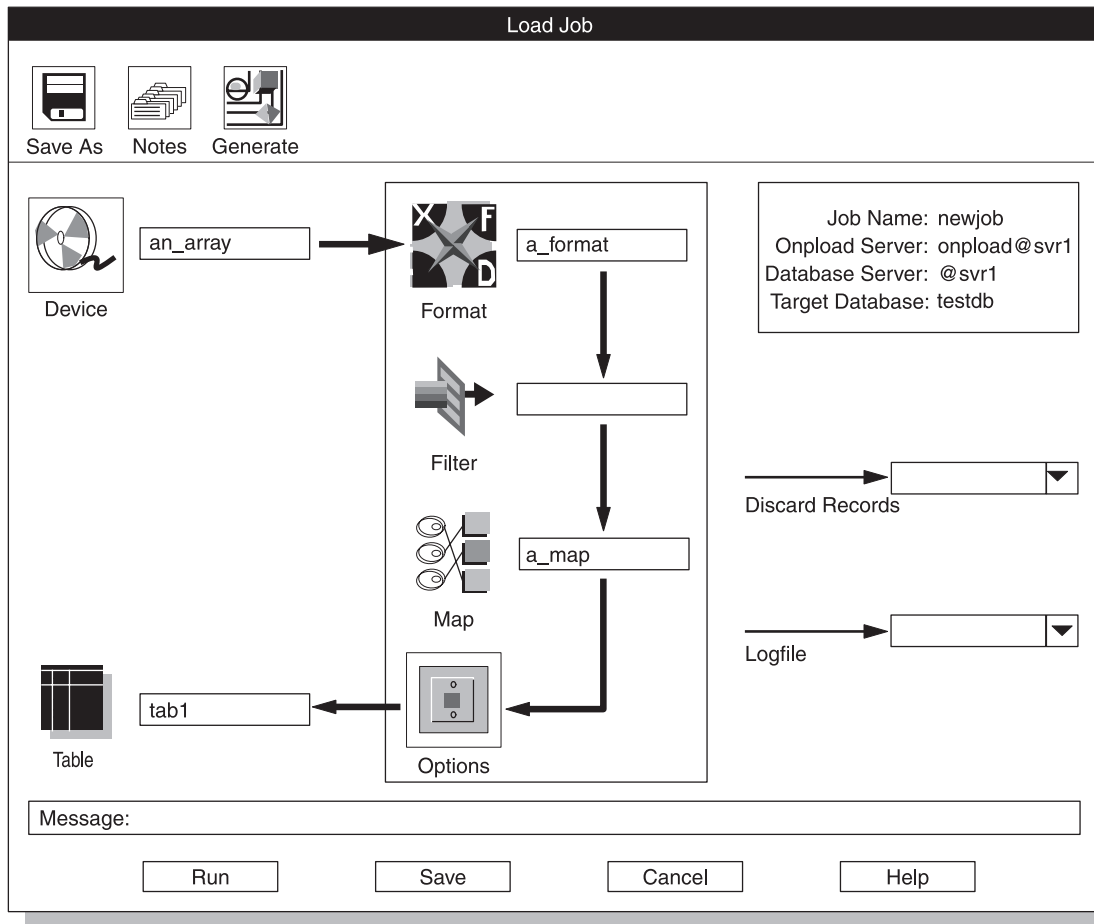


Figure 2-15. The Load Job window with all required component boxes completed

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,” on page 15-1. 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.

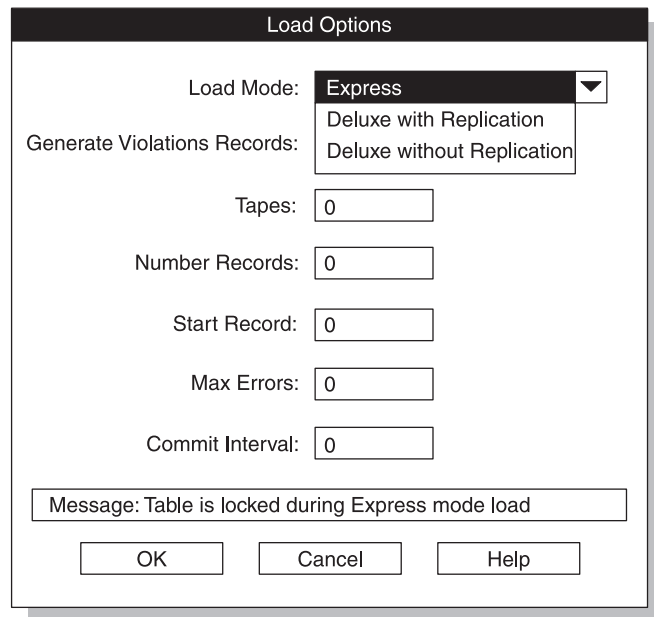


Figure 2-16. The Load Options window

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.

Running the ipload job

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 run the job, or you can return at a later time and run the job.

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

The Active Job window appears, as Figure 2-17 on page 2-20 shows.

Tip: The **ipload** utility generates an **onpload** command and then runs 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 12-7.

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.

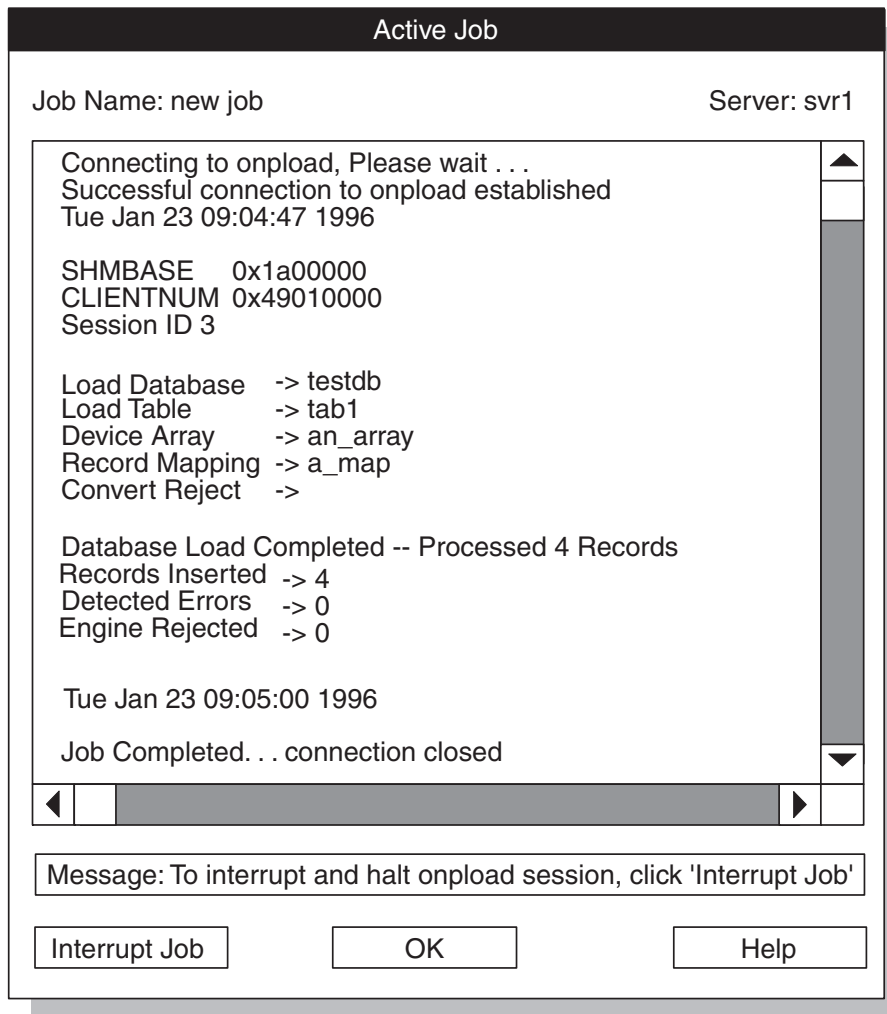


Figure 2-17. The Active Job window

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

Verifying the transfer of data

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.

ipload utility **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, “The **Generate** options of the **ipload** utility,” on page 13-1.

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 run the unload job.

Using the information you created with the ipload 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 (2-2)
- Start **ipload** (2-2)
- Check the defaults (2-3)

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 2-6) 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**.

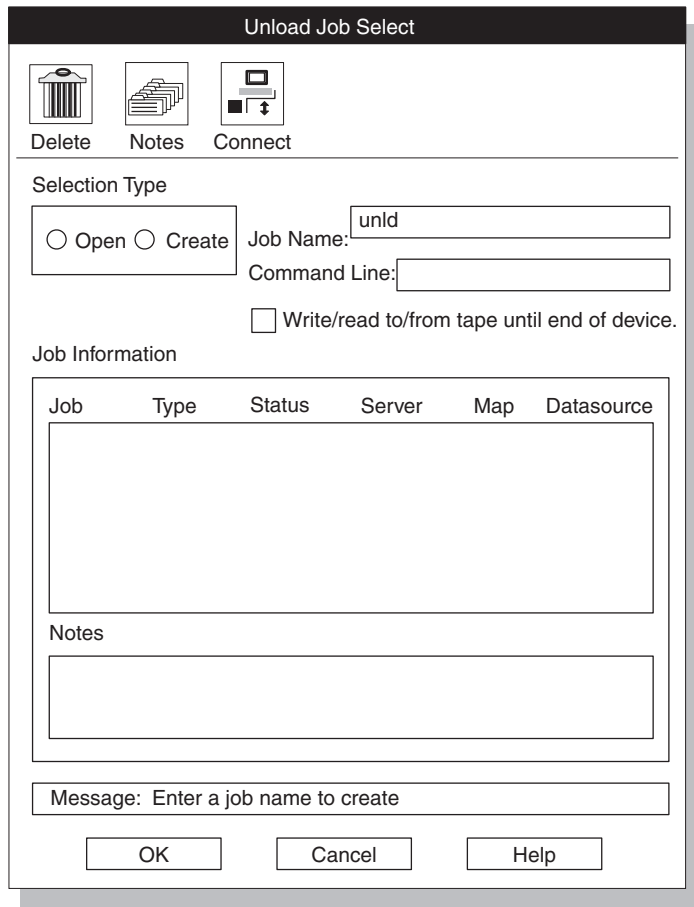


Figure 2-18. The Unload Job Select window

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.

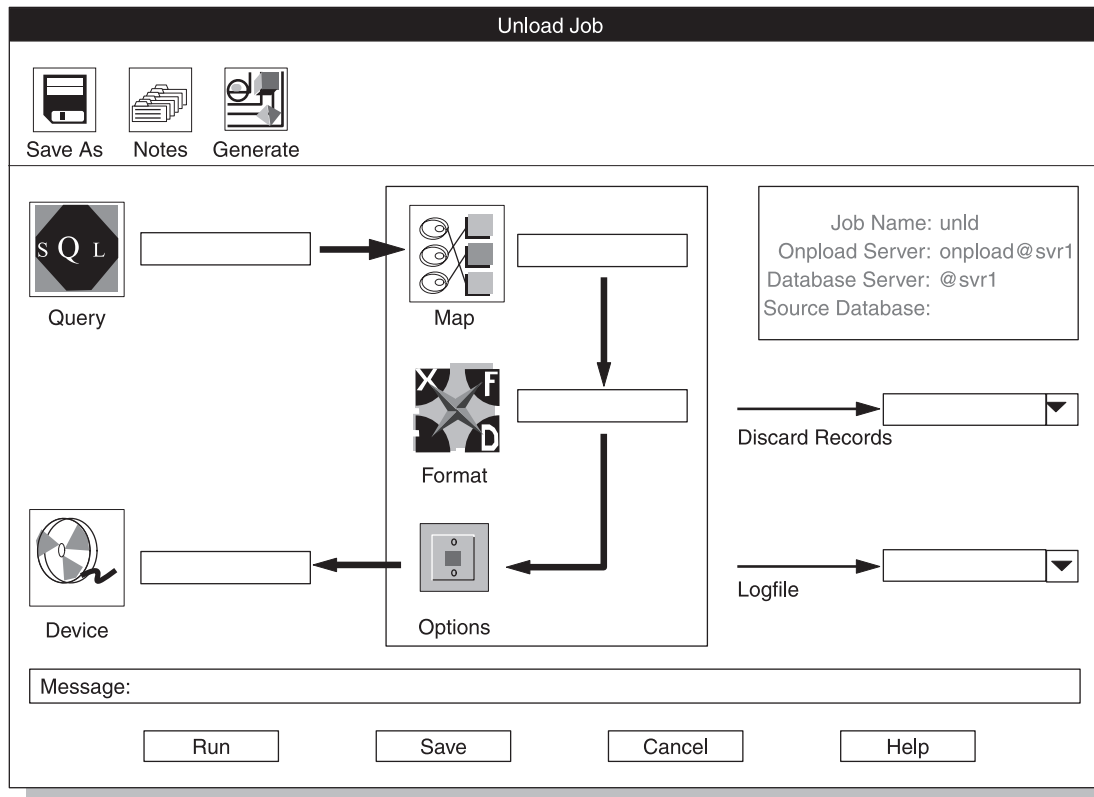


Figure 2-19. The Unload Job window

5. Click the **Generate** button.
The Autogenerate Unload Components window appears. Figure 2-20 on page 2-24 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 2-14 shows an example of a selection list.
8. Type items in the **Table** text box.

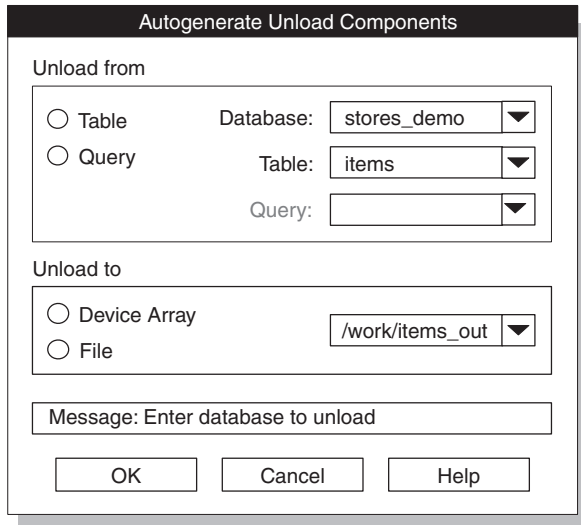


Figure 2-20. The Autogenerate Unload Components window

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 path name 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.

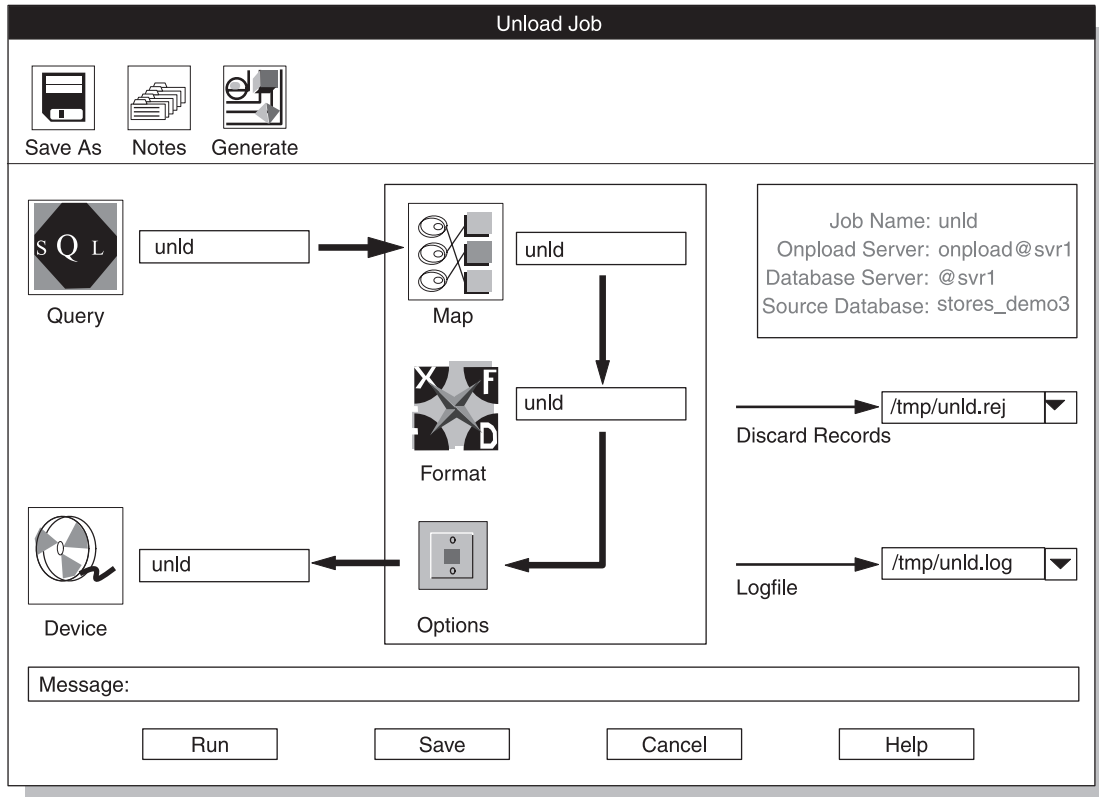


Figure 2-21. The Unload Job window

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 path names for the Discard Records file and the Logfile. Chapter 14, “HPL browsing options,” on page 14-1, describes the rejected records file and the log file.

Performing the unload job

You are now ready to perform the unload job.

Tip: At this point, you might want to preview the records that the query will select. See “Previewing data-file records” on page 14-1.

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 run the job, or you can return at a later time and run the job.

2. Click **Run** to run 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 2-20 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**.

Chapter 3. Using the ipload utility windows

Use the ipload utility GUI or the onpladm command-line interface	3-1
Starting the ipload utility	3-1
The ipload utility GUI	3-2
HPL main window	3-2
Initial options on the HPL main window	3-2
Options of the HPL main window	3-3
Component-selection windows.	3-3
Component-definition windows	3-5
Load job and unload job windows	3-6
Views windows.	3-7
Accessing views windows	3-7
Available options in a views window	3-8
Selection-list windows	3-9
Message windows	3-10
Using the HPL ipload utility buttons	3-11
HPL ipload utility toolbar buttons	3-11
Browse button.	3-12
Copy button	3-12
Delete button	3-13
Notes button	3-14
Print button	3-15
HPL ipload utility icon buttons	3-16
Buttons at the bottom of the HPL ipload utility display	3-18
Using the HPL online help.	3-18
Using UNIX keyboard commands to move the cursor	3-19

Use the ipload utility GUI or the onpladm command-line interface

The **ipload** utility is a graphical user interface (GUI) that contains windows, buttons, online help, and keyboard commands. The **onpladm** command-line interface is equivalent to the **ipload** utility.

This chapter contains information on using the **ipload** utility.

For information on using **onpladm** instead of using the **ipload** utility, see the Chapter 17, “The onpladm utility,” on page 17-1 chapter in this guide.

Starting the ipload utility

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

```
ipload
```

A 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 with your command, as follows:

```
ipload -n
```

The ipload utility 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

Important: While the **onpload** and **onpladm** utilities include support for object names that contain up to 128 characters, the **ipload** utility does not. If you use long database, table or column names and create jobs using **onpladm**, you cannot run these jobs using **ipload**. For **ipload**, database, table and column names cannot exceed 18 characters.

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.

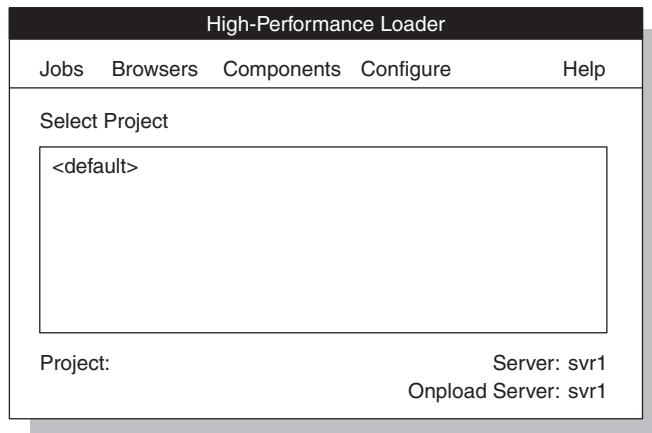


Figure 3-1. The HPL main window

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 4-4.
- 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 5-1.
- 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	See
Jobs	Load	Create a load job and use the Load Job window to load data into a database.	12-1
	Unload	Create an unload job and use the Unload Job window to unload data from a database to a file.	11-2
	Exit	Exit from the user interface.	2-2
Browsers	Record	Review records in a specified format, search the list of available formats, or edit a format.	14-3
	Violations	View records that passed the filter and conversion but were rejected by the database.	14-4
	Logfile	View load status and see where any errors occurred.	14-5
Components	Formats	Create or modify data-file formats.	7-1
	Maps	Create or modify maps that show the relationship between data-file fields and database columns.	9-1
	Query	Build, modify, or retrieve SQL-based queries.	8-1
	Filter	Create or modify filters that determine source data-file records for conversion and load.	10-1
	Devices	Specify a set of files, tapes, or pipes (UNIX only) that will be read simultaneously for loading or unloading the database.	6-1
	Generate Job	Automatically generate the components for load and unload jobs.	13-1
Configure	Server	Select the database servers that hold the onpload database and the target database.	5-1
	Project	Create a project under which formats, filters, queries, maps, and load and unload jobs are stored.	4-1
	Defaults	Specify the default character sets for the data file and databases.	5-2
	Machines	Specify the machine parameters that are used to convert binary data.	5-6
Help	Glossary	View definitions of terms that pertain to the HPL.	3-18
	Contents	View the main contents page that directs you to discussions of various HPL topics.	3-18

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.

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

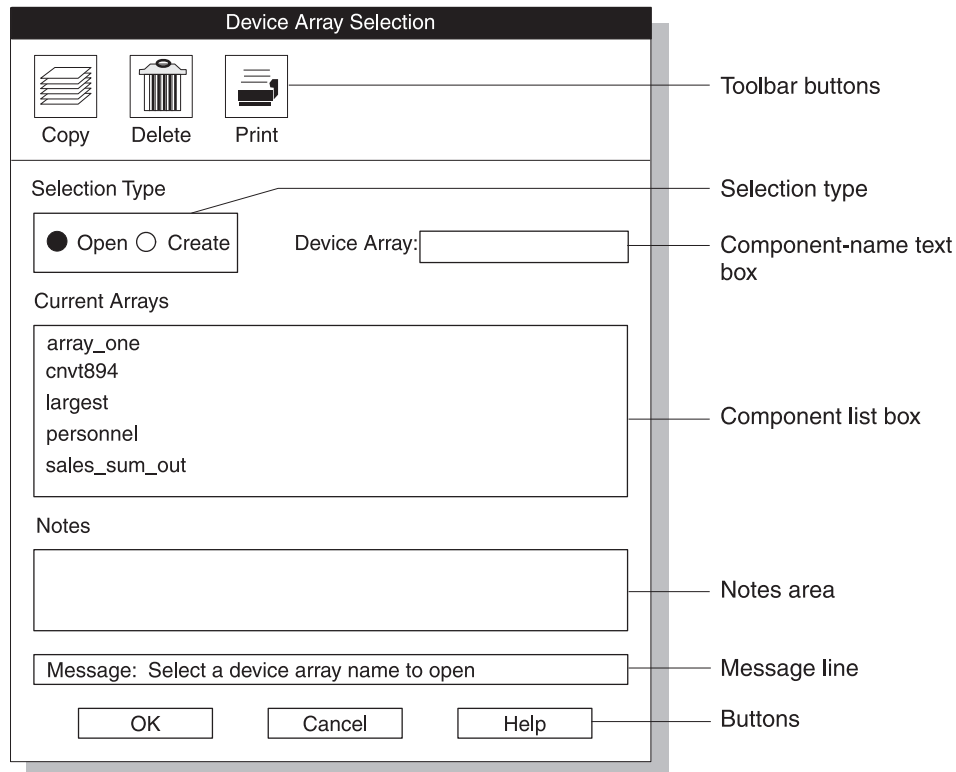


Figure 3-2. The Device Array Selection window

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 ipload utility buttons” on page 3-11 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.)

Device Array text box: 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 3-14.

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 iupload utility buttons” on page 3-11.

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.

Figure 3-3 shows an example of a Component-Definition window: the Device-Array Definition window.

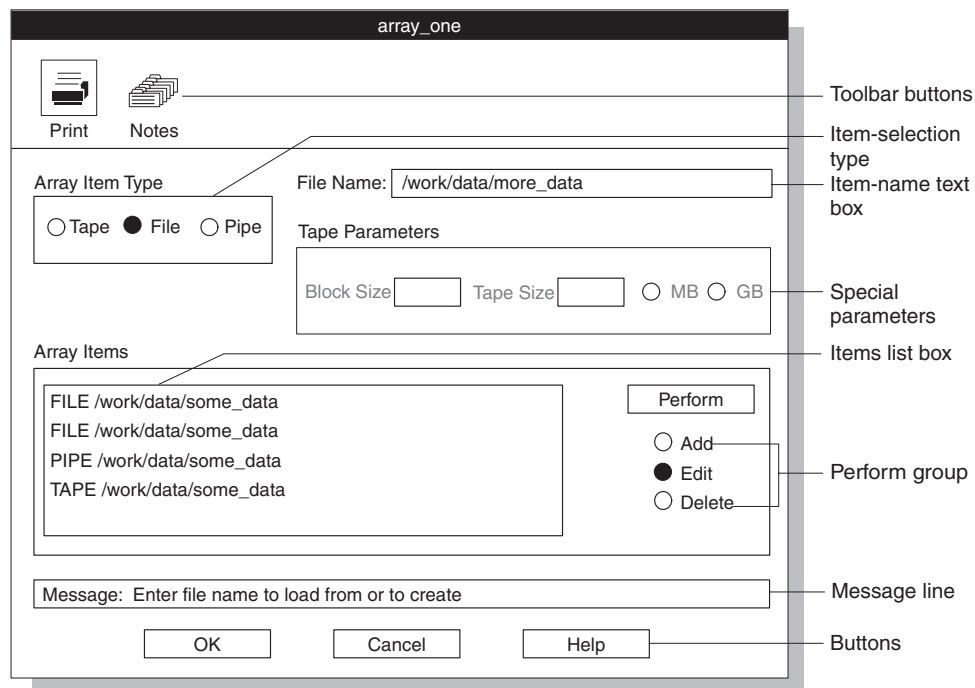


Figure 3-3. The Device-Array Definition window

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 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 iupload utility buttons” on page 3-11 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 path name 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 3-5) 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 (UNIX only) 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 returns 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 ipload utility buttons” on page 3-11.

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.

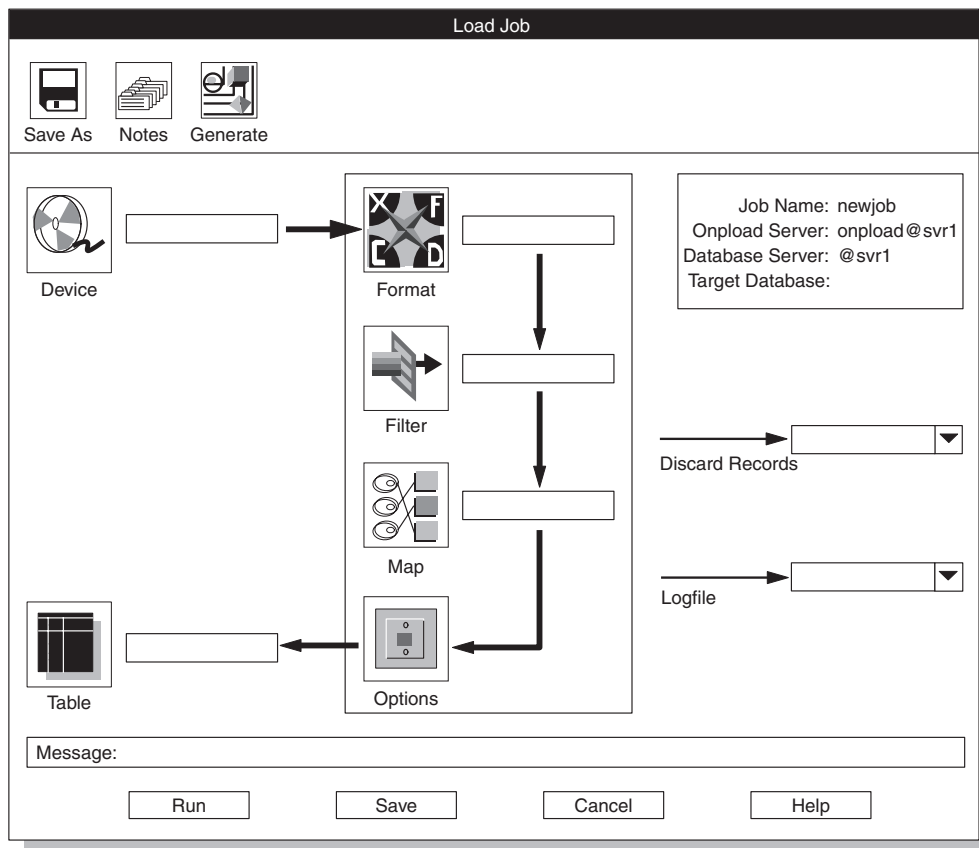


Figure 3-4. The Load Job window

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	See
Format Views	Shows 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* 	2-10

Window Name	Purpose	How to Access	See
Map Views	Shows 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* 	9-4
Database Views	Shows 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* 	8-2
Filter Views	Shows 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* 	10-6

* 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 in a views window: 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 in a views window: 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.

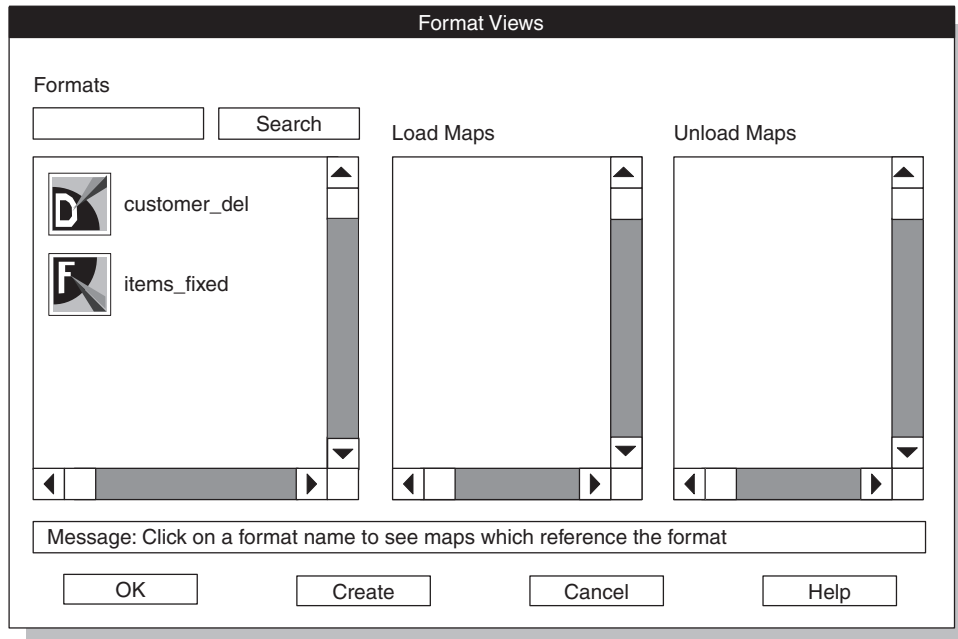


Figure 3-5. The Format Views window

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.

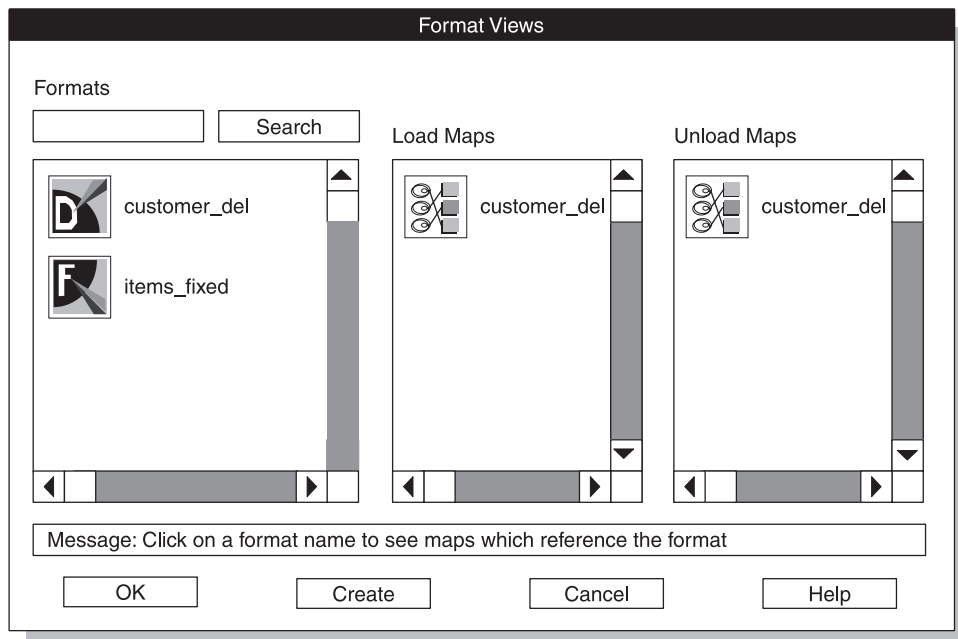


Figure 3-6. Expanded view of a format

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.

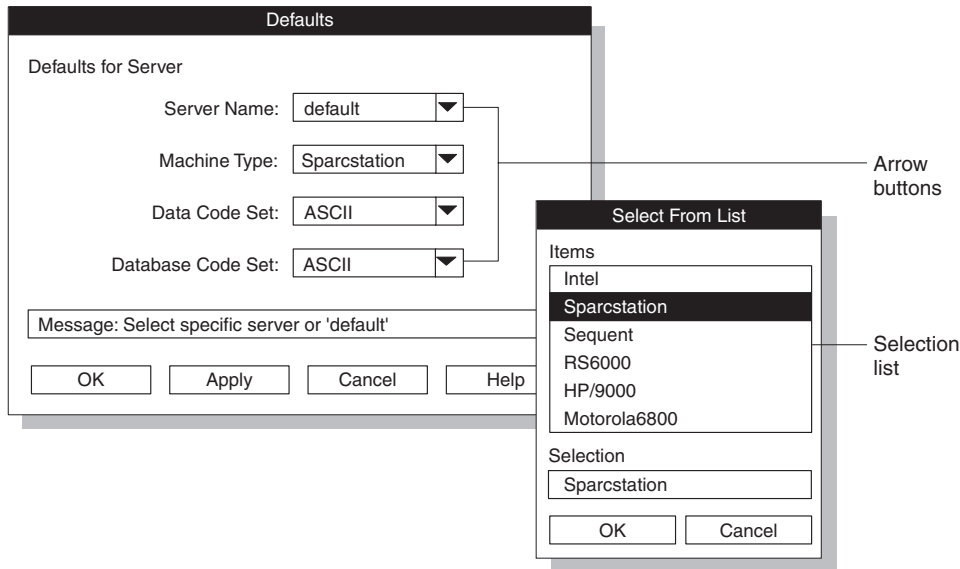


Figure 3-7. The Defaults window and a selection list

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 iupload utility buttons

After you move beyond the 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.

HPL iupload utility 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	See
	Browse	Displays the Browse window	14-1
	Copy	Copies the selected component (format, map, query, filter, device, or project) to a new item	3-12
	Connect	Lets you reattach to an active unload (or load) job from the Unload (or Load) Job Select window	12-4
	Delete (trash can)	Deletes the selected component (format, map, and so on)	3-13
	Delete (eraser)	Breaks the association between a database column and a data-file field	9-11
	File	Displays the Import/Export File Selection Window	8-8
	Find	Allows you to quickly locate a particular field or column in a map window	9-11
	Generate	Lets you generate jobs automatically	13-1
	Notes	Allows you to type descriptive text for an item	3-14
	Options	Displays an options window where you can change default values or supply additional parameters	11-7, 12-8
	Print	Prints the parameters for the selected item	3-15
	Save As	Saves a copy of the currently selected item (behaves in the same way as the Copy button)	3-12
	Search	Displays a Views window where you can see the relationships among components	3-7
	Specs	Displays the Specifications window, where you can view the attributes for selected columns or fields	9-12

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 14-1.

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 7-2.
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.

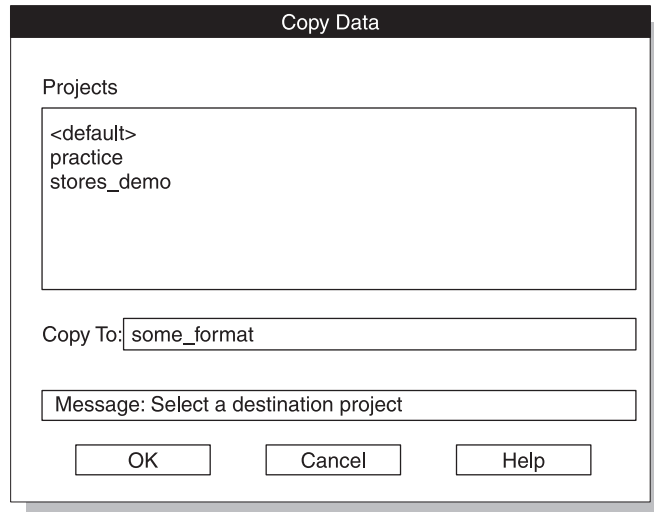


Figure 3-9. The Copy Data window

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 7-2.
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.

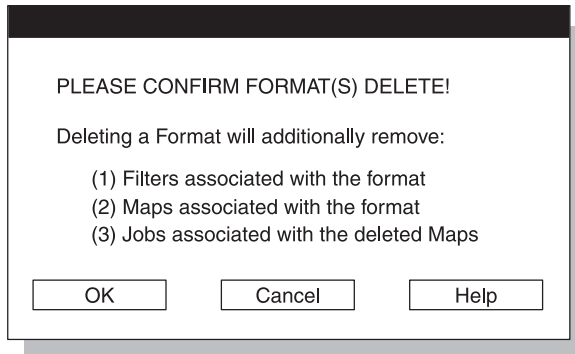


Figure 3-10. The Confirm Delete window

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.

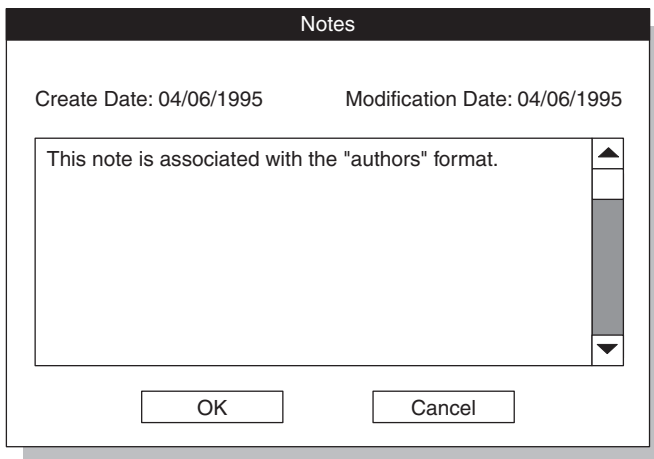


Figure 3-11. The Notes window

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.

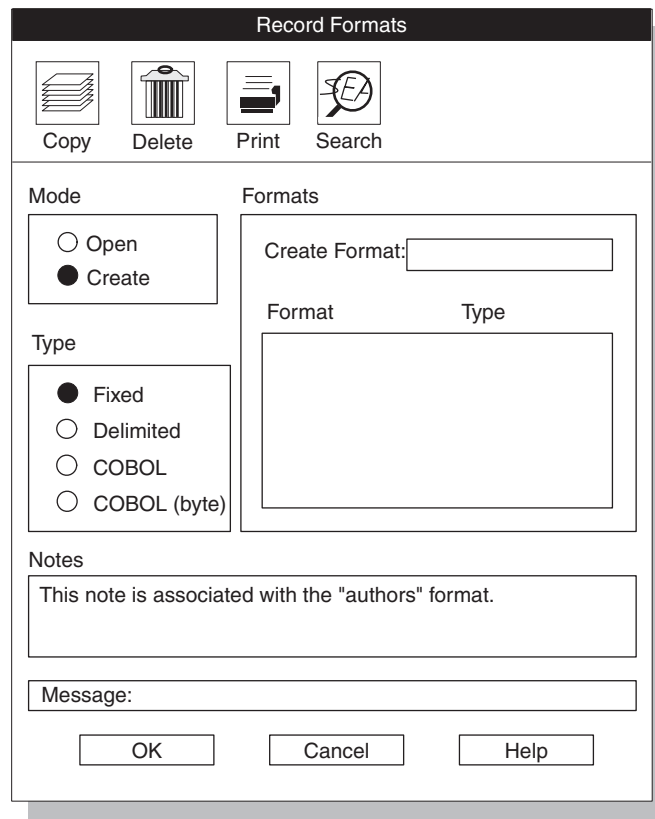


Figure 3-12. The Record Formats window with Notes text

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 in the onpload database” on page A-9.

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 publications.

If you click the **Print** button in the Map-Definition window in Figure 2-14 on page 2-17, 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





```


Figure 3-13. Print button output

HPL ipload utility 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.

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 12-8.
 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.

Component	Description	Window	Action
 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.

Buttons at the bottom of the HPL iupload utility display

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 HPL online help” on page 3-18.
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 3-14 shows the use of **OK** and **Cancel**.

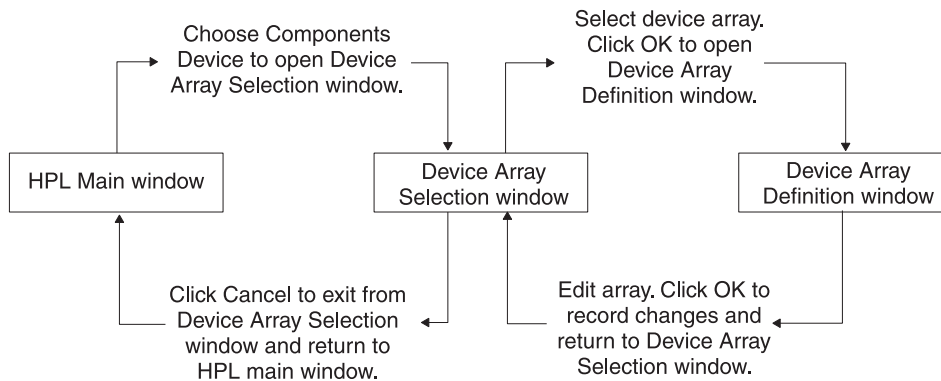


Figure 3-14. Using OK and Cancel from the HPL Main Window

Using the HPL 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 UNIX 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.

Chapter 4. Defining HPL projects

HPL projects	4-1
Project organization	4-1
Select or create a project with the Projects window	4-3
Creating a new project	4-4
Selecting a project	4-4

HPL projects

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.

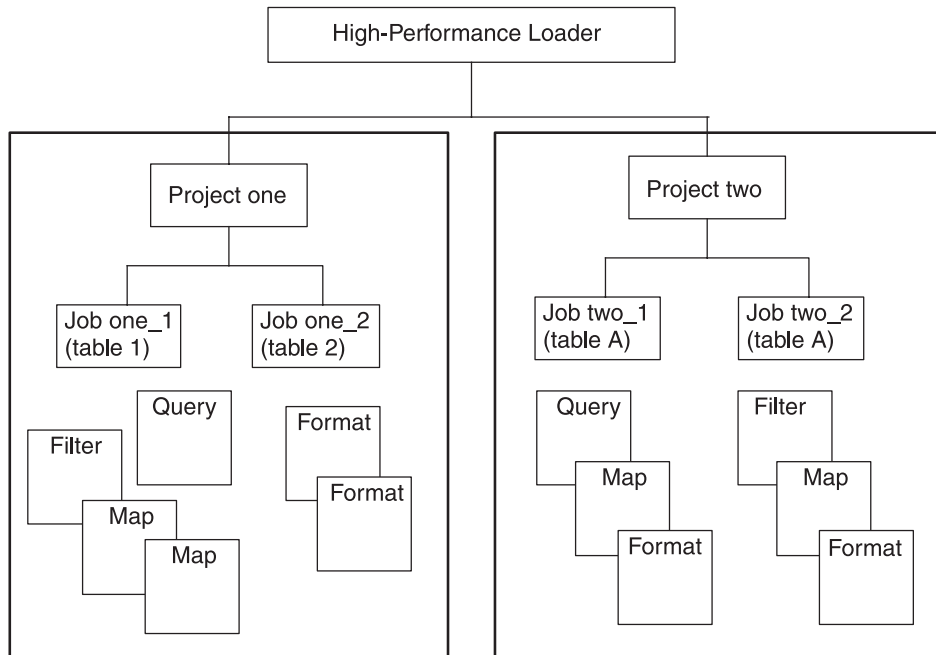


Figure 4-1. Illustration of project hierarchy

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.

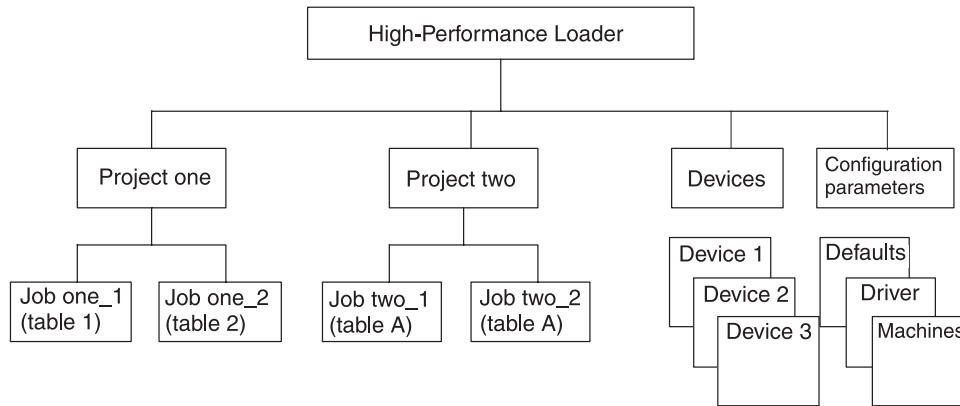


Figure 4-2. Relationship of projects, devices, and configuration parameters

Select or create a project with the 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 here: A-9.)

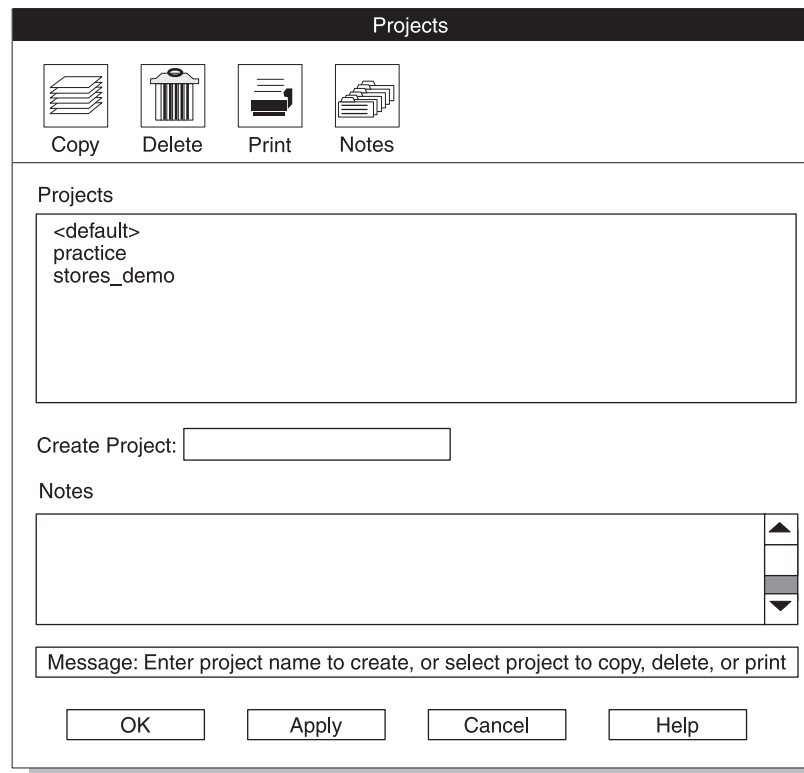


Figure 4-3. The Projects window

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.

Chapter 5. Configuring the High-Performance Loader

Configuring ipload	5-1
Selecting a database server	5-1
Using the Connect Server window	5-1
Creating the onpload database.	5-2
Modifying the onpload default values	5-2
Defaults window	5-2
Changing the onpload defaults	5-4
Selecting a driver	5-4
Drivers window	5-4
Using the Drivers window	5-5
Modifying the machine description	5-6
Machines window	5-6
Using the Machines window	5-7

Configuring ipload

When you configure the **ipload** utility, you describe the type of computer, code sets, and other aspects of your database server environment. 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

“HPL performance” on page 15-6 describes how to modify the configuration to improve performance.

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 5-2) 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.

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.

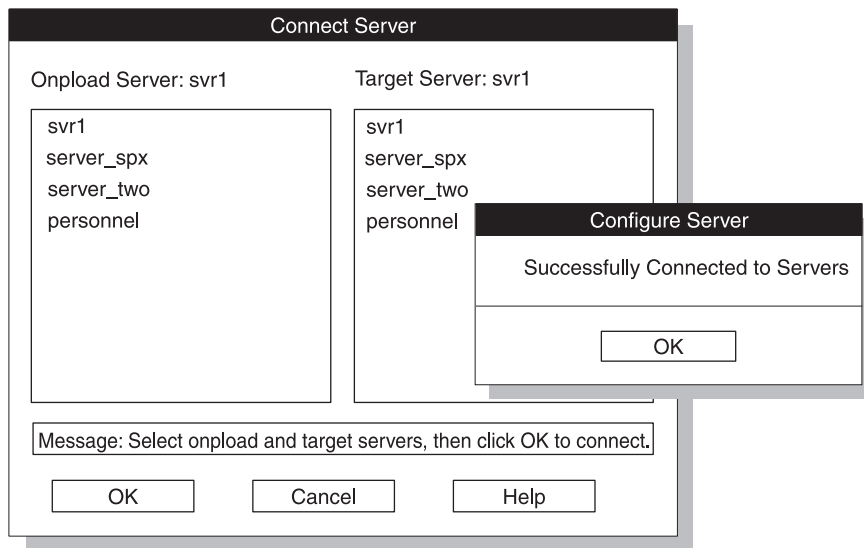


Figure 5-1. The Connect Server window

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 1-7.

Modifying the onpload default values

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.

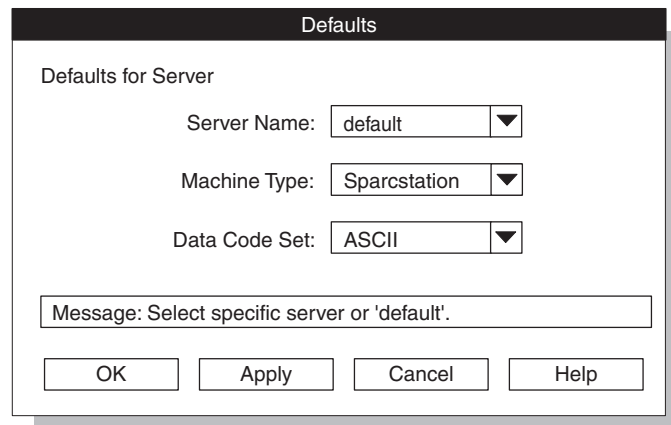


Figure 5-2. The Defaults window

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 A-1.

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 *IBM Informix 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 5-6).

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.

Global Language Support

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*.

End of Global Language Support

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 5-3 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**.

Selecting a driver

UNIX Only

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**.

End of UNIX Only

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-10 to write the driver.
2. Add the driver name to the **onpload** database.
For details, see “Using the Drivers window” on page 5-5.
3. Select the driver from the **Driver** selection list in the Format Options window.
For details, see “Format options” on page 7-18.

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.

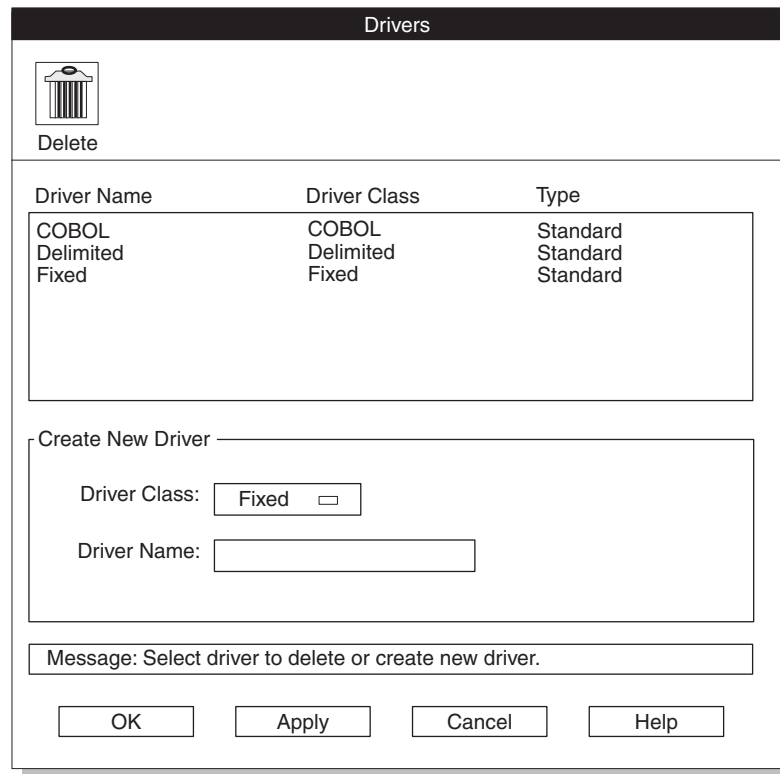


Figure 5-3. The Drivers window

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 “Rebuilding the shared-library file” on page H-3.

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 5-5 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

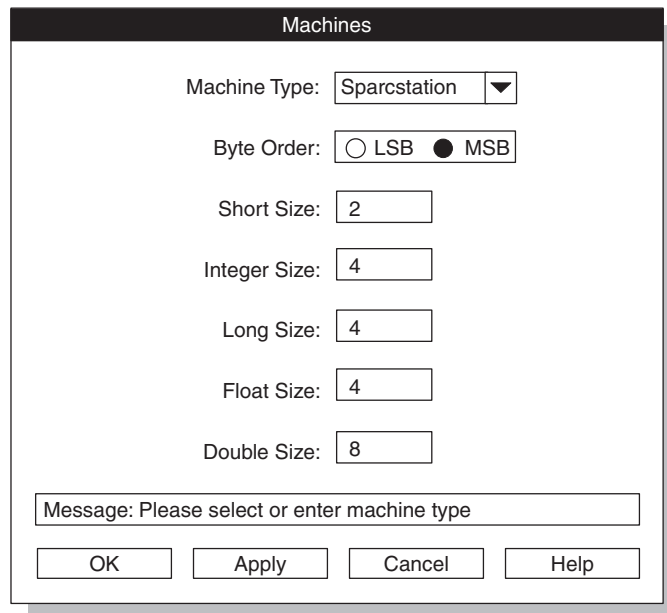
The information that the **Machines** option of the **Configure** menu stores describes the characteristics of a specific computer. The HPL uses these characteristics to control the conversion of binary data formats. Unless you are converting binary data, you do not need to be concerned about the machine 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 5-4.) The information from the Machines window is stored in the **machines** table of the **onpload** database. (For more information on the **machines** table, see A-6.) 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 dialog box titled "Machines". It contains the following fields and controls:

- Machine Type: A dropdown menu with "Sparcstation" selected.
- Byte Order: Radio buttons for "LSB" (unselected) and "MSB" (selected).
- Short Size: A text input field containing "2".
- Integer Size: A text input field containing "4".
- Long Size: A text input field containing "4".
- Float Size: A text input field containing "4".
- Double Size: A text input field containing "8".
- A message box at the bottom containing the text: "Message: Please select or enter machine type".
- Four buttons at the bottom: "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 type 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
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

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 5-6 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 5-6 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**.

Chapter 6. Defining device arrays

Device arrays	6-1
Using multiple devices in a device array	6-1
Using the Device-Array Selection window	6-1
Using the Device-Array Definition window	6-2
Adding, editing, and removing devices	6-4

Device arrays

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 set up simultaneous access to one or more tape devices, files, or pipes (UNIX only) 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 (UNIX only), and tape devices in a single array. “Devices for the device array” on page 15-8 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.

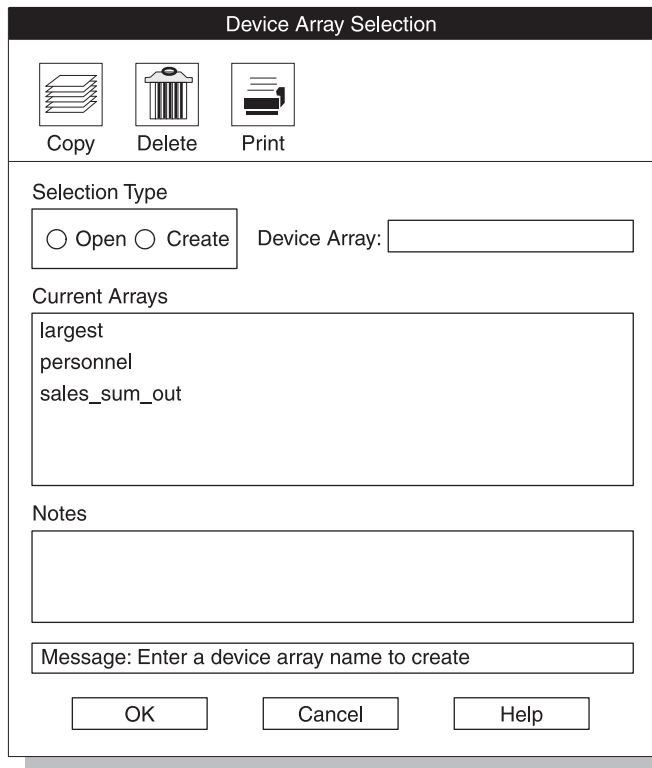


Figure 6-1. The Device-Array Selection window

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 6-2 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 6-3 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 6-2 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 6-3 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.

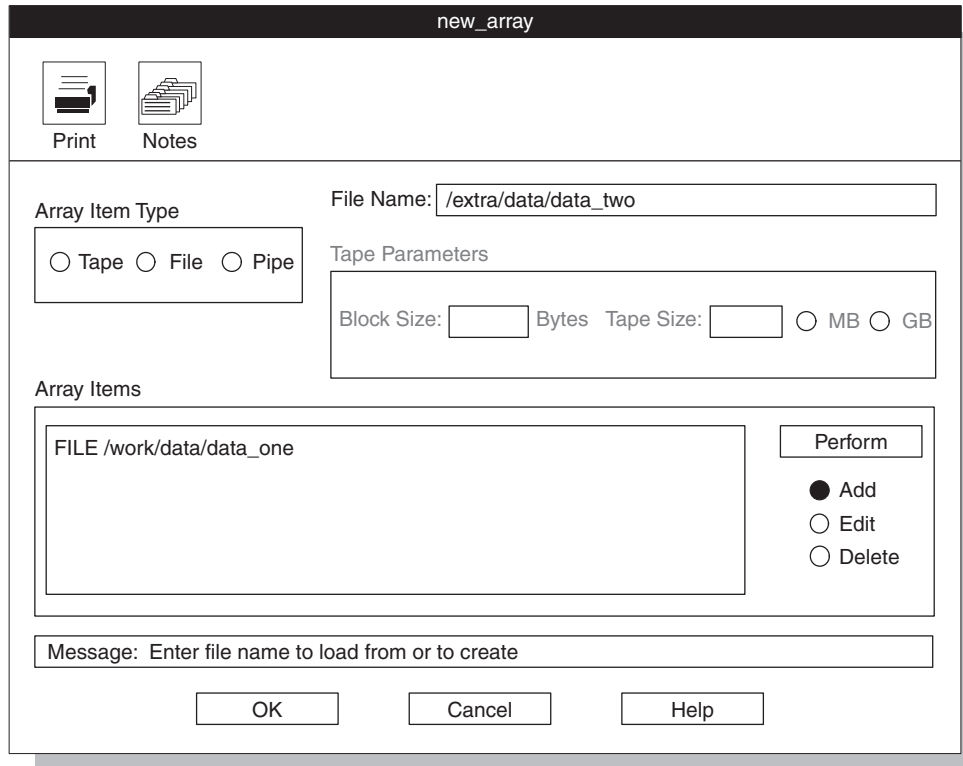


Figure 6-2. A Partially Completed Device-Array Definition window

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** (UNIX only). Fill in this text box as follows.

Device Type	What to Type
Tape Device	The full path name of the tape device (example: /dev/rmt/0)
File Name	The full path name of the file (example: /work/mydata)
Pipe Command	The full path name of the executable pipe command or shell script (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**.

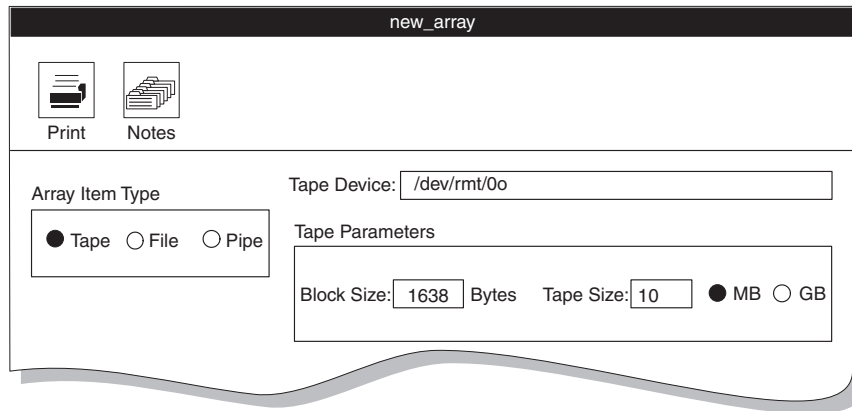


Figure 6-3. The Tape Parameters Group

Windows Only

Windows does not support remote tape load and unload.

End of Windows Only

Adding, editing, and removing devices

You can add devices to the device array, you can edit the devices, and you can remove them.

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 path name 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 6-4 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. Specify 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 path name 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 path name 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.

Chapter 7. Defining formats

Formats	7-1
Formats of supported data-file records	7-2
Fixed-length records	7-2
Creating a fixed format	7-2
Data types allowed in a fixed format	7-5
Editing a format	7-6
Creating a fixed format that uses carriage returns	7-7
Creating a fixed format that includes BYTE or TEXT data.	7-8
Inline data	7-8
Data in a separate file.	7-9
Creating a fixed format that includes Ext type or simple LO data.	7-10
Fixed-length data	7-10
Inline data	7-10
Simple LO data in a separate file	7-11
Delimited records.	7-12
Creating a delimited format	7-12
Data types allowed in a delimited format.	7-12
Creating a delimited format that includes BYTE or TEXT data.	7-13
Creating a delimited format with extended data types	7-14
COBOL records	7-15
Creating a COBOL format	7-16
Picture and usage descriptions	7-17
Picture description	7-17
Usage description.	7-17
Packed-decimal conversions	7-17
Other formats	7-18
Fast format	7-18
Fast job	7-18
Format options	7-18
Modifying fixed and COBOL formats	7-18
Modifying delimited-format options	7-19
Using a CSV file as ipload utility input	7-20
Format Views window	7-22

Formats

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 publication 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, “The Generate options of the iupload utility,” on page 13-1.

Formats of supported data-file records

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 14-1.

The **iupload** 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 7-18.

The **iupload** 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 A-3 and A-6, 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.



```
aaabbbbcccdggghhh
```

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 **iupload** 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.

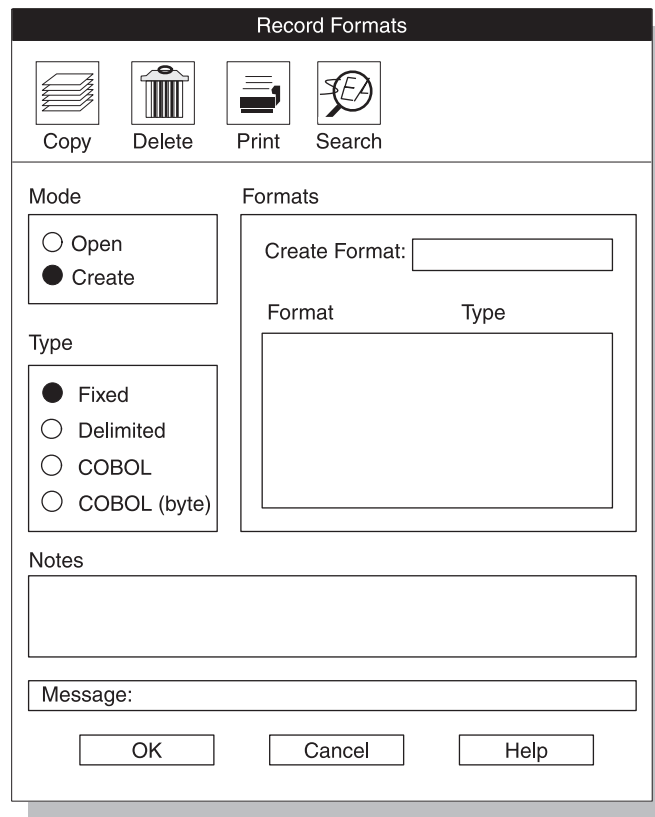


Figure 7-2. The Record Formats window

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 7-2 shows.

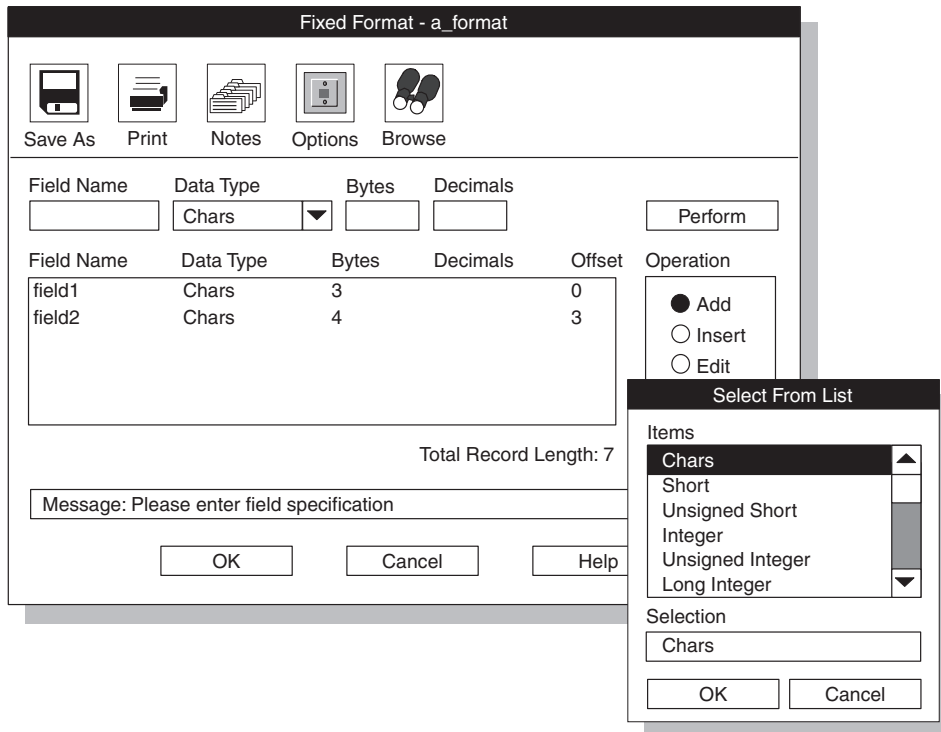


Figure 7-3. A completed Fixed-Format definition window with an open selection list

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 7-5 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 5-6.) 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) that follow this record
Blob File	A file that contains BYTE and TEXT data
BLOB and CLOB	A file that contains smart large object (BLOB or 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
BIGINT	An ASCII character string or an eight-byte integer type
BIGSERIAL	An ASCII character string or an eight-byte integer type
Ext Type String	The ASCII representation of an extended data type (Ext Type) value
Ext Type String Length	The length of an ASCII Ext Type value. The Ext Type value follows at the end of the input/output record. Use Ext Type String Length data type if you have data that contains null UDT values.
Ext Type Binary	The binary representation of an Ext Type value
Ext Type Binary Length	The length of the binary representation of an Ext Type value. The binary value follows at the end of the input/output record. Use Ext Type Binary Length data type if you have data that contains null UDT values.

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 7-4, 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 5-6. 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 7-4, 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 7-2.
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 7-2.
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 7-2.
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 7-2.
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 9-3), 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.

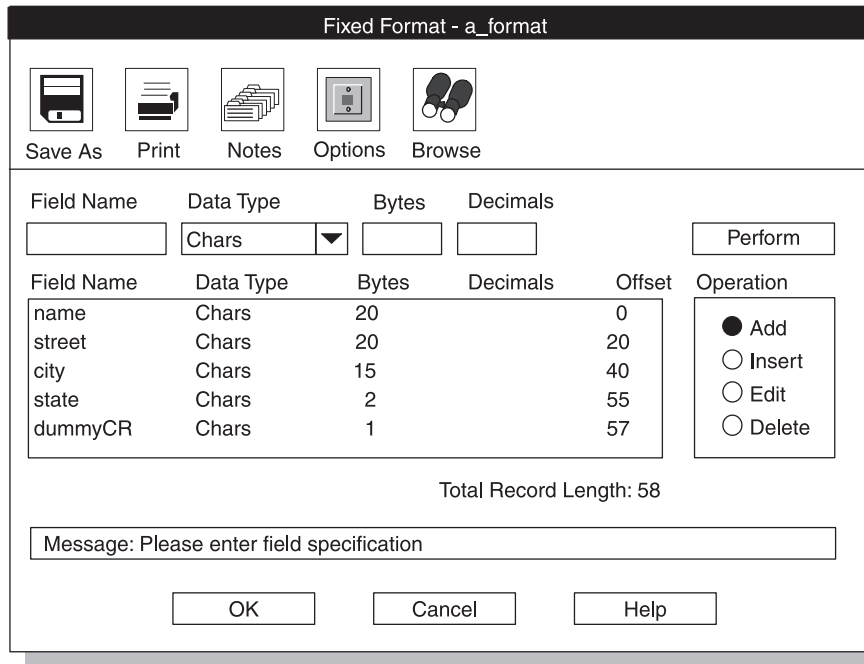


Figure 7-5. Fixed format with dummy entry for carriage return

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.

```
field1  textlength  field2  textdata
```

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.

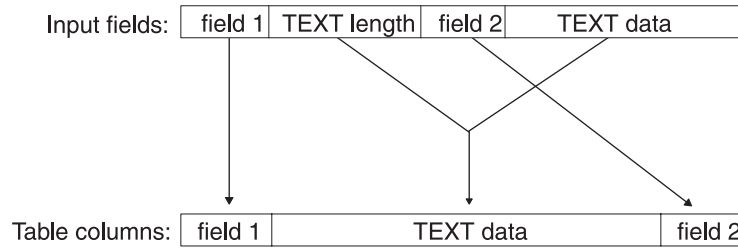


Figure 7-7. Inline TEXT Data

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.

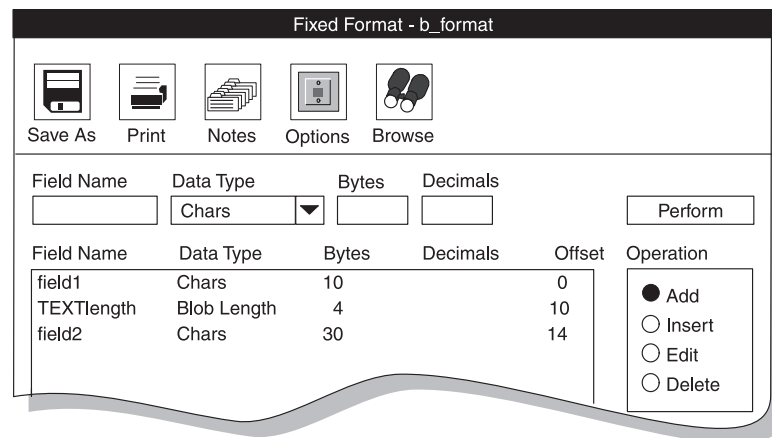


Figure 7-8. Fixed format that includes TEXT data

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 job, the file is created, and BYTE and TEXT data is written to the file. When the fixed-format input contains the path name of a data file, the HPL uses that path name 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.

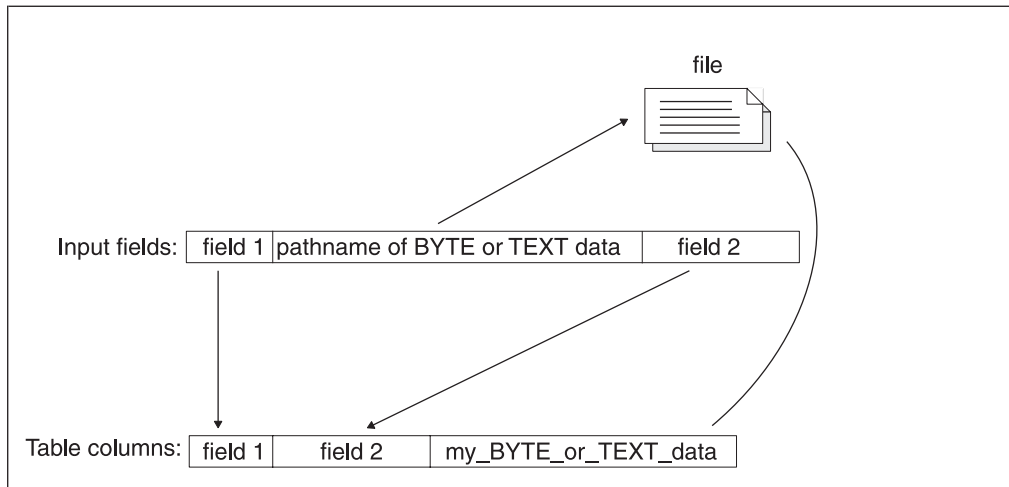


Figure 7-9. BYTE or TEXT data in a file

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 7-11 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.

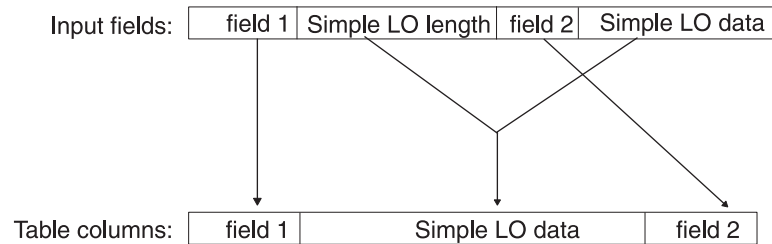


Figure 7-11. Inline TEXT data

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.

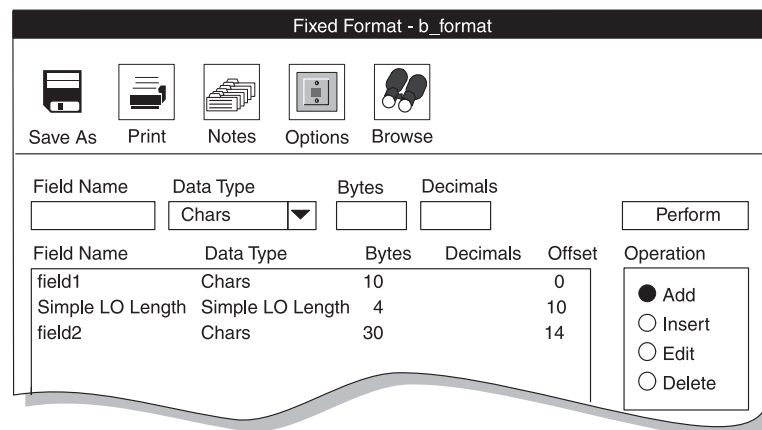


Figure 7-12. Fixed format that includes a simple LO

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 job, the file is created and BYTE and TEXT data is written to the file. When the fixed-format input contains the path name of a data file, the HPL uses that path name 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.

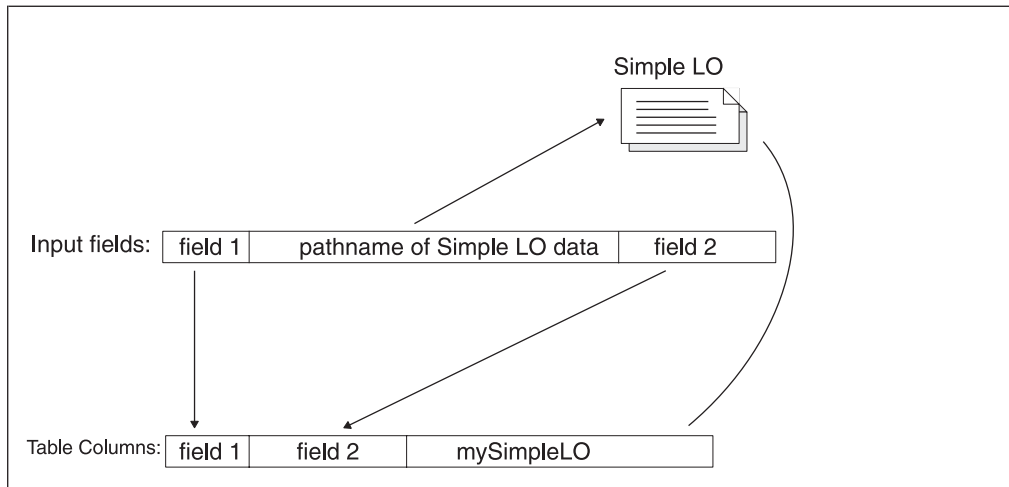


Figure 7-13. Simple LO data in a file

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 7-19.

Creating a delimited format

To create a format for delimited records, follow the same steps as in “Creating a fixed format” on page 7-2, 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. The onpload utility translates the data into binary format before it loads the data into the database.
BLOB or CLOB	File that contains smart large object (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. The onpload utility translates the data into binary format 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 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 path name 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.

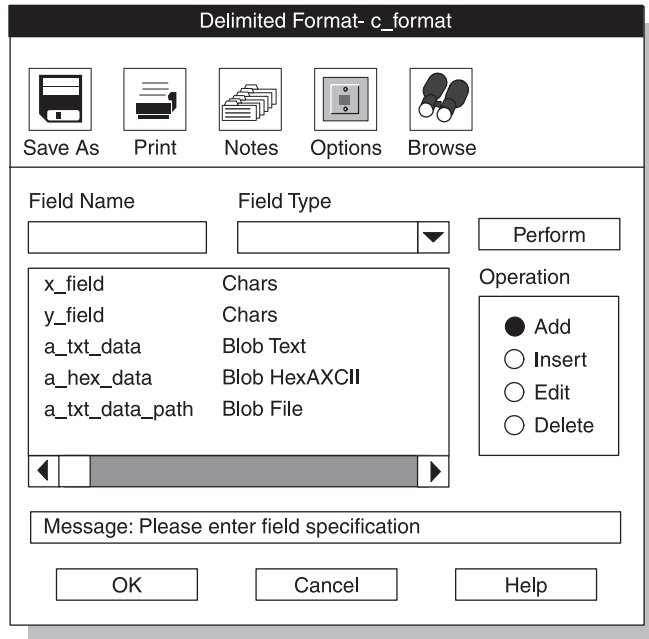


Figure 7-15. Delimited format with BYTE or TEXT entries

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 is always written to a file. CLOB data can be ASCII or hexadecimal data. BLOB data can be binary data. The path name 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 path name.

```
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.

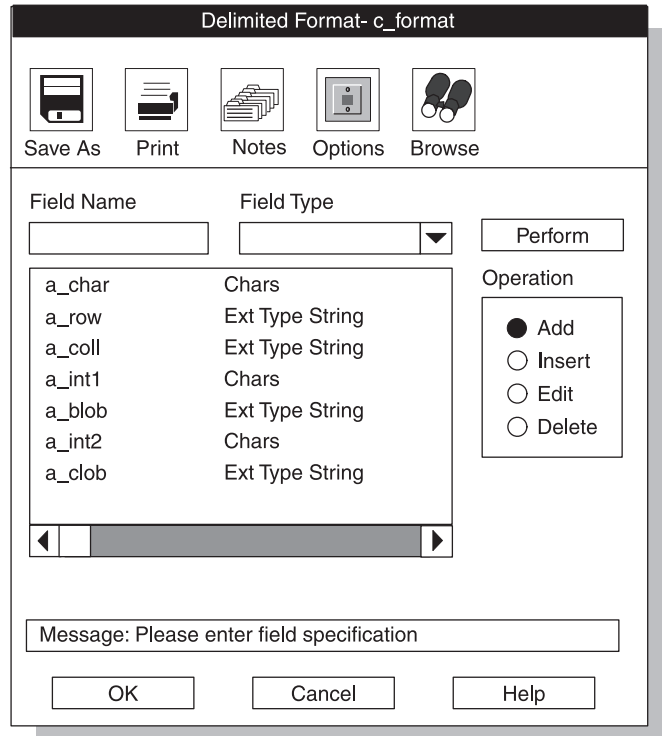


Figure 7-17. Delimited format with extended data type entries

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 path name.

```
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.

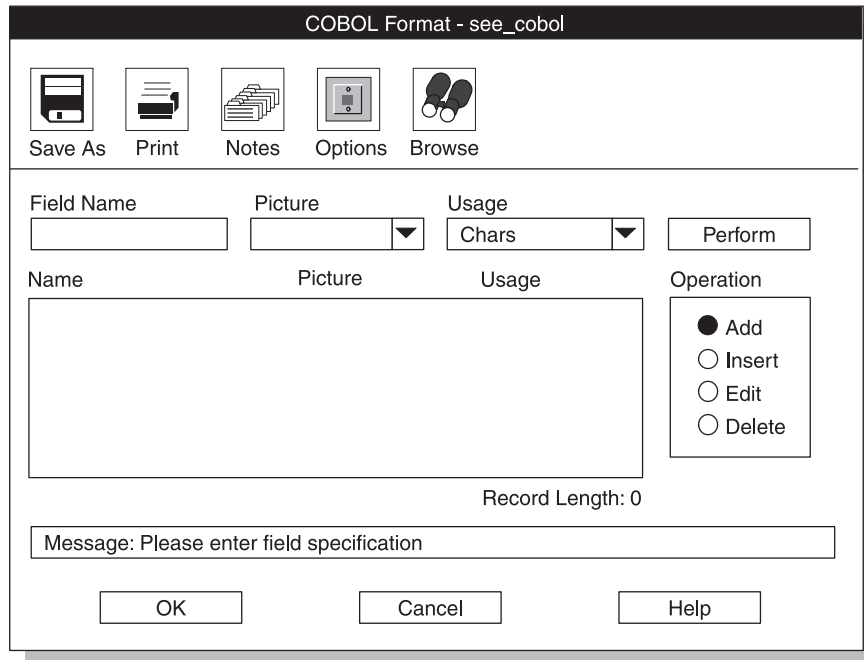


Figure 7-19. Fixed-format definition window for a COBOL format

Creating a COBOL format

To create a format for COBOL records, follow the same steps as in “Creating a fixed format” on page 7-2, 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.

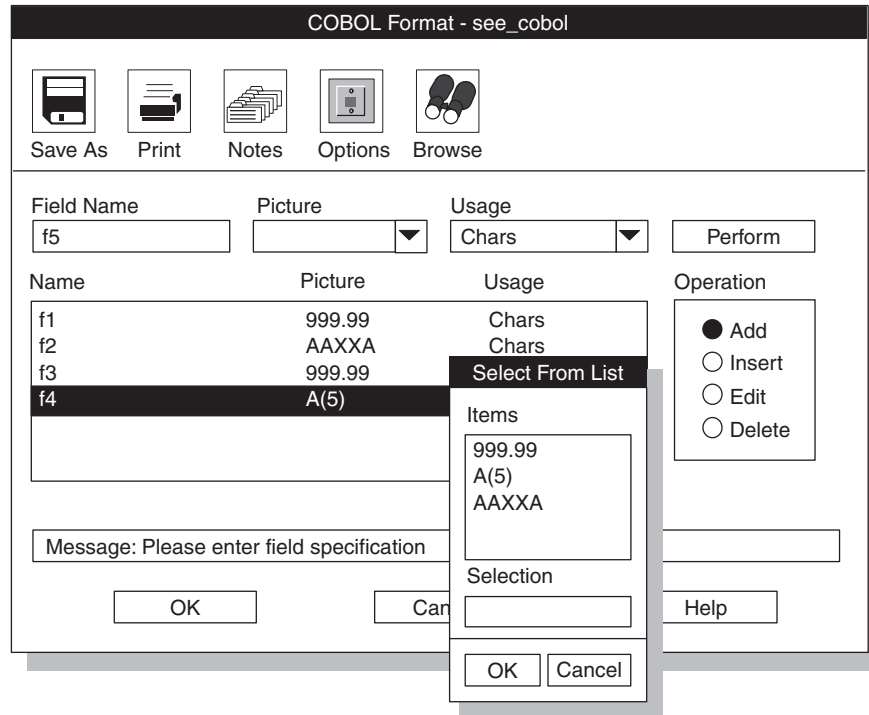


Figure 7-20. Fixed-format definition window for a COBOL format

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 publication.

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, “The Generate options of the ipload utility,” on page 13-1.

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 13-6.

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,” on page 16-1.

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 A-6.

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 5-4.
Machine	The machine type that produced the data files. For more information, see “Modifying the machine description” on page 5-6.

Global Language Support

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*.

End of Global Language Support

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 7-2.
2. Click **Options**.
The Options window (in this example, the Fixed Format Options window) appears, as Figure 7-21 shows.

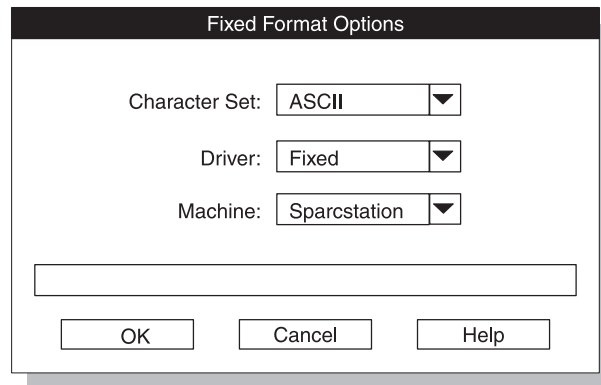


Figure 7-21. The Fixed Format Options window

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 5-4.
Delimiting characters	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. You can specify the delimiting characters in ASCII, HEX, OCTAL, or DECIMAL format.

Global Language Support

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*.

End of Global Language Support

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 7-2 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.

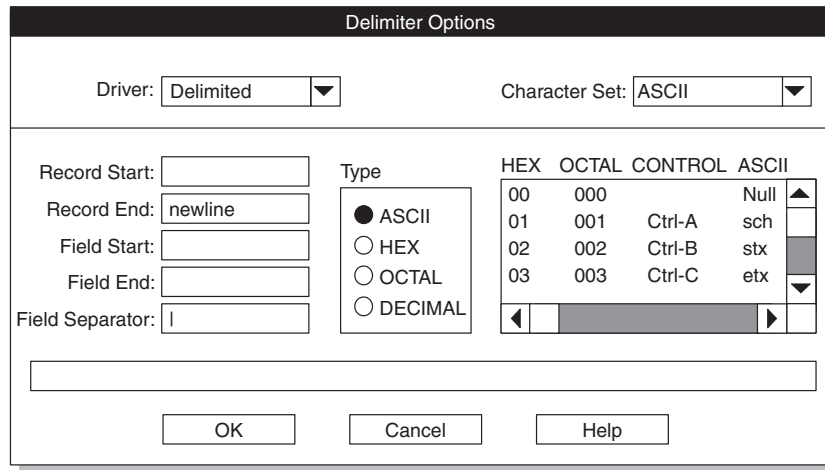


Figure 7-22. The Delimiter Options window

3. Modify the options that you want to change, or add information.
For example, add **Field Start** and **Field End** information to mark the beginning and end of a field (column).

The default escape character is \ (the backslash).

If the **Field Start** and **Field End** values contain both single or double quotes and specified characters are not present in a field, an error occurs. The error occurs because the input does not match the expected layout. If one of these characters appears in a field, you must mark the character with the escape key (to escape the character).

For comma-separated values (CSV), enter , (a comma) in the **Field Separator** field. For more information, see “Using a CSV file as **ipload** utility input.”

An error can also occur if the **Field Separator** value appears inside a quoted field or any special character (such as the end field, end record, or escape character) appears in a field. In this case, you must also escape the character.

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*.

Using a CSV file as **ipload** utility input

Files containing comma-separated values (CSV) are a commonly used for transferring simple text data between programs. The CSV format uses a comma as

a field separator and a new line as a record separator . Double quotes are used to embed commas, new lines, or double quotes within strings. Some applications (for example, some spreadsheet applications) provide the option of exporting text data in the CSV format.

The following procedure shows how to test importing sample data from a spreadsheet into Informix.

To test importing a CSV file

1. Create a test database and simple table in Informix.
2. In your spreadsheet application:
 - a. Create a spreadsheet with columns that correspond to the columns in your Informix table.
 - b. Populate the columns with some test data.
 - c. Export the data by saving it in **.csv** format.
3. Copy the resulting **.csv** file and paste it in a location that you can access from **ipload**.
4. Start **ipload** and choose **Components > Generate Job** to create a new job.
5. In the **Generate Job** window:
 - a. Select **Load/Unload Job** in the **Generate** group.
 - b. Select **Delimited** in the **Format Type** group.
 - c. Specify the **.csv** file as the source for input to the new job you are creating. To do this, specify a name in the **Generate Name** field, select your database in the **Database** field, select you table in the **Table** field, and then insert the full path to the **.csv** file in the **Device** field.
 - d. Click **OK**.
6. Load the job by choosing **Jobs > Load**.
The **Load Job** window appears.
7. While you can do many things (such as filtering and mapping) on the **Load Job** window, for this test, select the **Format** button.
The **Delimited Format** window appears. You use this window to define the format of the input file.
8. In the **Delimited Format** window, click the **Options** button to display the **Delimited Options** window.
9. In the **Delimiter Options** window, change the value in the **Field Separator** field to **,** (a comma) and click **OK**.
The **ipload** utility returns you to the **Delimited Format** window.
10. In the **Delimited Format** window, click **OK**.
The **ipload** utility returns you to the **Load Job** window.
11. Click **Run** to run the job.
The **ipload** utility displays status information while the job runs, and then it displays the results of the job.
12. Click **OK** twice to return to the main **ipload** window.
13. As a sanity check, you can use DB-Access to verify that the data was successfully loaded.

For more information, see “Generating load and unload components” on page 13-7.

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 3-7 discusses how to use Views windows.

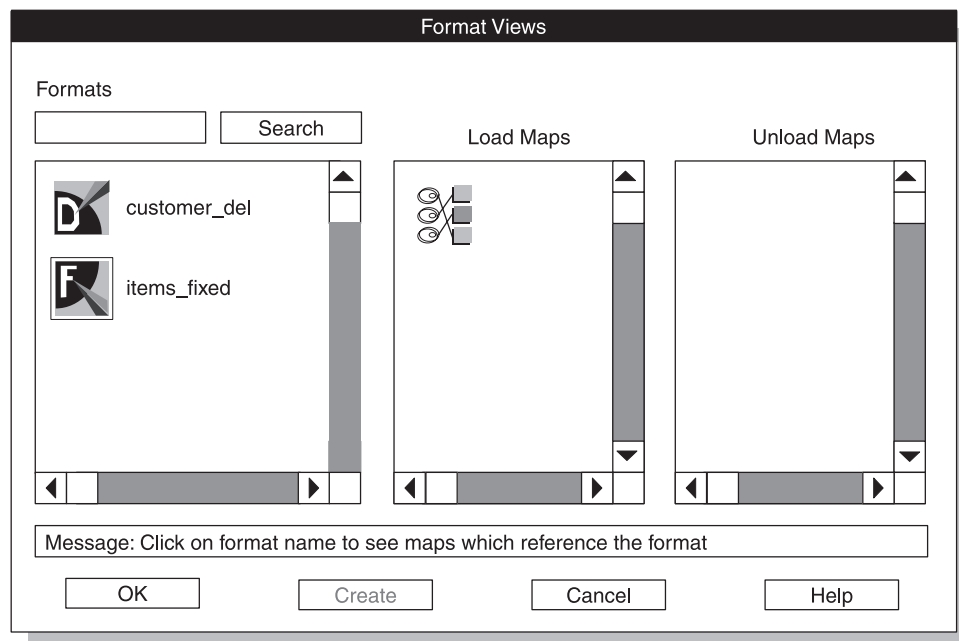


Figure 7-23. The Format-Views window

Chapter 8. Defining queries

HPL queries	8-1
Creating a query	8-1
Using the Table button	8-3
Editing the WHERE clause	8-6
Editing a query	8-7
Exporting and importing queries	8-8
Importing a query	8-8
Exporting a query	8-9
Database Views window	8-10

HPL queries

The **ipload** query component lets you build an SQL statement.

The **ipload** utility stores query information in the **query** table of the **onpload** database. (For more information about the **query** table, see A-10.) The SQL statement is stored as TEXT data.

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.

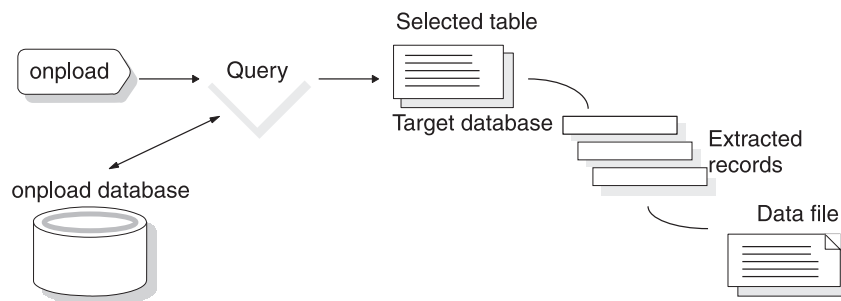


Figure 8-1. Extracting data from a database table

This publication 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

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.

- In the **Database** text box, type the name of the database that contains the tables 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.

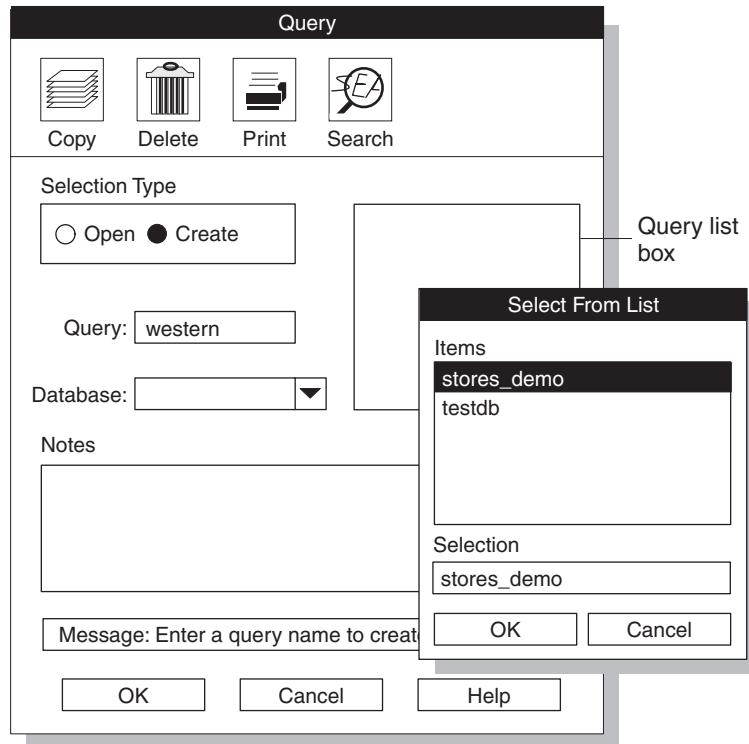


Figure 8-2. The Query window

- Click **OK**.

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

- 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.

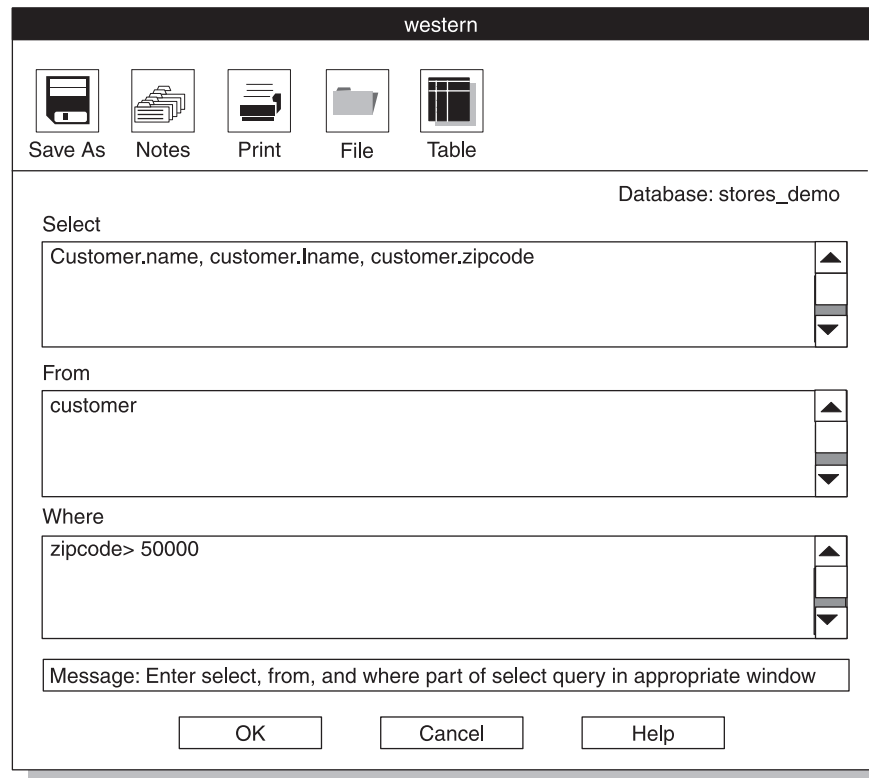


Figure 8-3. The Query-Definition window

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 8-1 to display the query-definition window (Figure 8-3).
2. Click **Table**.

The Column Selection window appears, as Figure 8-4 on page 8-4 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.

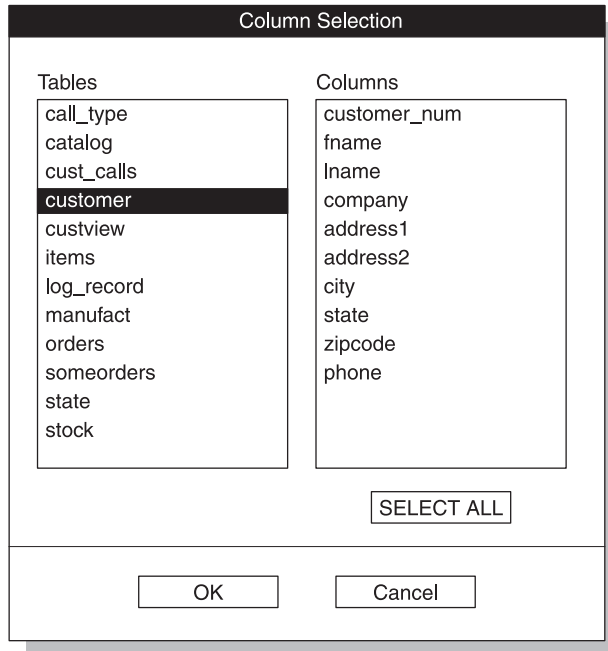


Figure 8-4. The Column Selection window after selecting a table

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.
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.

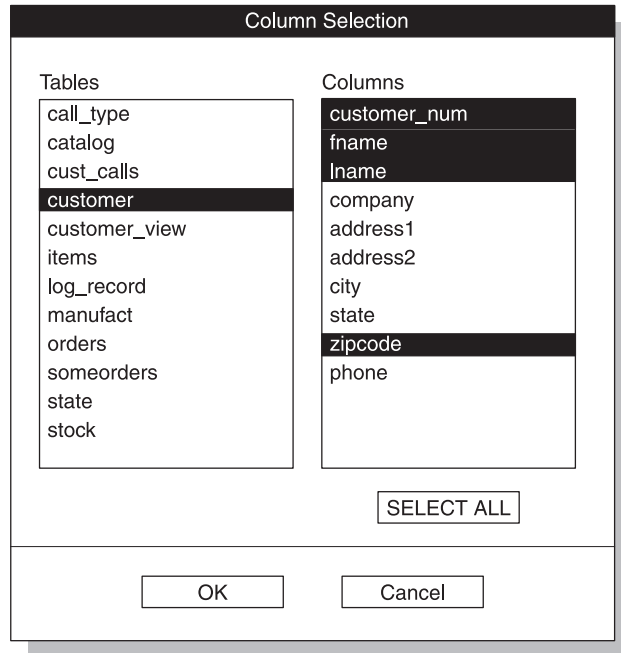


Figure 8-5. Columns selected from a table

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.

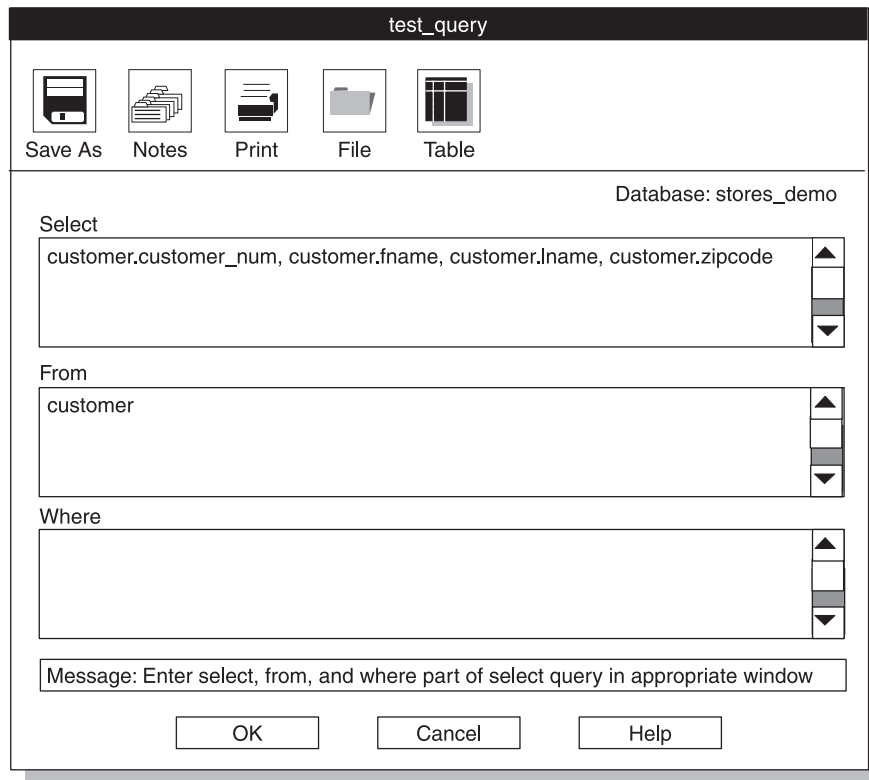


Figure 8-6. The Query-Definition window after using the table button

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 8-6.
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 8-4) 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.

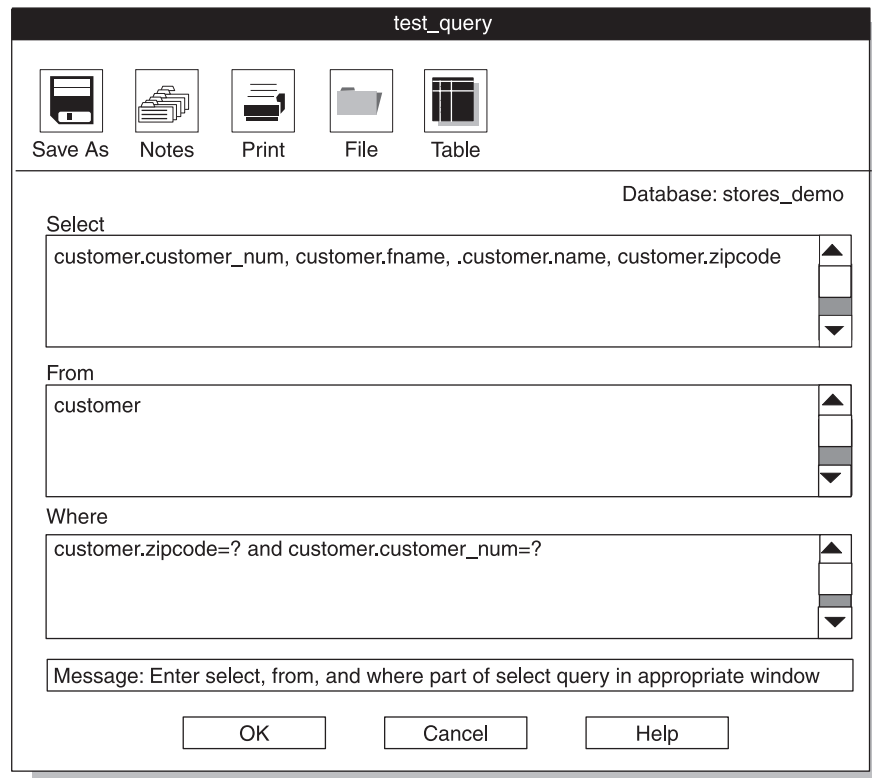


Figure 8-7. The *Where* text box entry after you use the *Table* button

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 8-7 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,” on page D-1.

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 8-2 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 8-3 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 8-3) by following the steps in “Creating a query” on page 8-1.
2. Click **File**.

The Import/Export File Selection window appears, as Figure 8-8 on page 8-8 shows.

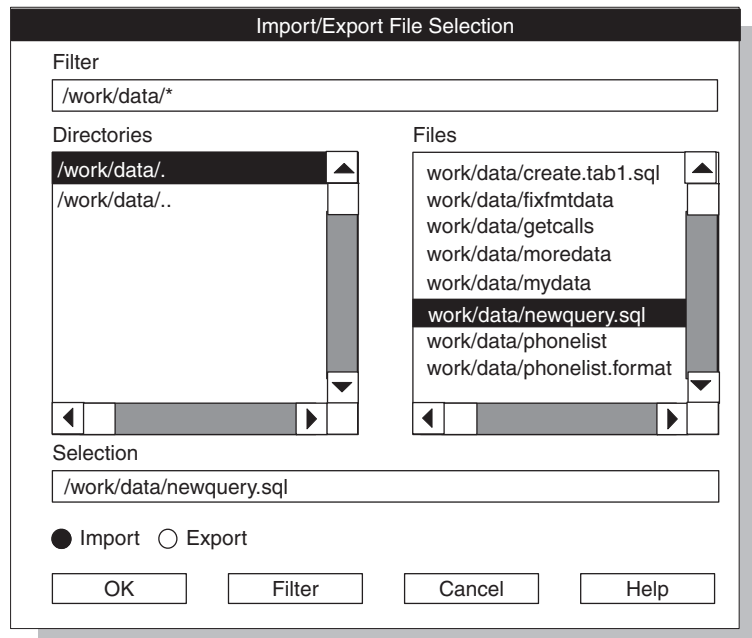


Figure 8-8. The Import/Export File Selection window

3. Click **Import**.
4. Specify the file that you want to import.
You can do this in either of the following ways:

- Type a path name and appropriate wildcards 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 path name 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 8-1.)
- Open an existing query.
- Import an already prepared query and modify it. (See “Importing a query” on page 8-8.)

To export a file:

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

2. Click **File**.

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

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 path name in the **Selection** text box and click **OK**.
- Type a path name 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 3-7 discusses how to use Views windows.

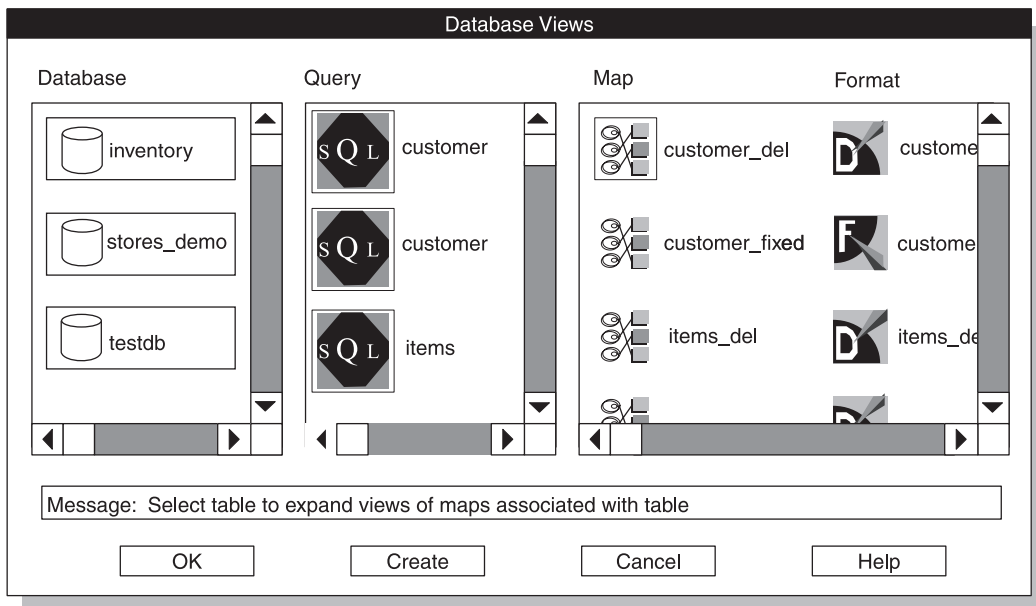


Figure 8-10. The Database Views window

Chapter 9. Defining maps

Load and unload maps	9-1
Using the map-definition window	9-2
Using the Table and Format panes	9-3
Using unassigned or multiple-assigned fields and columns	9-3
Using identical field names and column names	9-3
Creating a load map	9-3
Unload maps	9-5
Creating an unload map	9-5
Unloading data using functions	9-7
Mapping options	9-8
Using the mapping options	9-8
Setting the mapping options	9-9
Justification option.	9-9
Case Convert option	9-9
Default Value option.	9-10
Transfer Bytes option	9-10
Column Offset option	9-10
Field Offset option	9-10
Field Minimum and Field Maximum options	9-10
Fill Character option.	9-10
Picture option	9-10
Function option	9-10
Editing options	9-10
Using the Delete button.	9-11
Using the Find button	9-11
Using the Specs button	9-12
Map Views window	9-13

Load and unload maps

You can use the **ipload** utility 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.

The figure below shows the relationship between load and unload maps.

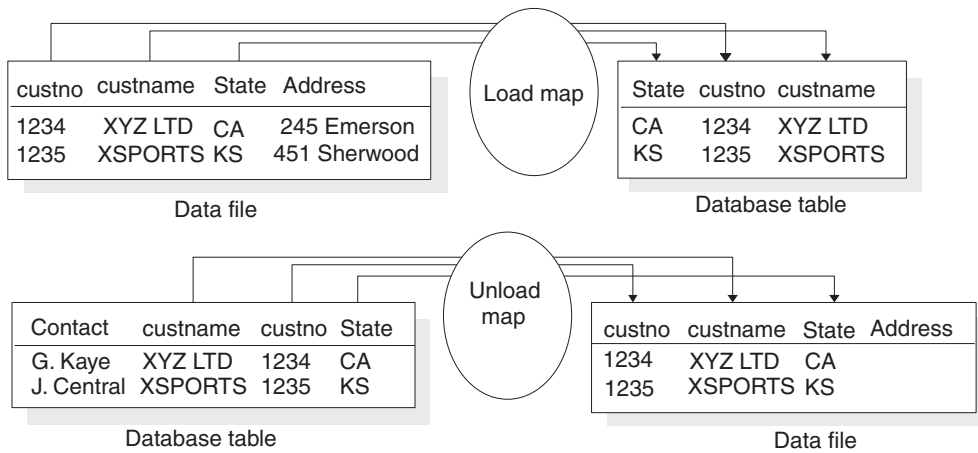


Figure 9-1. Using a map

The **ipload** utility stores information about maps in the **maps**, **mapitem**, **mapoption**, and **mapreplace** tables of the **onpload** database. Appendix A, “onpload database,” on page A-1 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.

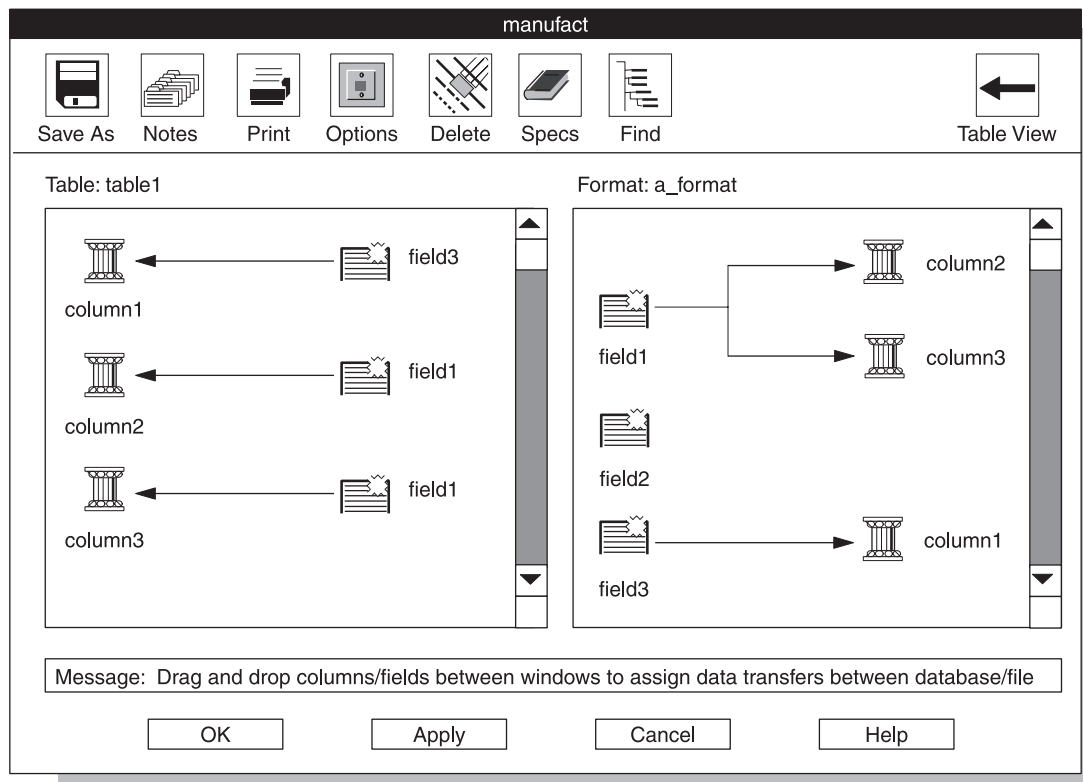


Figure 9-2. The map-definition window

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 9-7 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 9-5 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,” on page 7-1.

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 7-5 or “Data types allowed in a delimited format” on page 7-12

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.

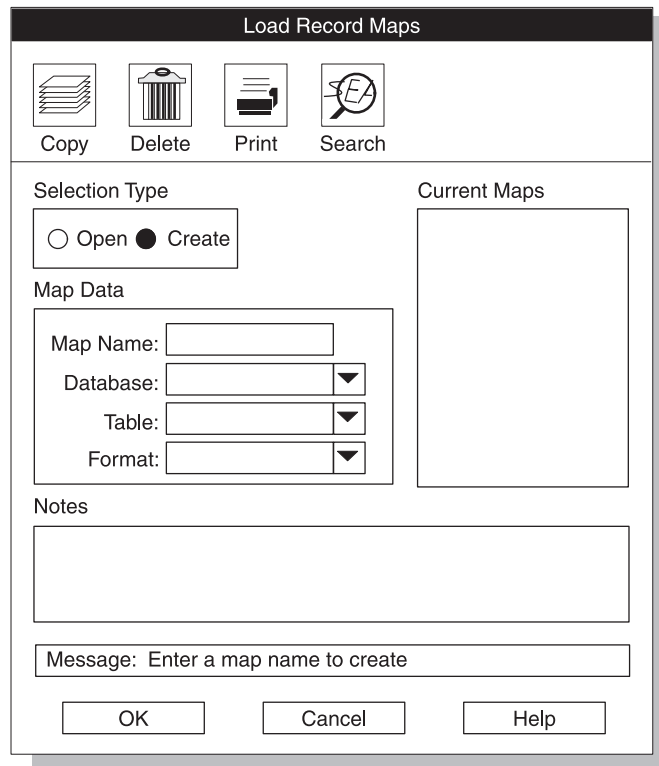


Figure 9-3. The Load Record Maps window

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.

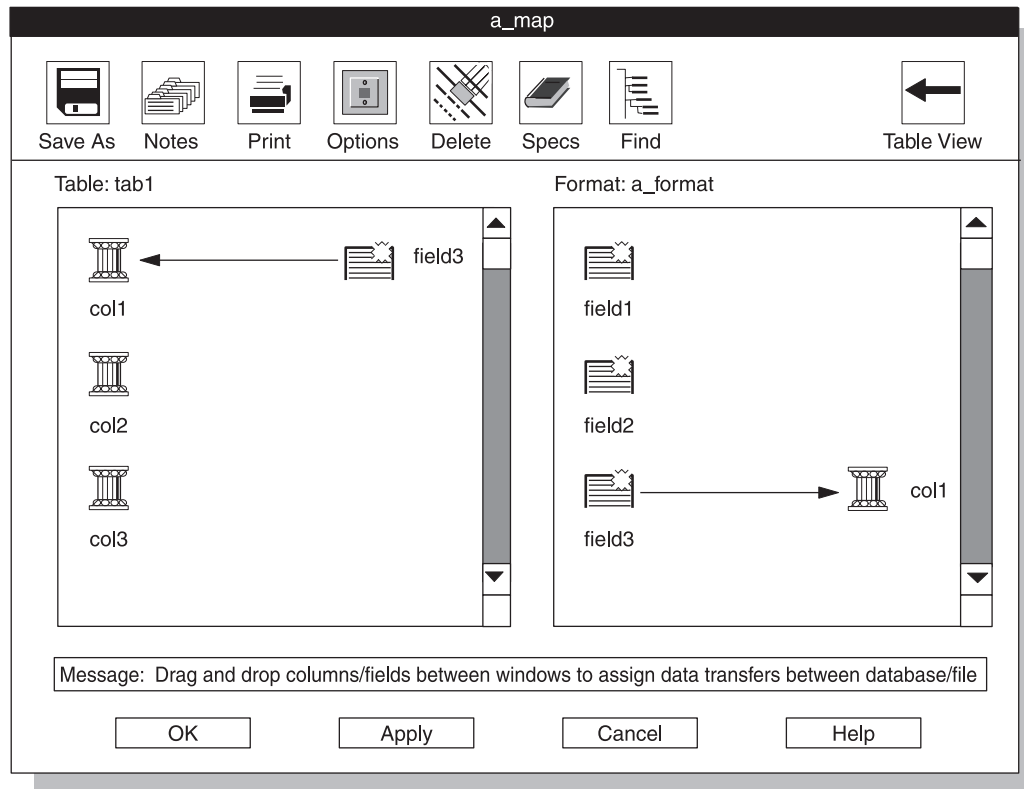


Figure 9-4. Map-definition window, one association completed

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 the mapping options” on page 9-8.
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 “HPL queries” on page 8-1.

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 7-5 or “Data types allowed in a delimited format” on page 7-12.

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.

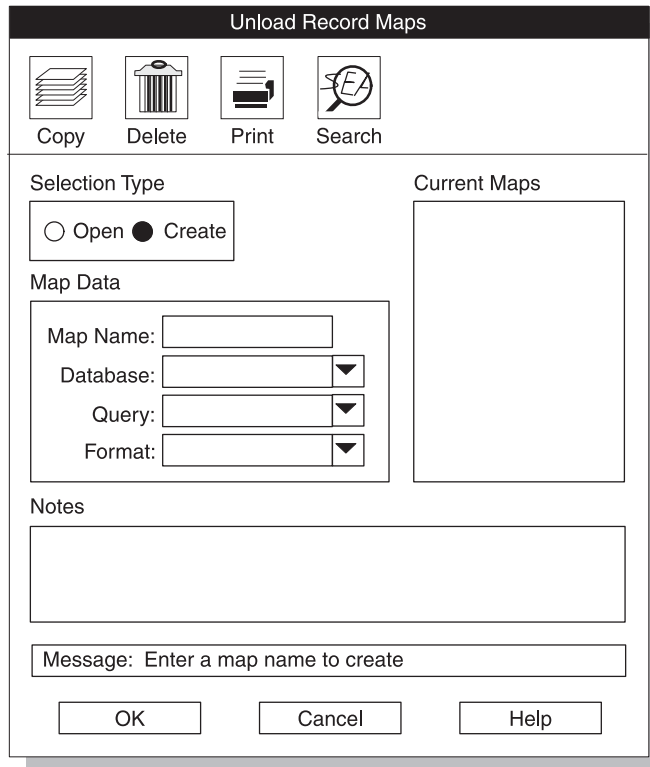


Figure 9-5. The Unload Record Maps window

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.

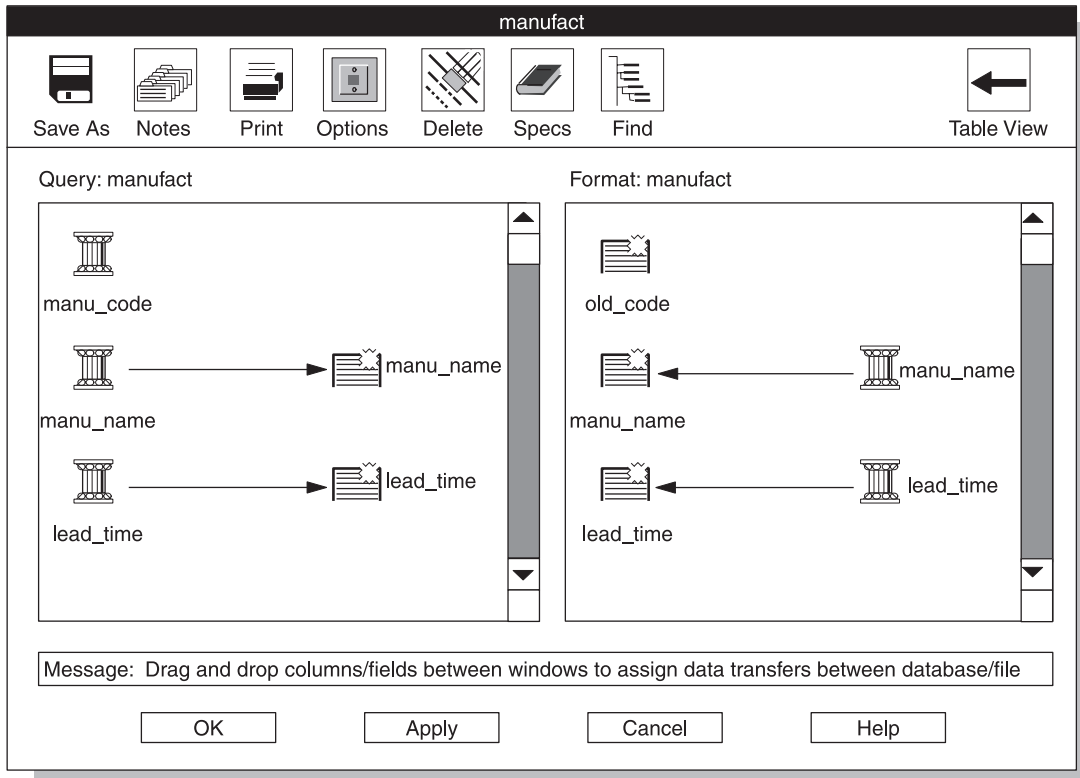


Figure 9-6. The map-definition window

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” on page 9-8.
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.

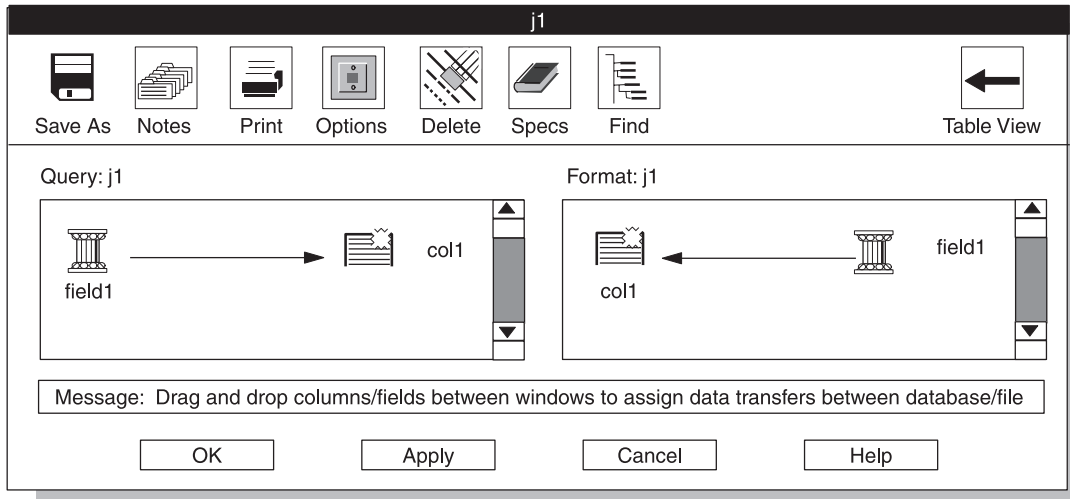


Figure 9-7. The map-definition window

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 Appendix A.)

Using the 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 9-3, or “Creating an unload map” on page 9-5.
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.

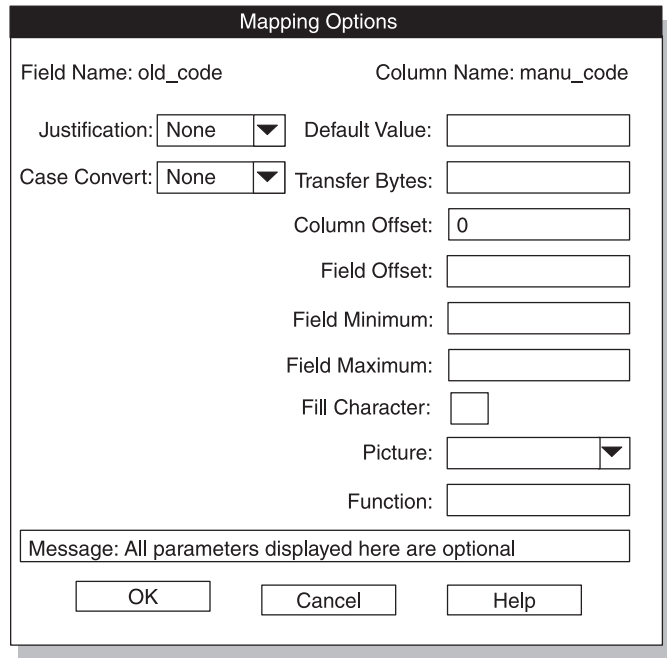


Figure 9-8. The Mapping Options window

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.

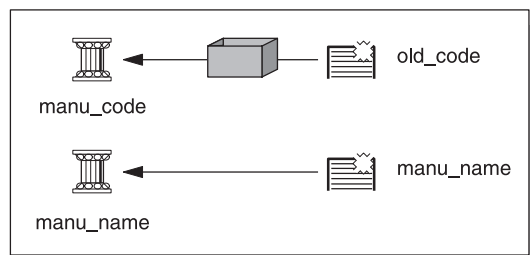


Figure 9-9. Fragment of the map-definition window showing an options symbol

Setting the mapping options

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

Justification option

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 option

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 option

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

Transfer Bytes option

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 option

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 option

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 options

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 option

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 option

The **Picture** option lets you reformat and mask data from the field of a record before the data is transferred to the database. Appendix C, "Picture strings," on page C-1, explains picture strings.

Function option

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, "Custom-conversion functions," on page E-1.

Editing options

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

Using the Delete button

The **Delete** button in the map-definition window 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 in the map-definition window 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 9-11 shows the upper portion of the map-definition window after you select the **Format** pane.

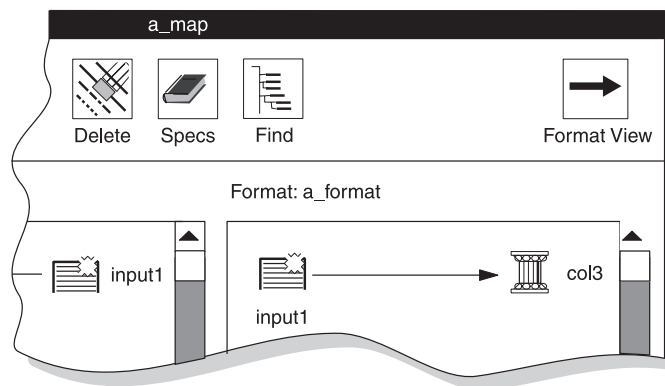


Figure 9-10. The view indicator

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.

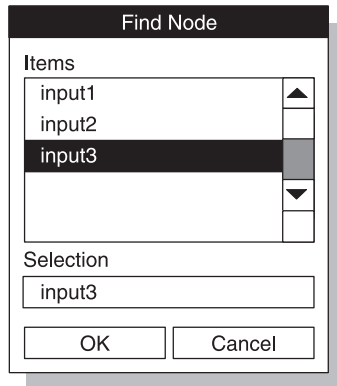


Figure 9-11. The Find Node window

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.

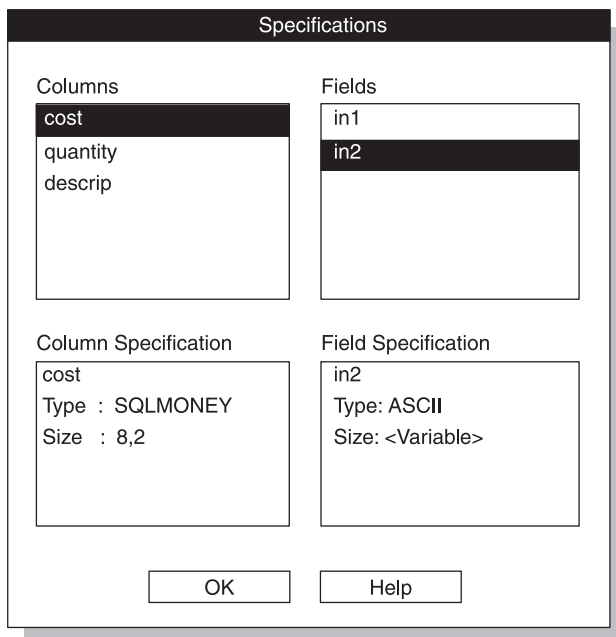


Figure 9-12. The Specifications window

To use the Specifications window:

1. Click **Specs** in the map-definition window.
The Specifications window appears as Figure 9-12 on page 9-12 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 7-18.) 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.

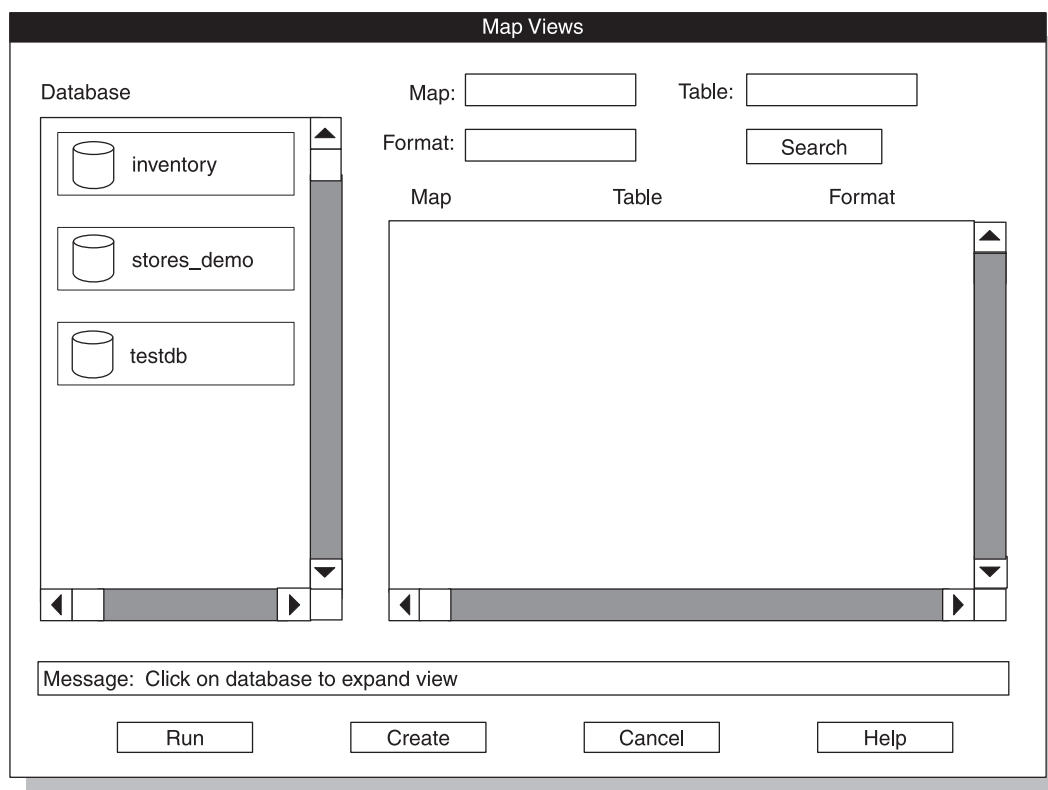


Figure 9-13. The Map Views window for a load map

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 **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.

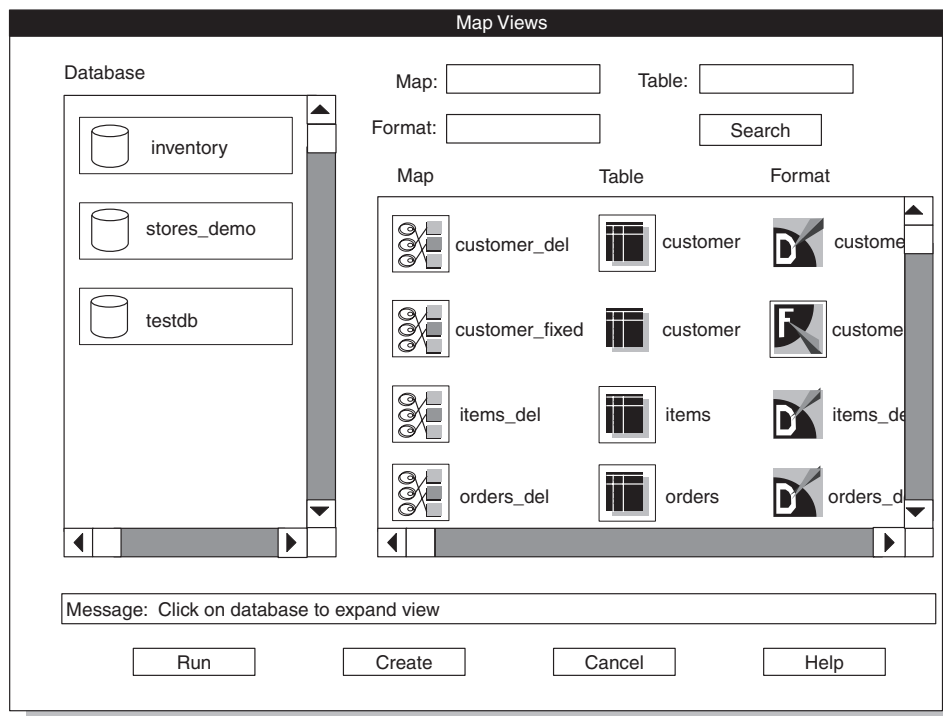


Figure 9-14. The Map Views window with the view expanded

To see selected load maps:

1. Open the Map Views window (Figure 9-13 on page 9-13).
2. Select a database.
3. Type the name or partial name of a map, table, 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.

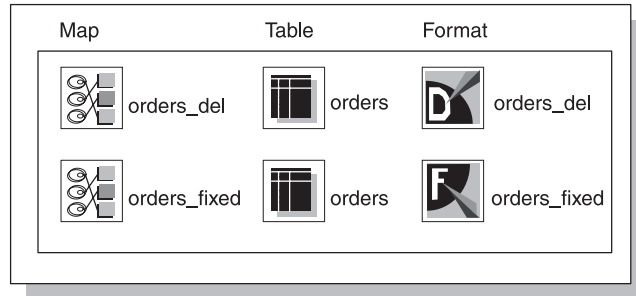


Figure 9-15. The maps that a search found

Chapter 10. Defining filters

Filters	10-1
Using a filter	10-1
Creating a filter	10-2
Editing a filter	10-5
Filter views	10-6
Filters with code-set conversion (GLS)	10-7

Filters

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 12-5.

Filters have the following restrictions:

- You cannot use filters with Ext Type data types.
- The DATE and DATETIME data filters 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," on page 7-1.

The **ipload** utility stores the filter information in the **filters** table of the **onpload** database. For more information about the **filters** table, see A-3.

To create a filter:

1. Choose **Components > Filter** from the HPL main window.
The Filters window appears, as Figure 10-1 shows.

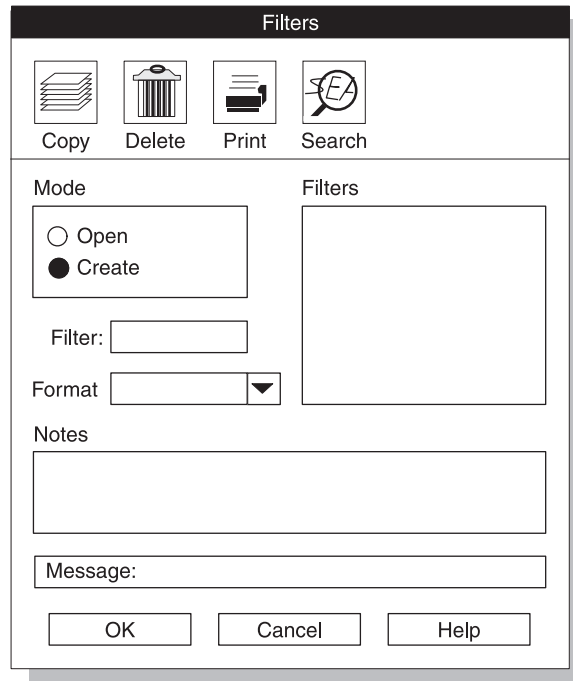


Figure 10-1. The Filters window

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.

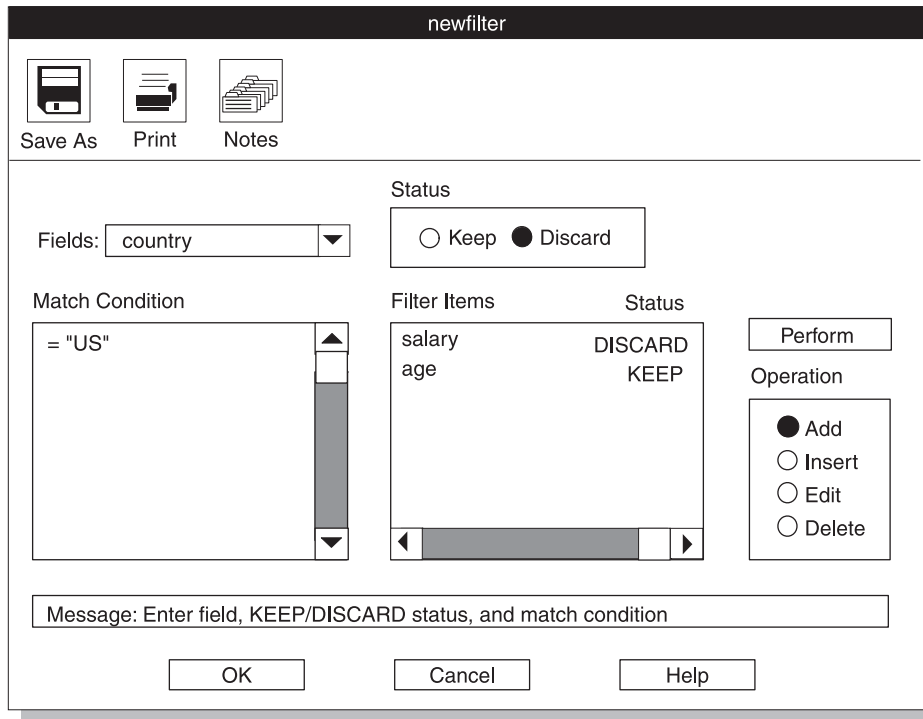


Figure 10-2. The Filter-Definition window

The 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 10-3 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 10-4 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 3-7 discusses the use of Views windows.

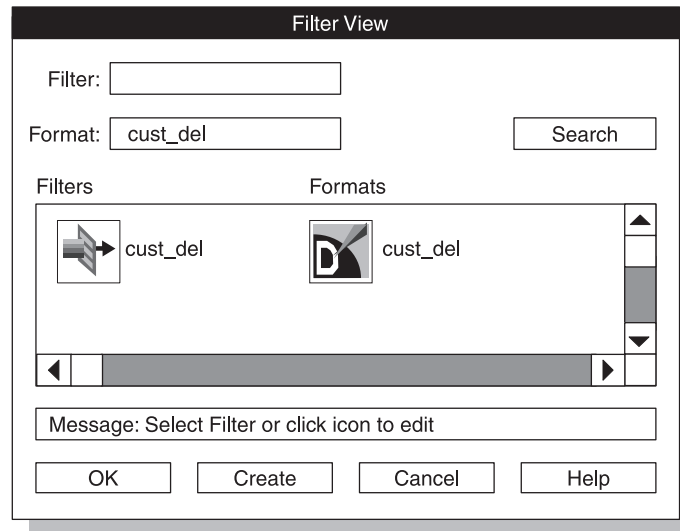


Figure 10-3. The Filter Views window

Filters with code-set conversion (GLS)

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*.

Chapter 11. Unloading data from a database

Unload jobs.	11-1
Components of the unload job	11-1
Choosing the database server.	11-2
Running multiple jobs	11-2
Unload Job windows	11-2
Creating an unload job	11-2
Running the unload job.	11-5
Problems during an unload job	11-5
Specifying to write to the end of the tape.	11-6
Using the command-line information	11-6
Changing the unload options	11-7
Editing an unload job	11-7
Generate options for an unload job	11-8

Unload jobs

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 run an unload job from the Unload Job window of **ipload**, or you can run 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,” on page 16-1.

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,” on page 6-1.
- The format that describes the organization of the data file into which you are unloading data
See Chapter 7, “Defining formats,” on page 7-1.
- The query that extracts the desired records from the database
See Chapter 8, “Defining queries,” on page 8-1.
- 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,” on page 9-1.

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, “The Generate options of the ipload utility,” on page 13-1.

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, "onpload database," on page A-1.

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.

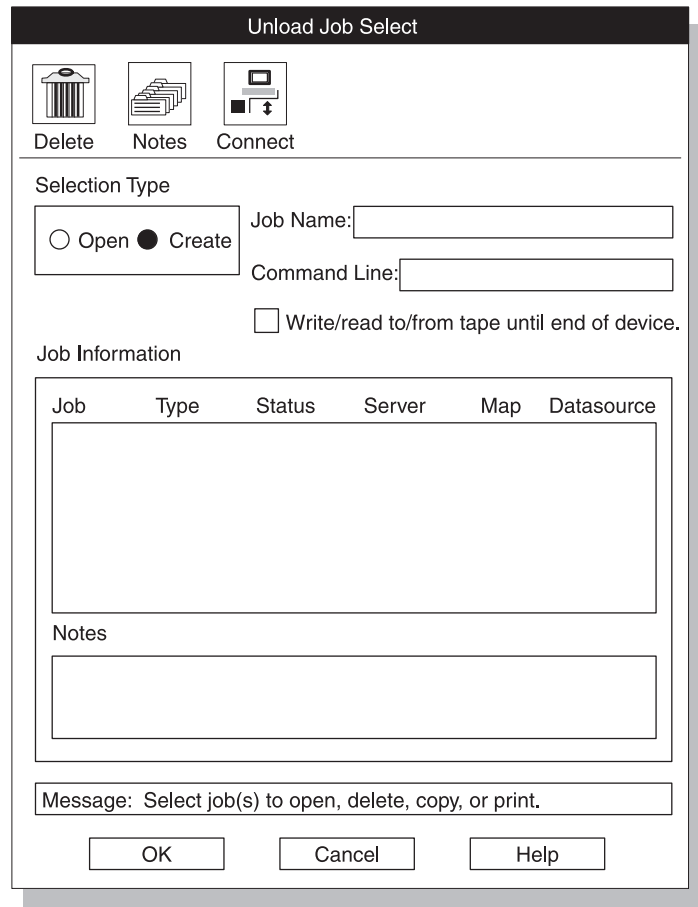


Figure 11-1. The Unload Job Select window

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 11-6.
5. Click **OK**.

The Unload Job window appears, as Figure 11-2 shows. Chapter 3, “Using the ipload utility windows,” on page 3-1, provides detailed descriptions of the buttons in the Unload Job window.

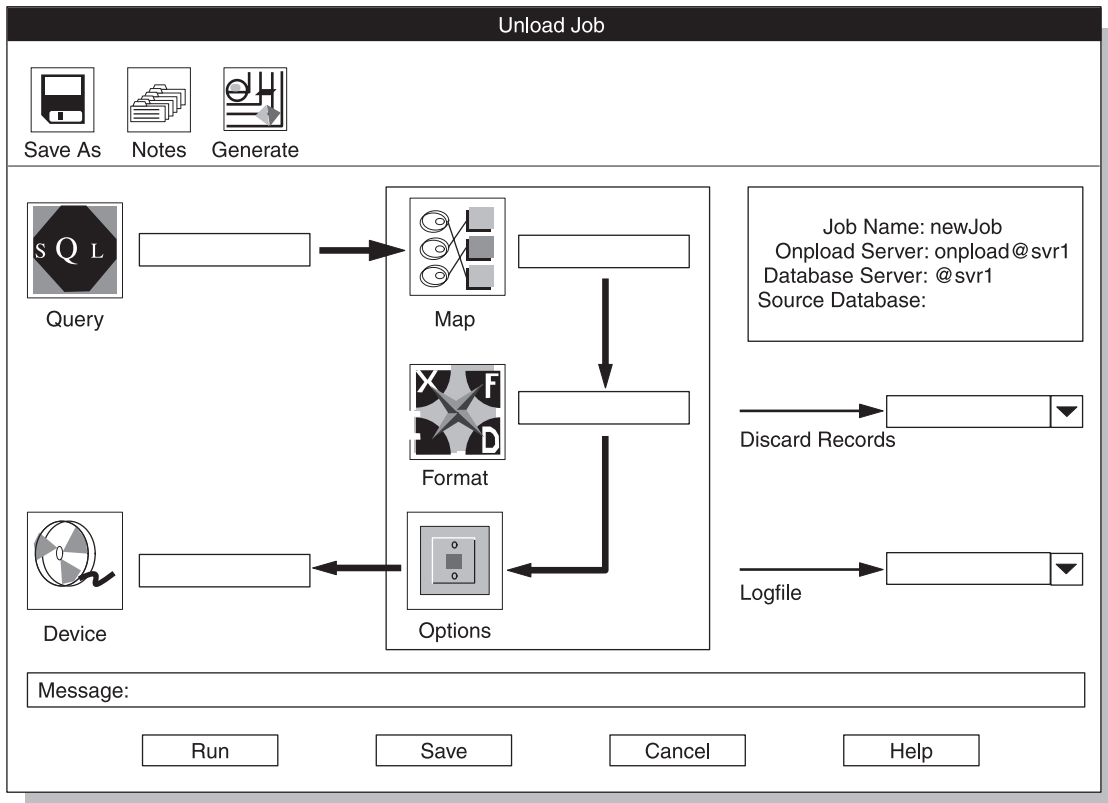


Figure 11-2. The Unload Job window

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 11-7.
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 14-5.

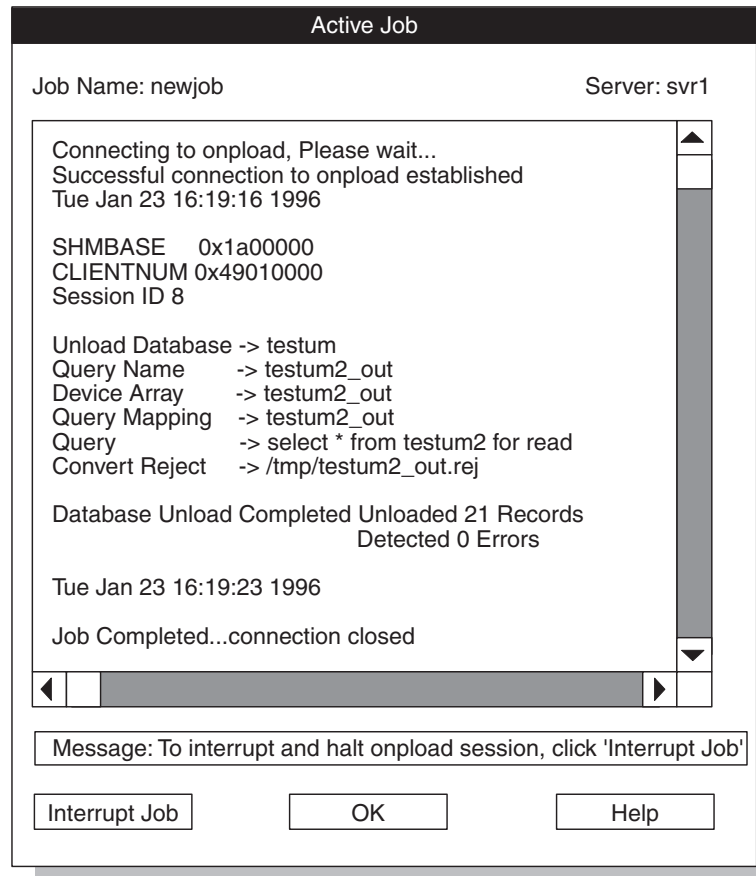


Figure 11-3. The Active Job window

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, “HPL browsing options,” on page 14-1.

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** check box. 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 check box, 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 unload jobs and 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 11-5.

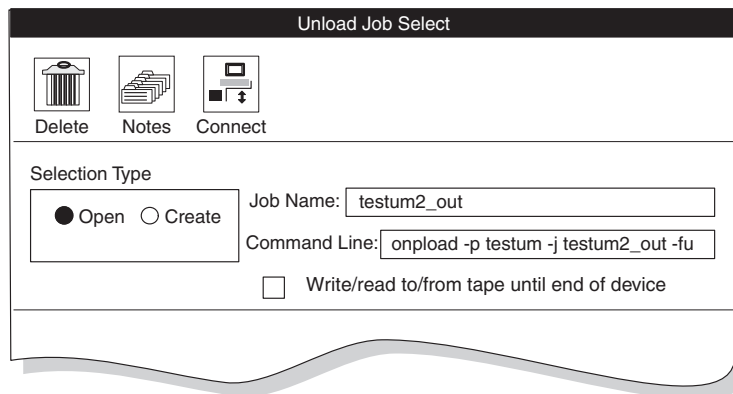


Figure 11-4. Fragment of the Unload Job Select window

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 run 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 11-2.
2. Click **Options**.
The Unload Options window appears, as Figure 11-5 shows.

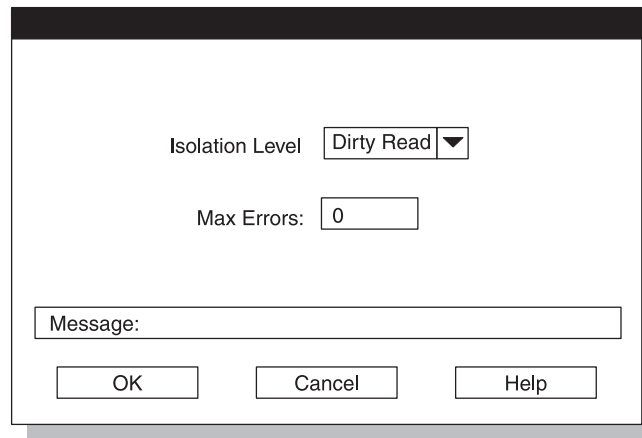


Figure 11-5. The Unload Options window

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 11-4 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 11-7.
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 for an unload job

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, “The Generate options of the ipload utility,” on page 13-1, 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.

Chapter 12. Loading data to a database table

Load jobs	12-1
Components of the load job	12-1
Choosing the database server.	12-1
Running multiple jobs	12-2
Preparing user privileges and the violations table	12-2
Setting user constraints	12-2
Managing the violations and diagnostics tables	12-2
Load Job windows	12-3
Creating a load job	12-4
Running the load job	12-6
Making a level-0 backup	12-6
Problems during a load job	12-6
Specifying to read to the end of the tape	12-6
Using the command-line information	12-7
Changing the load options.	12-8
Editing a load job.	12-9
Generate options for a load job	12-9

Load jobs

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, although 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.

UNIX Only

If you are using a UNIX **cron** job to run the load or unload jobs, let one job finish before you start the next.

End of UNIX Only

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 12-8.

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 they are 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 does 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 window (Figure 12-1 on page 12-4) and the Load Job window (Figure 12-2 on page 12-5) let you create, save, and run 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 run it immediately.

The **ipload** utility assigns path names 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 (A-10) 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,” on page A-1.

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

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 12-6.
5. Click **OK**.

The Load Job window appears, as Figure 12-2 shows.

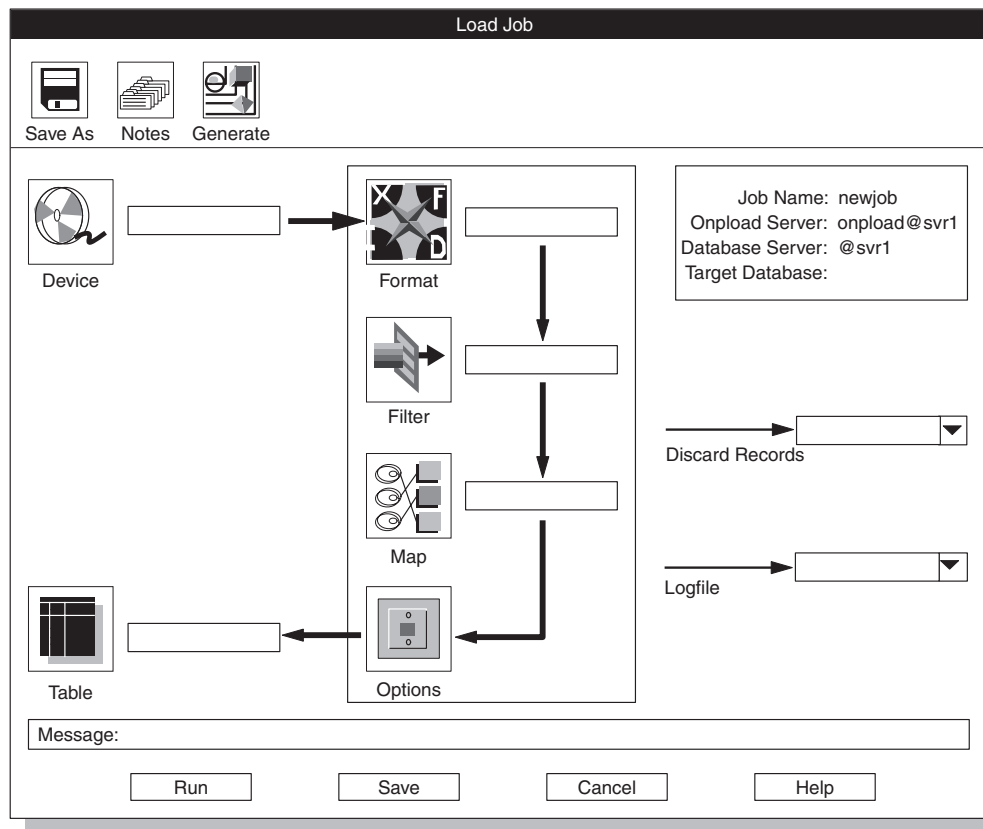


Figure 12-2. The Load Job window

6. Type the appropriate values for the components of the load.

“HPL ipload utility icon buttons” on page 3-16 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 14-3 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 14-5.
9. Click **Options** to change the load options.

For more information on these options, see “Changing the load options” on page 12-8.
10. Click **Save** to save this load 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 2-20 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 14-1.

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 *IBM Informix Administrator's Guide*. For information about making backups, see the *IBM Informix Backup and Restore Guide*.

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, “HPL browsing options,” on page 14-1.

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** check box. 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 check box, 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 jobs 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 2-18 shows.

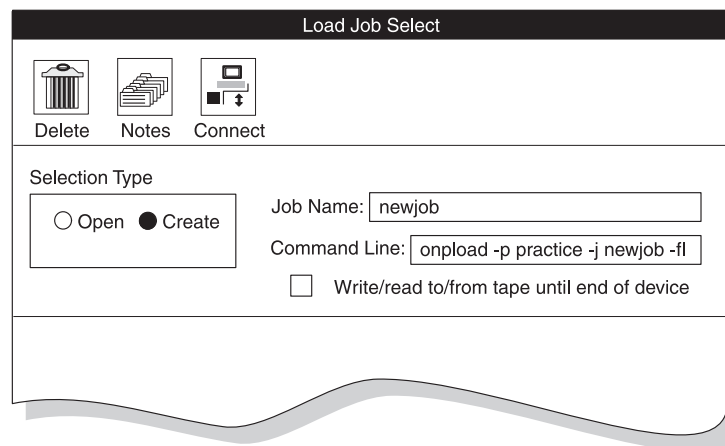


Figure 12-3. Fragment of the Load Job Select window

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 run 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, “onpload database,” on page A-1.

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 ViolationsRecords	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 12-4.
2. Click **Options**.
The Load Options window appears, as Figure 12-4 shows.

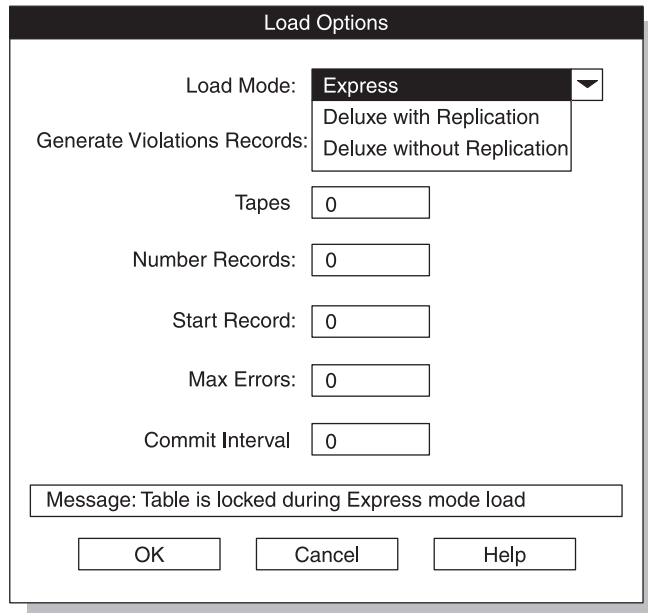


Figure 12-4. The Load Options window

3. Change the desired options.
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 12-4 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 12-5 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 for a load job

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, “The Generate options of the iupload utility,” on page 13-1, 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.

Chapter 13. The Generate options of the ipload utility

Overview of the ipload Generate options	13-1
Tasks that generate load or unload components.	13-1
Generating from the Load Job window	13-1
Using the Autogenerate Load Components window	13-2
Generating from the Unload Job window	13-3
Using the Autogenerate Unload Components window	13-3
Generating from the Components menu	13-5
Generate window.	13-5
Generate group	13-6
Format type group	13-6
Generating load and unload components	13-7
Using the No Conversion Job option	13-8

Overview of the ipload Generate options

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.

Tasks that generate load or unload components

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 12-4 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 12-5 shows.
5. Click **Generate**.
The Autogenerate Load Components window appears, as Figure 13-1 shows.

The screenshot shows a dialog box titled "Autogenerate Load Components". It is divided into two main sections: "Load From" and "Load To".

- Load From:** Contains two radio buttons. The "Device Array" radio button is selected. To the right of the radio buttons is a text input field followed by a downward-pointing arrow.
- Load To:** Contains two dropdown menus. The first is labeled "Database:" and the second is labeled "Table:". Both have downward-pointing arrows.
- Message:** A text box containing the message "Message: Enter file name to load from".
- Buttons:** At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

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 path name 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 path names in the **Discard Records** and **Logfile** text boxes.
11. Click **Save** to save the components and the job.
12. Click **Run** to run 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 job 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 8-1 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 11-3 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 11-4 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 /work/cust_out

File

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.

- 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 job, you must first create the file.

- Click **OK** to generate the components of the unload job.

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.

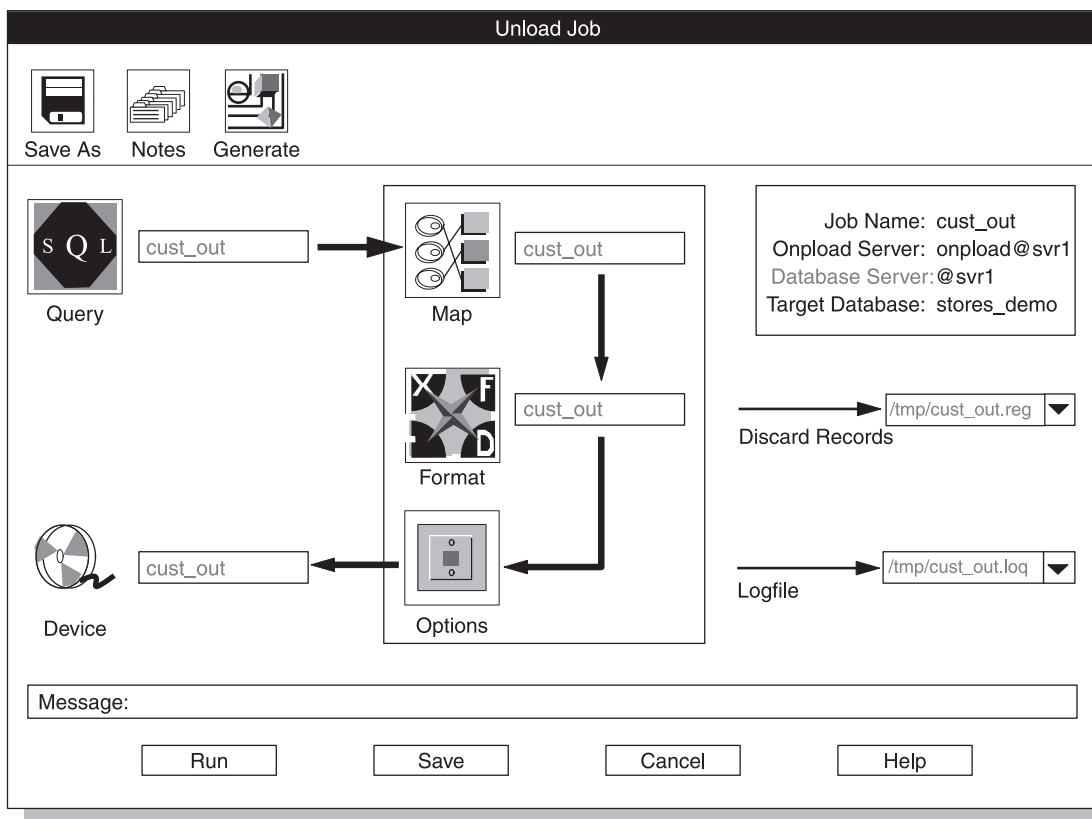


Figure 13-3. The Unload Job window

- 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.
- To run the job, click **Run**.
The Active Job window appears, as Figure 2-17 on page 2-20 shows.
- 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:

- Choose **Jobs > Unload** from the HPL main window.
The Unload Job Select window appears, as Figure 11-1 on page 11-3 shows.
- 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 11-4 shows.
5. Click **Generate**.
The Autogenerate Unload Components window appears, as Figure 13-2 on page 13-3 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 job.
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 2-20 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.

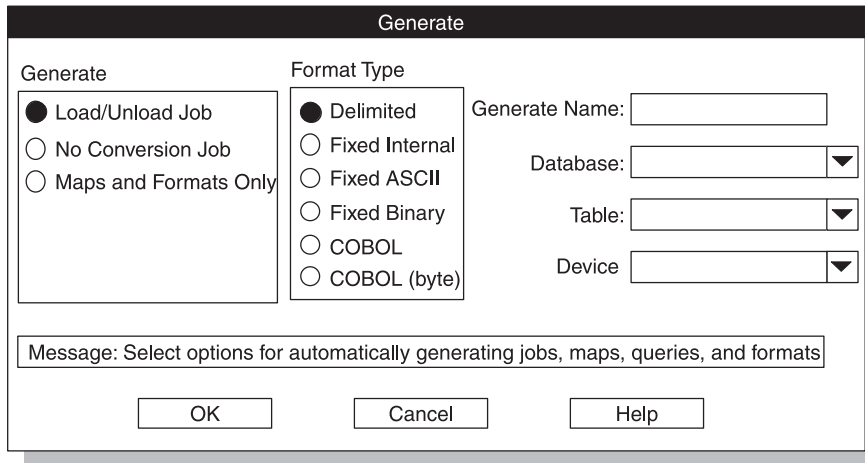


Figure 13-4. The Generate window

Generate group

The **Generate** group specifies the type of generate to perform. The **Generate** group has the following choices.

Choice	Effect	See
Load/Unload Job	Generates both load and unload jobs	13-7
No Conversion Job	Generates a job that treats an entire database record as one entity	13-8
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
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.	7-19
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.	7-18
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.	7-2

Choice	Description	See
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.	7-2
COBOL	The data file is formatted according to COBOL 86 standards. All COBOL data types are supported.	7-15
COBOL (byte)	The data file is formatted using byte alignments for COMP-4 data type. All COBOL data types are supported.	

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 7-18.

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 13-6 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.

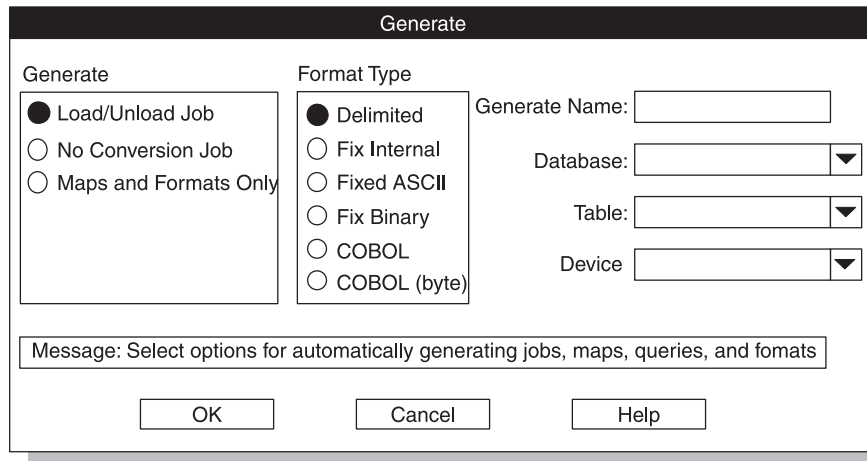


Figure 13-5. The Generate window

For information on generating a job from a **.csv** file, see “Using a CSV file as **ipload** utility input” on page 7-20.

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.



Fast Job Startup

Job Name: a_fast_job

Database: stores_demo ▼

Table: customer ▼

Device Array some_array ▼

Message: Enter database to unload

Run Cancel Help

Figure 13-6. The Fast Job Startup window for a load job

9. Click **Run** to run the job.
The Active Job window appears, as Figure 2-17 on page 2-20 shows.
10. When the Active Job window displays Job Completed, click **Cancel** to return to the main HPL window.

Chapter 14. HPL browsing options

Browsing options	14-1
Previewing data-file records	14-1
Using the Record Browser window	14-1
Reviewing records that the conversion rejected	14-3
Viewing the violations table	14-4
Viewing the status of a load job or unload job	14-5
Viewing the log file	14-5
Sample log file.	14-6

Browsing options

The browsing options of the HPL let you preview records from the data file, review records after you perform a load or unload job, review various files associated with the HPL, and view the status of a load or unload job.

Previewing data-file records

Before you actually run 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 7-6).

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.

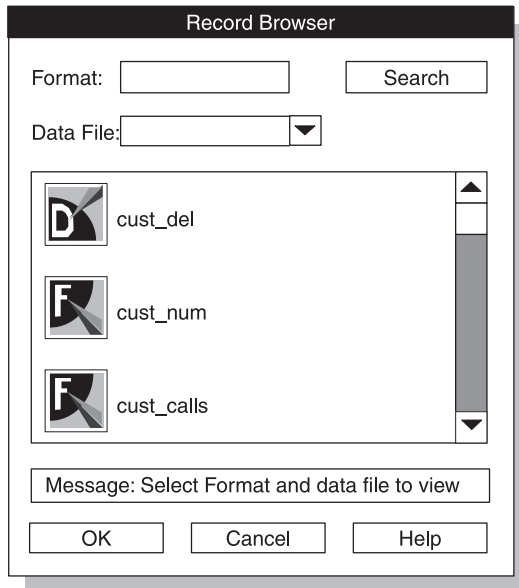


Figure 14-1. The Record Browser window

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.

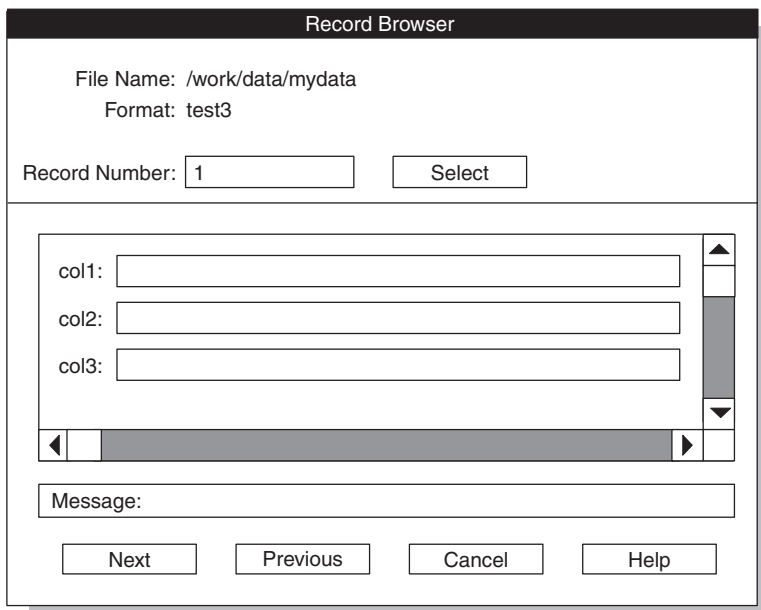


Figure 14-2. The Record Browser window

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 14-2 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 7-6.

Reviewing records that the conversion rejected

When you run 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 12-4. 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 14-2 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 14-2 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.

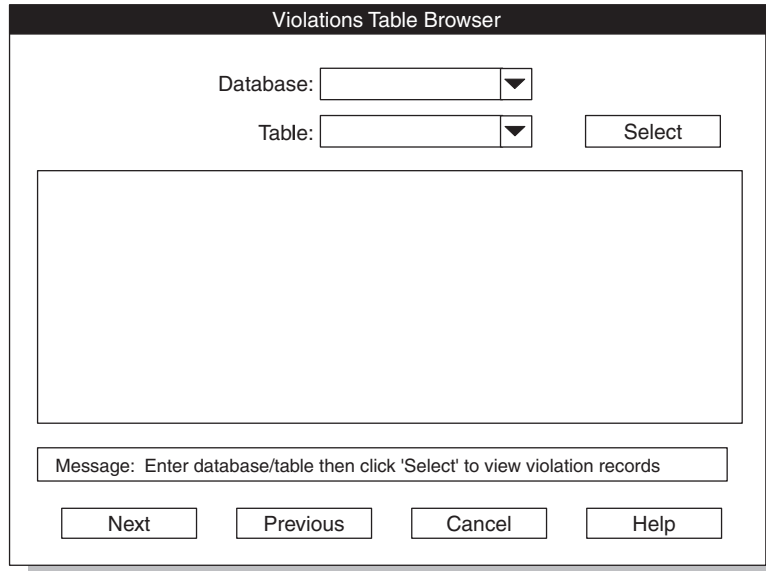


Figure 14-3. The Violations Table Browser window

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.

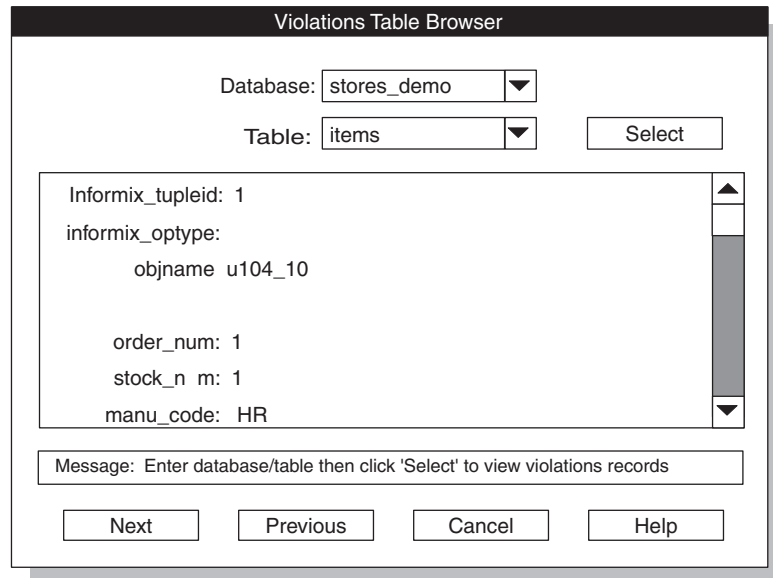


Figure 14-4. A Record in the Violations Table Browser window

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, “HPL log-file and pop-up messages,” on page G-1.

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 12-5) or Unload Job Window (Figure 11-2 on page 11-4).

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.

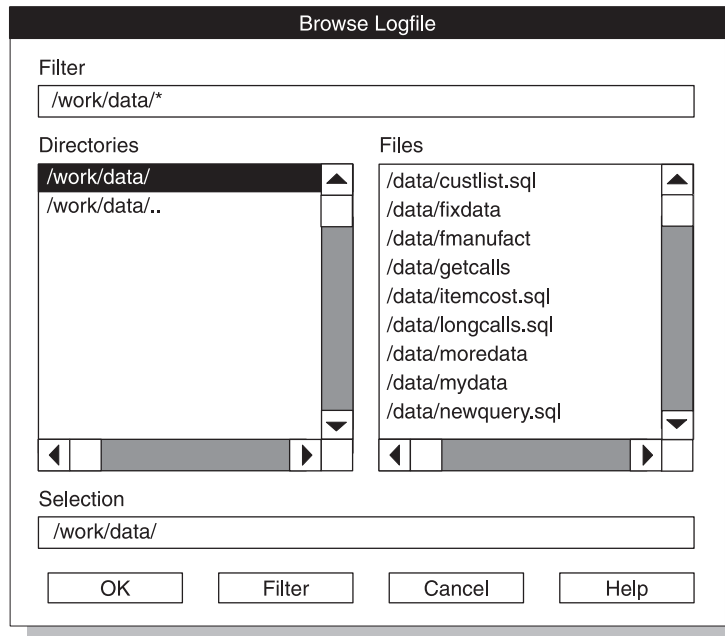


Figure 14-5. The Browse Logfile window

2. In the **Filter** text box, type the full path name 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 path name 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 path name of the log file, you can simply type the path name 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.

Chapter 15. Managing the High-Performance Loader

Manage modes, errors, and performance	15-1
HPL modes	15-2
The HPL deluxe mode	15-2
The HPL express mode	15-2
How the express and deluxe modes work	15-3
Foreign-key constraints	15-4
Comparison between an express-mode and a deluxe-mode load operation.	15-4
HPL load and unload errors	15-5
Rejected records from the input file.	15-5
Constraint violations.	15-5
Viewing error records	15-6
HPL performance.	15-6
onpload configuration parameters	15-6
Express-mode limitations	15-7
onstat options for onpload.	15-7
Devices for the device array	15-8
HPL usage tasks	15-8
Reorganizing computer configuration	15-8
Altering the schema of a table	15-8
Assessing information for loading or unloading external data	15-8
Settings for a no-conversion load or unload job.	15-9
Running no-conversion load jobs with tables with hidden columns	15-9
An express-mode load with delimited ASCII	15-10
HPL performance hints	15-10
Choose an efficient format	15-10
Ensure enough converter threads and VPs	15-11
Ensure enough memory	15-11
Ensure enough buffers of adequate size	15-11
Increase the commit interval.	15-12
Limitation when using the Excalibur Text DataBlade Module indexes	15-12

Manage modes, errors, and performance

When you manage the HPL, you manage:

- Modes
- Violations (errors)
- Performance

You also need to avoid the limitation that occurs when using an Excalibur Text DataBlade® module index

HPL 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.

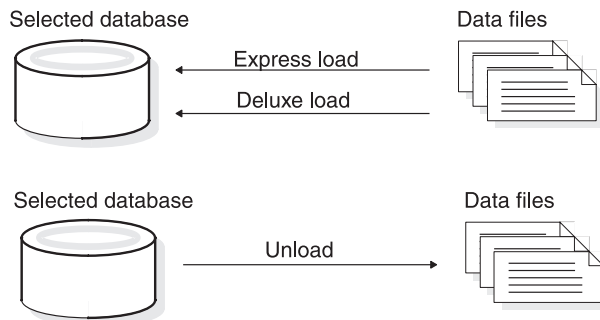


Figure 15-1. The load and unload modes of the HPL

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.

The HPL 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
- Re-enables all objects after loading, if possible, and flags objects that cannot be re-enabled in the violations and diagnostic tables
- Supports loading of raw tables

The express mode has the following limitations:

- Does not support:
 - Logging mode or ANSI mode
 - Tables that contain smart large objects (BLOB or CLOB data types)
 - 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 *IBM Informix 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 the express and deluxe modes work

This topic describes how HPL implements the deluxe and express modes. You do not need to understand this discussion to use the HPL.

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*.

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 12-6.
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, re-enable 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**.

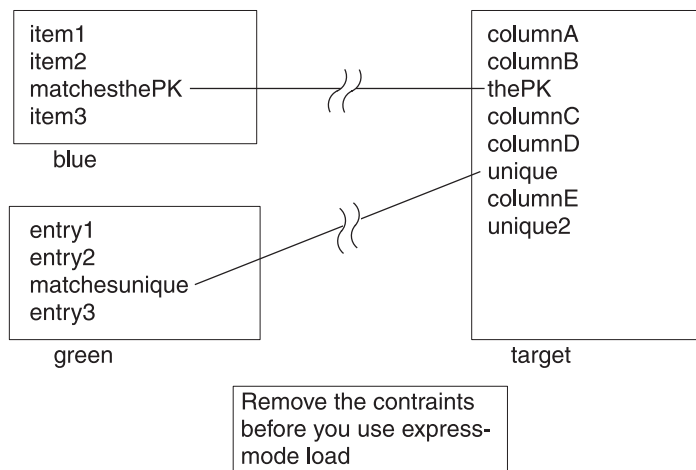


Figure 15-2. Foreign-key constraints

Comparison between an express-mode and a deluxe-mode load operation

The following table contrasts the operation of express deluxe modes. For additional information on the differences between these two modes, see “The HPL deluxe mode” on page 15-2 and “The HPL express mode” on page 15-2.

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
Complete a level-0 backup to make the database writable	user		
Resolve violators	user	Resolve violators	user

HPL load and unload errors

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 14-1.

HPL 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

onpload 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.

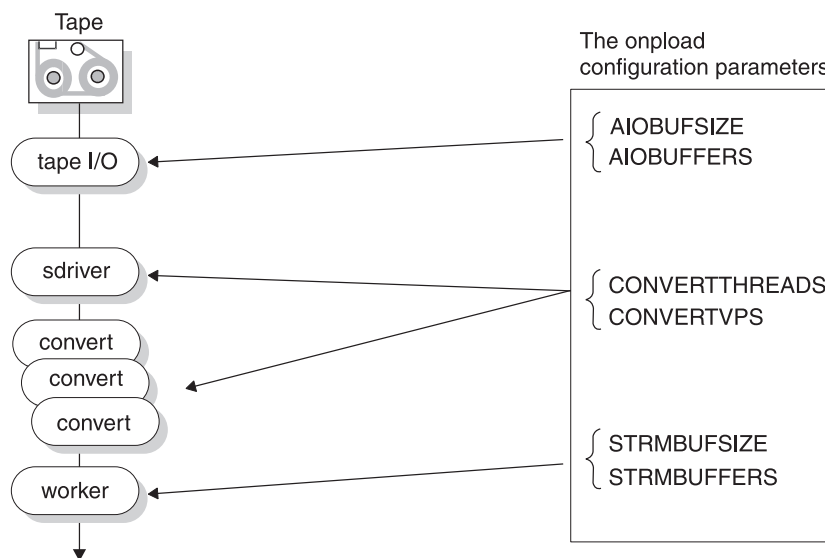


Figure 15-3. The onpload configuration parameters

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:

UNIX Only
• \$INFORMIXDIR/etc/\$PLCONFIG
End of UNIX Only

Windows Only
• %INFORMIXDIR%\etc\%PLCONFIG
End of Windows Only

Appendix B, “High-Performance Loader configuration file,” on page B-1 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 “HPL usage tasks” on page 15-8 and “HPL performance hints” on page 15-10.

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 “The HPL express mode” on page 15-2. For more information on characteristics of the express mode and the deluxe modes, see “HPL modes” on page 15-2.

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, “onstat -j option,” on page F-1 of this publication.

The **onstat** utility is documented in your *IBM Informix 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.

HPL usage tasks

Three tasks that you might need to perform for the HPL are:

- Reorganizing computer configuration
- Altering the schema of a table
- Assessing information for 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 job or a load job, 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 5-6.

For information about preparing for a no-conversion unload/load with **ipload**, see “Using the No Conversion Job option” on page 13-8. 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,” on page 16-1.

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 13-6.

Assessing information for loading or unloading external 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 job

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 13-8.

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

Running no-conversion load jobs with tables with hidden columns

If you run no-conversion jobs, the physical image of the table in the load file must match the physical image of the target table and all columns must exist in the same order and be of the same type.

By default a no-conversion unload job does not contain the hidden columns, because the job resulted from unloading a "SELECT *" query and this type of SELECT query does not return any hidden columns. Under these circumstances, the target table where the load will occur must not contain any hidden columns. If it does contain hidden columns, the image of the file will not be a match.

If your application requires that the unload file contains the hidden columns (for example, if the target table has hidden columns and it is not possible to alter the table to remove the columns while the load occurs) , you must use a SELECT statement that explicitly indicates the columns in the select list.

The target table should not contain hidden columns, because these columns retain internal information for internal activities of the engine. Loading hidden columns can create unexpected results.

Hidden columns tables include but are not restricted to fragmented tables with rowid columns, tables with VERCOLS, and tables with CRCOLS. Also fragmented tables with rowid columns do not support no-conversion jobs.

An 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
CONVERTTHREADS	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	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

HPL 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 highly 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 **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 or if the **PDQPRIORITY** 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
STRMBUFFSIZE	64	Should be some multiple of AIOBUFSIZE for best performance
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 Figure 12-4 on page 12-9). 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.

Chapter 16. The onpload utility

Overview of the onpload utility	16-1
onpload filename size limitations on UNIX	16-1
Starting the onpload utility	16-2
Using the onpload utility	16-2
The onpload utility syntax.	16-2
Setting the onpload run mode with the -f option	16-4
Typing the onpload -f flags	16-5
Interpreting the onpload -d and -f options together	16-6
Modifying the size of onpload database parameters	16-6
Using the onpload -i option	16-9
Overriding the onpload database values	16-9
Loading data into collection data type columns	16-12

Overview of 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. However, you cannot load or unload binary large objects (BLOBs) or character large objects (CLOBs) to or from multiple files.

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.

onpload filename size limitations on UNIX

On UNIX you cannot have a filename size greater than 256 characters.

Suppose you have a 128-character database name and a 128-character table name and automatically generate the project (auto job generation), the generated device filenames have the following format:

directory_specified / databasename_ tablename.unl

In this situation, the device name would exceed 256 characters and the job would not run. To work around this problem, modify the device filenames (assuming the specification filename is **device.hpl**):

1. Describe the device with the following command:

```
onpladm describe device devicename -F device.hpl
```

For more information on **onpladm**, see Chapter 17, “The onpladm utility.”

2. Modify the **device.hpl** specification file to shorten the names of **.unl** files.
3. Modify the device attributes with the following command:

```
onpladm modify device devicename -F device.hpl
```

Alternatively, do not use the auto generation option of **onpladm** when you have both database and table names that would produce a generated name exceeding the 256 UNIX filename limit. Instead, manually provide a device filename.

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 12-5) or in the Unload Job window (Figure 11-2 on page 11-4), **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 (12-5) or Unload Job window (11-4) to prepare the load or unload job.

After you prepare the job, the **Command Line** text box on the Load Job Select window (12-4) or the Unload Job Select window (11-3) 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.

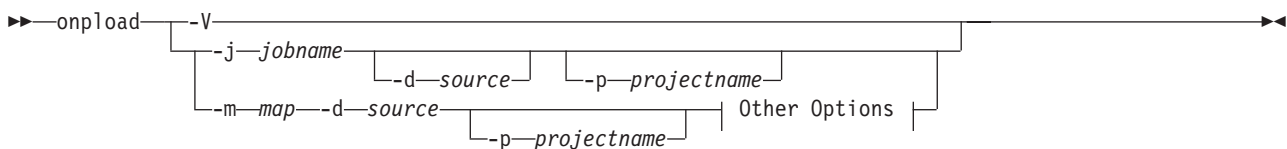
Tip: Enter **onpload** with no options at the command line to display a command-line listing of all **onpload** options and their functions.

When you use the **onpload** utility, you cannot load or unload smart large objects (BLOB or CLOB data types) to or from multiple files.

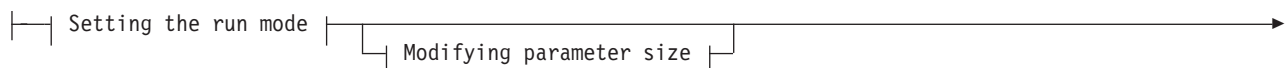
Note: The **onpload** and **onpladm** utilities include support for long object names up to 128 characters, but the **ipload** utility does not.

The following sections give additional information about the syntax and individual options of the **onpload** utility.

The onpload utility syntax



Other Options:



For more information on the options shown in the diagram, see “Setting the onpload run mode with the -f option” on page 16-4, “Modifying the size of onpload database parameters” on page 16-6, and “Overriding the onpload database values” on page 16-9.

The table below contains an explanation of the elements in the diagram.

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 path name of the file, tape, or pipe (UNIX only) or the name of the device array to use for the load or unload session	Additional Information: If the -f option is not set to a , d , or p , onpload assumes that the data source is a file. Additional Information: To use ipload , see “Interpreting the onpload -d and -f options together” on page 16-6.
-j <i>jobname</i>	Names a load or unload job from the onpload database	Additional Information: When using the -j option, you can override only a few values from the onpload database. See “Overriding the onpload database values” on page 16-9. To set using ipload , see “Components of the unload job” on page 11-1 and “Components of the load job” on page 12-1.
-m <i>map</i>	Names a map from the onpload database	Additional Information: To use ipload , see Figure 9-1 on page 9-2.
-p <i>projectname</i>	Identifies the project where the format and map are stored	Additional Information: To use ipload , see “Project organization” on page 4-1.

Element	Purpose	Key Considerations
-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 check box on the Load Job Select or Unload-Job Select windows.</p>

The **onpload** 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 -j bigload -p zz
```

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:

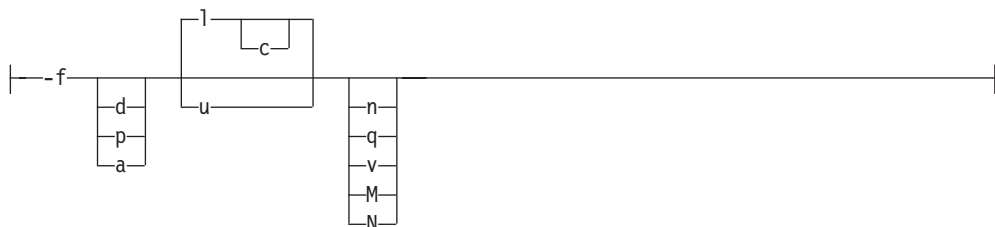
```
onpload -j bigload -p zz -fl -e 1000 -l /mylogs/buggytape.log
```

For information on the **-fl** option, see “Setting the onpload run mode with the **-f** option” on page 16-4.

Setting the onpload 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. If used along with the **-j** option, you can override only a few values from the **onpload** database. See “Overriding the onpload database values” on page 16-9.

Setting the Run Mode:



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 <i>-j job</i> 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 6-1.
c	Sets mode to deluxe mode.	Additional Information: If this flag is not set, onpload uses express mode. To use ipload , see "HPL modes" on page 15-2.
d	Treats data source as a device (tape or file).	Additional Information: To set this option using ipload , see "Device arrays" on page 6-1.
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 12-1.
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 13-8.
p	Treats data source as a program to run and reads interface to the program by way of a pipe (on UNIX only).	Additional Information: To use ipload , see "Device arrays" on page 6-1.
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 11-1.
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.

Typing the onpload -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 -flnc
```

```
onpload -m mymap -d mydev -fl -fn -fc
```

Interpreting the onpload -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 path name 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 -m mapname -d filename
```

- The **-fd** option in the following command causes **onpload** to treat **/dev/rmt/rst11b** as the name of a tape device:

```
onpload -m mapname -d /dev/rmt/rst11b -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 -m mapname -d tapearray3 -fa
```

UNIX Only

- 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 -m mapname -d apipename -fp
```

End of UNIX Only

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 -m mapname -d /dev/rmt/rst11 -fdu
```

Modifying the size of onpload database parameters

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
--I--commit_int
--a--iobufsize
--b--bufsize
--e--maxerrors
--i--prog_interval
--n--numrecs
--s--startrec
--t--numtapes

```

Element	Purpose	Key Considerations
<i>-A 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 in the onpload database” on page A-10.</p>
<i>-B 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 Figure 6-3 on page 6-4.</p>
<i>-G 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 Figure 12-4 on page 12-9.</p>
<i>-a 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 configuration parameter” on page B-2.</p>

Element	Purpose	Key Considerations
-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 (STRMBUFFSIZE) set in the HPL configuration file (plconfig).</p> <p>References: For specific details on this option, see “STRMBUFFSIZE configuration parameter” on page B-4.</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 Figure 12-4 on page 12-9.</p>
-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 onpload -i option” on page 16-9.</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 Figure 12-4 on page 12-9.</p>

Element	Purpose	Key Considerations
<code>-s startrec</code>	Sets the starting record to load.	<p>Additional Information: This option is used to skip records. For example, setting <code>-s</code> to 10 starts the loading at the tenth record, so that if the file contains 20 records, 11 records are loaded. Setting <code>-s</code> 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>The <code>-s</code> option does not affect unload operations because <code>pload</code> cannot maintain row ordering. To use <code>ipload</code>, see Figure 12-4 on page 12-9.</p>
<code>-t numtapes</code>	Specifies the number of tapes to load.	<p>Additional Information: If you do not set this option, the default value is 1. To use <code>ipload</code>, see Figure 12-4 on page 12-9.</p>

Using the `onpload -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 the progress message count in the `pload` log file as equal to the number of rows processed, but rounded down to the nearest multiple of the value of `progress_interval`. For example, if the number of rows processed is 910 and the value of `progress_interval` is 100, then the progress message count is 900.

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 topic let you enter size information that overrides existing parameters in the `onpload` database. You can override only these parameters. You cannot override other options, such as run mode.

Overriding the onpload Database Values:

```

-C caseconvert
-D override_db
-F filter
-L trace_level
-M converters
-R rejectfile
-S servername
-T target_db
-l 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> <p>U or u = uppercase L or l = lowercase P or p = proper names <i>blank</i> = no conversion</p> <p>To use ipload, see Figure 9-8 on page 9-9.</p>
-D <i>override_db</i>	Overrides the database specified in the map used for the load	<p>Additional Information: To use ipload, see Figure 9-3 on page 9-4.</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 10-1.</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>

Element	Purpose	Key Considerations
-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 CONVERTTHREADS 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 configuration parameter” on page B-3.</p>
-R <i>rejectfile</i>	Identifies the file destination for rejected records	<p>Additional Information: The file is named <i>rejectfile.rej</i>. To use ipload, see Figure 12-2 on page 12-5.</p>
-S <i>servername</i>	Sets the onpload database server	<p>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.</p> <p>Additional Information: To use ipload, see Figure 5-1 on page 5-2.</p>
-T <i>target_db</i>	Sets the target database server	<p>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.</p> <p>Additional Information: To use ipload, see Figure 5-1 on page 5-2.</p>
-l <i>logfile</i>	Specifies the name of a file to which onpload sends messages	<p>Additional Information: If you do not specify a log file, onpload sends messages to stdout. To use ipload, see Figure 11-2 on page 11-4 and Figure 12-2 on page 12-5.</p>

Loading data into collection data type columns

The **onpload** utility might need to create temporary smart large objects while loading data into collection data type columns. To create temporary smart large objects, **onpload** obtains sbspace information from the SBSPACETEMP configuration parameter, or if that parameter is not set, from the SBSPACENAME configuration parameter. For more information on temporary sbspaces, see the *IBM Informix Administrator's Guide*.

If **onpload** does not find a temporary sbspace, the load job fails with the following error message in the **onpload** log file:

```
Fatal error in server row processing - SQL error -9810 ISAM error -12053
```

If you see the above error message in **onpload** log file, configure a temporary sbspace using the SBSPACETEMP configuration parameter or a default sbspace using the SBSPACENAME configuration parameter and then restart the load job.

Chapter 17. The onpladm utility

Overview of the onpladm utility.	17-2
The onpladm utility features	17-2
Specification-file conventions	17-3
Error handling.	17-4
Defining onpladm utility jobs.	17-4
Creating onpladm jobs	17-4
Creating conversion jobs	17-5
Creating no-conversion jobs	17-10
Modifying a job by using a detailed specification file	17-13
Describing a job	17-14
Listing all jobs in a project	17-14
Running a job	17-15
Using the onpladm utility when referential constraints are on tables	17-16
Deleting a job	17-16
Defining device arrays.	17-17
Creating a device array	17-17
Modifying a device array.	17-18
Describing a device array.	17-18
Listing project device arrays.	17-18
Deleting a device array	17-19
Defining maps	17-19
Creating maps	17-19
Creating a map by using a quick command	17-19
Creating a map with a detailed specification file	17-21
Deleting a map	17-24
Describing a map	17-24
Modifying a map using a detailed specification file	17-25
Listing all maps in a project	17-25
Defining formats	17-26
Creating a format	17-26
Modifying a format using a specification file	17-28
Describing a format	17-28
Listing all formats in a project	17-29
Deleting a format	17-29
Defining queries.	17-30
Creating a query.	17-30
Modifying a query	17-30
Describing a query	17-30
Listing all queries in a project	17-31
Deleting a query.	17-31
Defining filters	17-32
Creating a filter	17-32
Modifying a filter	17-33
Describing a filter	17-33
Listing all filters in a project.	17-33
Deleting a filter	17-34
Defining projects	17-34
Creating a new project.	17-34
Running all jobs in a project.	17-34
Listing all projects	17-35
Deleting a project	17-35
Defining machine types	17-36
Creating a machine type	17-36
Modifying a machine type	17-36
Describing a machine	17-37

Listing all existing machine types	17-37
Deleting a machine type	17-37
Defining database operations	17-38
Creating a database project	17-38
Configuring target-server attributes	17-40
Setting target-server attribute values	17-40
Listing target-server defaults	17-41

Overview of the onpladm utility

The **onpladm** command-line interface is equivalent to the **ipload** 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 the database server running on Windows” on page I-1.

Note: While the **onpload** and **onpladm** utilities include support for object names that contain up to 128 characters, the **ipload** utility does not. If you use long database, table or column names and create jobs using **onpladm**, you cannot run these jobs using **ipload**. For **ipload**, database, table and column names cannot exceed 18 characters.

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.
- Run individual jobs or projects independently.

Important: The most effective way to use **onpladm** is to create and modify HPL objects is to:

1. Create the default objects using default job creation options.
2. Use describe *object* syntax to describe the objects.
3. Manually modify these objects with correct values for different configuration parameters.
4. Use modify *object* syntax to correctly create these objects.

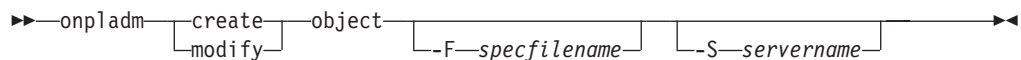
Specification-file conventions

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.

Creating or Modifying an Object with a Specification File



Element	Purpose	Key Considerations
-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.

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
TAPEDEVICESIZE 0
PIPECOMMAND
  
```

```

END SEQUENCE
BEGIN SEQUENCE
TYPE PIPE
FILE
TAPEBLOCKSIZE 0
TAPEDEVICESTRIP 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 onpladm utility 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 12-1.

Creating onpladm 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

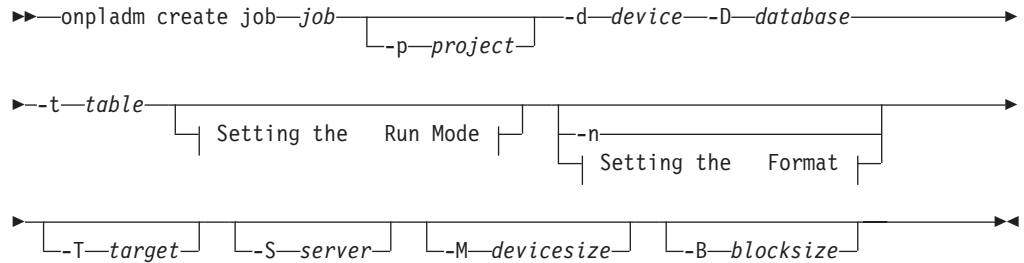
When you use command-line parameters to create a conversion load or unload job, all maps, formats, and query objects automatically have the same name as the job name. If you delete a job, the **map.format** or the query objects will remain intact unless you specifically delete the format or query objects.

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.

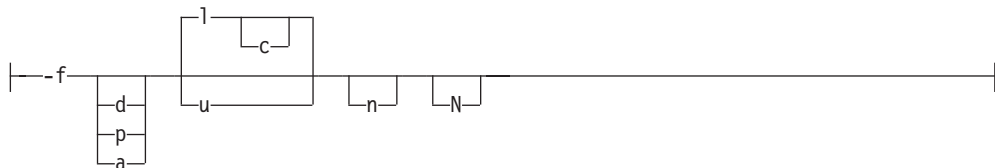
Creating a Conversion Job



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.

Setting the Run Mode:



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

Element	Purpose	Key Considerations
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 run 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.

Setting the Format:



Element	Purpose	Key Considerations
D	Sets the format to delimited	Additional Information: This is the default value. See “Delimited records” on page 7-12.
FI	Sets the format to fixed internal	Additional Information: See “Fixed-length records” on page 7-2.
FA	Sets the format to fixed ASCII	Additional Information: See “Fixed-length records” on page 7-2.
FB	Sets the format to fixed binary format.	Additional Information: See “Fixed-length records” on page 7-2.
C	Sets the format to COBOL	Additional Information: See “COBOL records” on page 7-15.
CB	Sets the format to COBOL (byte)	Additional Information: See “COBOL records” on page 7-15.

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 17-3 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

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.
<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

Creating a conversion unload job: Use the syntax shown in “Specification-file conventions” on page 17-3 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

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.
<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

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.

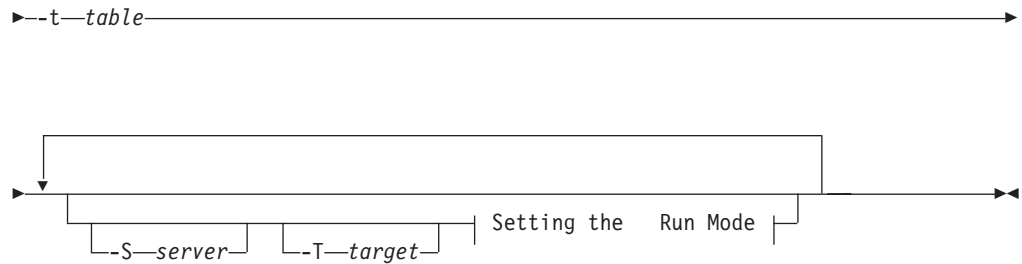
Creating no-conversion jobs using a quick command: When you create a job using a quick command, the **onpladm** utility creates all HPL objects associated with that job.

The following diagram illustrates the syntax to create a no-conversion job.

Creating a No-Conversion Job

```

▶▶ onpladm create job—job—┌──────────┐ —n—-d—device—-D—database—▶▶
                             └─p—project─┘
  
```



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.

Setting the Run Mode:



Element	Purpose	Key Considerations
a	Treats the data source as a device-array	Additional Information: The definition of the device array is extracted from the onpload database.
d	Treats the data source as a tape device	None
l	Loads data into the database	Additional Information: This is the default flag, as opposed to u , which unloads data from the database.
p	Treats the data source as a program to run, and reads the interface to the program by way of a pipe (UNIX only)	None
u	Unloads data from the 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 17-3 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

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 17-3 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

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

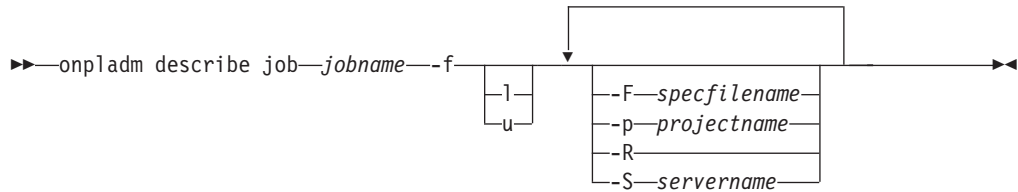
Use the syntax shown in “Specification-file conventions” on page 17-3 to modify a job by using a specification file.

For information about the job-specification file, refer to “Creating no-conversion jobs using detailed specification files” on page 17-12.

Describing a job

The following diagram illustrates the syntax to describe a job.

Describing a Job



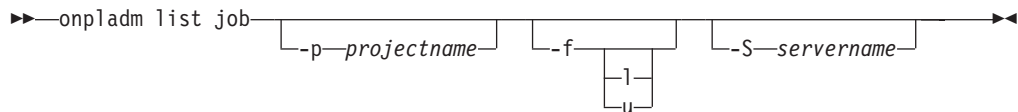
Element	Purpose	Key Considerations
<code>-F specfilename</code>	Sets the specification file	Additional Information: The default value is the standard output.
<code>-f</code>	Flags to specify the type of job	Additional Information: The default is load job.
<code>l</code>	Specifies a load job	None
<code>u</code>	Specifies an unload job	None
<code>jobname</code>	Names a job from the onpload database	None
<code>-p projectname</code>	Identifies the project where the format and map are stored	None
<code>-R</code>	Deletes all associated objects if they are not referenced by other objects	None
<code>-S servername</code>	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 17-8.

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

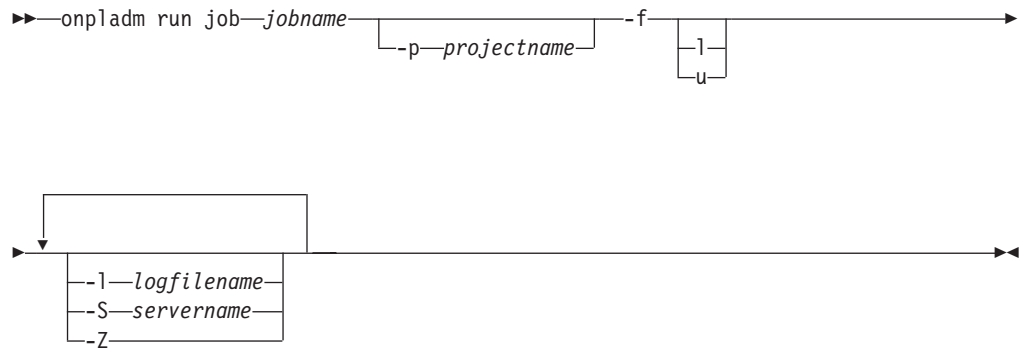


Element	Purpose	Key Considerations
-f	Flags to specify the type of job	Additional Information: The default is load job.
l	Specifies a load job	None
u	Specifies an unload job	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload 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.

Running a Job



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 run this command for a database server running on Windows, see “Running onpladm on UNIX with the database server running on Windows” on page I-1.

Using the onpladm utility when referential constraints are on tables

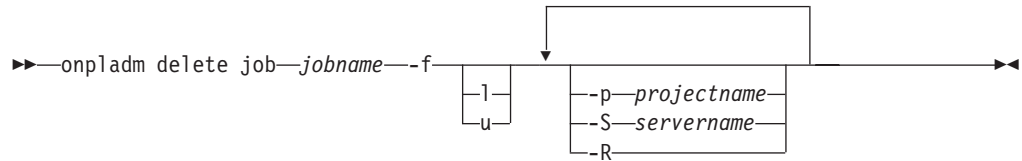
The **onpladm** utility unloads and loads an entire database by creating individual table unload and load jobs. If the load mode is **express**, the default load mode, the **onpladm** utility disables referential constraints during load jobs. If database tables have constraints on them, problems can occur. For example, if one table has a referential constraint to another table and that table is not yet loaded, a violation that prevents a table from loading can occur.

If tables have constraints, you can manually disable the constraints after the load job is complete.

Deleting a job

The following diagram illustrates the syntax to delete a job.

Deleting a Job



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 6-1 and “Devices for the device array” on page 15-8.

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 17-3 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)
<i>tapeblock_size</i>	Tape-block size
<i>tapedevice_size</i>	Tape-device size in megabytes
<i>pipe_commandname</i>	Pipe command name

Modifying a device array

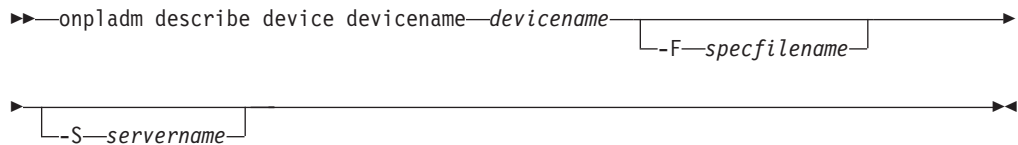
You can only modify a device array with a specification file. For information about the device array specification file, refer to “Creating a device array” on page 17-17.

Use the syntax shown in “Specification-file conventions” on page 17-3 to modify a device array.

Describing a device array

The following diagram illustrates the syntax to describe a device array.

Describing 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
<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.

For information about the specification file used to describe a device array, see “Creating a device array” on page 17-17.

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
<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 device array

The following diagram illustrates the syntax to delete a device array.

Deleting Project Device Arrays

```
▶▶ onpladm delete device devicename [-S servername]
```

Element	Purpose	Key Considerations
<code>devicename</code>	Sets the name of device, such as a file, device array, tape, or pipe	None
<code>-S servername</code>	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 9-1.

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 17-3.

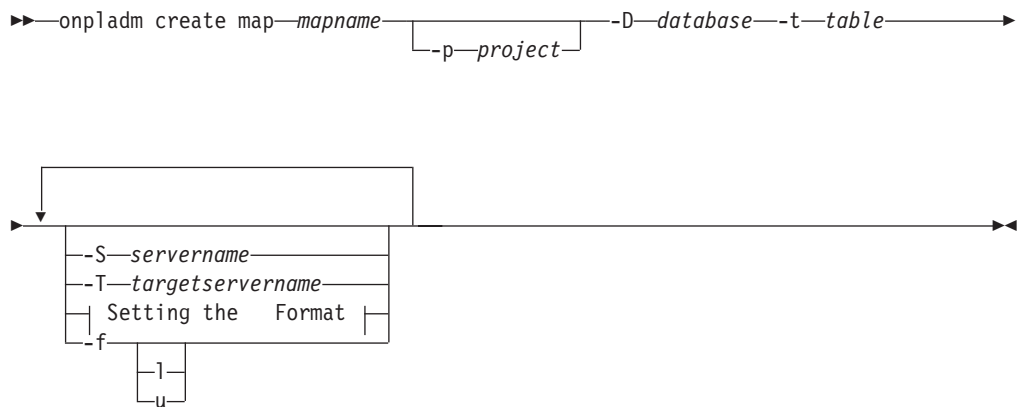
Creating a map by using a quick command

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.

Creating a Map



Element	Purpose	Key Considerations
-D <i>database</i>	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
<i>mapname</i>	Sets the map	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 format type with the **-z** option.

Setting the Format:



Element	Purpose	Key Considerations
D	Sets the format to delimited	Additional Information: This is the default value. See “Delimited records” on page 7-12.
FI	Sets the format to fixed internal	Additional Information: See “Fixed-length records” on page 7-2.
FA	Sets the format to fixed ASCII	Additional Information: See “Fixed-length records” on page 7-2.
FB	Sets the format to fixed binary	Additional Information: See “Fixed-length records” on page 7-2.
C	Sets the format to COBOL	Additional Information: See “COBOL records” on page 7-15.
CB	Sets the format to COBOL (byte)	Additional Information: See “COBOL records” on page 7-15.

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 17-3 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

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 7-11.
<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
<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," on page E-1.
<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

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*

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 7-11.
<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
<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," on page E-1.

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 17-21.

Modifying a map using a detailed specification file

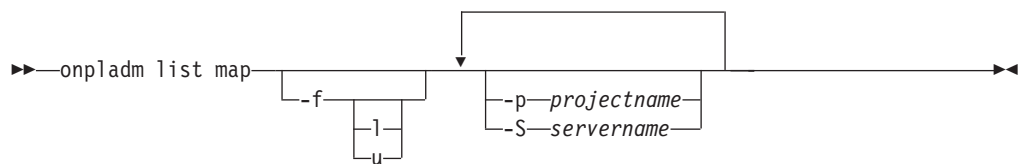
Use the syntax shown in “Specification-file conventions” on page 17-3 to modify an existing map with a specification file.

For information about the map-specification file, refer to “Creating a map with a detailed specification file” on page 17-21.

Listing all maps in a project

The following diagram illustrates the syntax to list all the maps in a project.

Listing All 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 of supported data-file records” on page 7-2.

Creating a format

You can only create formats by using a detailed specification file.

Use the syntax shown in “Specification-file conventions” on page 17-3 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 7-5.
<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
<i>machine_type</i>	Type of computer that produced the data, such as a SPARCstation See “Listing all existing machine types” on page 17-37.
<i>offset_bytes</i>	Number of bytes of the field offset in the record
<i>projectname</i>	Name of existing project

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/registry
<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
<i>picture_description</i>	Picture description that matches the record FD from the COBOL program See Appendix C, "Picture strings," on page C-1.
<i>projectname</i>	Name of existing project
<i>usage_description</i>	Number of bytes that the field occupies in the record

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 7-12.
<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.
<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

Modifying a format using a specification file

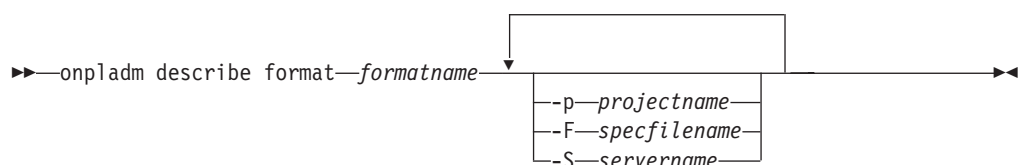
You can only modify a format using a specification file. Use the syntax shown in "Specification-file conventions" on page 17-3 to modify a format.

For more information on the corresponding specification-file format, see "Creating a format" on page 17-26.

Describing a format

The following diagram illustrates the syntax to describe a format to a file.

Describing a Format



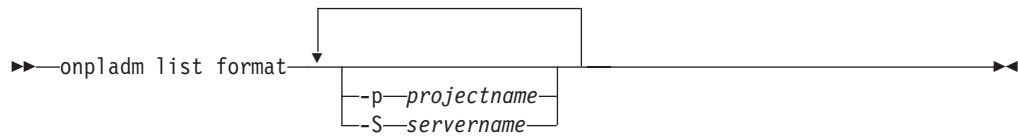
Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>formatname</i>	Sets the format name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	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 17-26.

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

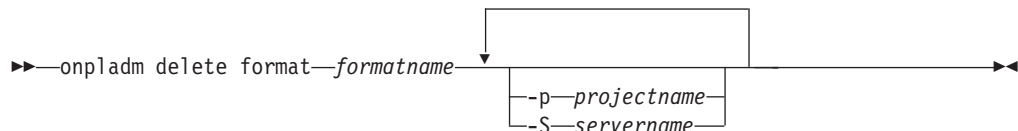


Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a format

The following diagram illustrates the syntax to delete a format.

Deleting All Formats in a Project



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 17-3 to create a query.

Use the following syntax to create a 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 17-30.

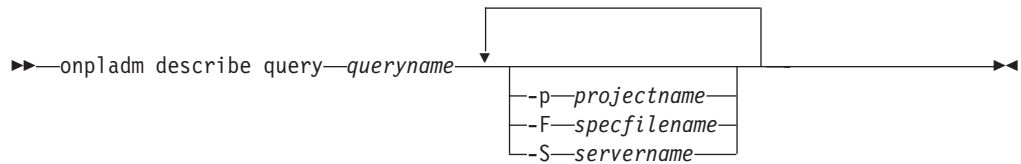
Use the syntax shown in “Specification-file conventions” on page 17-3 to modify a query by using a specification file.

For more information on the corresponding specification-file format to modify a query, see “Creating a query” on page 17-30.

Describing a query

The following diagram illustrates the syntax to describe a query.

Describing a Query



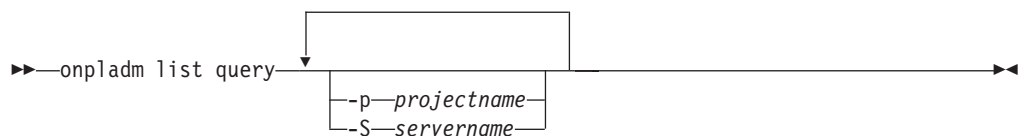
Element	Purpose	Key Considerations
-F <i>specfilename</i>	Sets the specification file	Additional Information: The default value is the standard output.
<i>queryname</i>	Sets the query name	None
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	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 17-30.

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

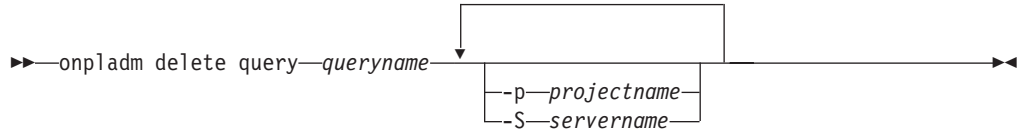


Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a query

The following diagram illustrates the syntax to delete a query.

Deleting 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 **onploadm** utility. For more information, see “Using a filter” on page 10-1.

Creating a filter

You can only create a filter by using a specification file.

Use the syntax shown in “Specification-file conventions” on page 17-3 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,” on page D-1.
<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

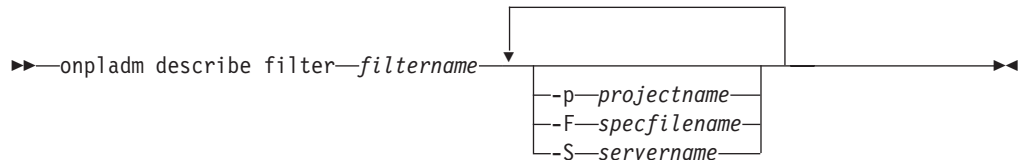
You can only modify a filter by using a detailed specification file. For information about the filter specification file, see “Creating a filter” on page 17-32.

Use the syntax shown in “Specification-file conventions” on page 17-3 to modify a filter.

Describing a filter

The following diagram illustrates the syntax to describe a filter.

Describing a Filter



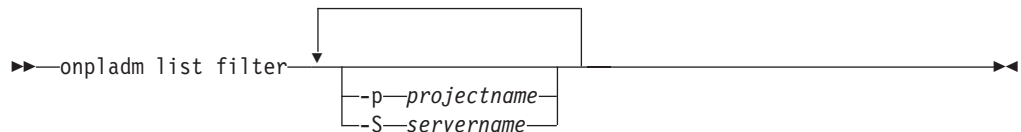
Element	Purpose	Key Considerations
<code>-F specfilename</code>	Sets the specification file	Additional Information: The default value is the standard output.
<code>filtername</code>	Sets the filter name	None
<code>-p projectname</code>	Identifies the project where the format and map are stored	None
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information about the specification file to describe a filter, see “Creating a filter” on page 17-32.

Listing all filters in a project

The following diagram illustrates the syntax to list all the filters in a project.

Listing All Filters in a Project

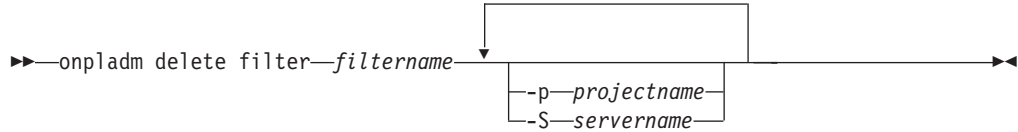


Element	Purpose	Key Considerations
<code>-p projectname</code>	Identifies the project where the format and map are stored	None
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Deleting a filter

The following diagram illustrates the syntax to delete a filter.

Deleting a Filter



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 4-1.

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

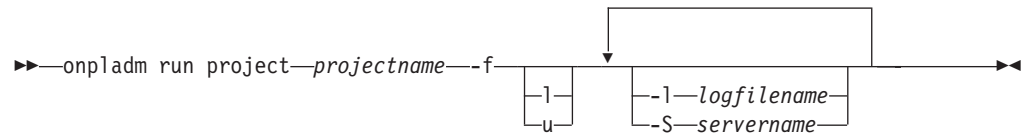


Element	Purpose	Key Considerations
-p <i>projectname</i>	Identifies the project where the format and map are stored	None
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Running all jobs in a project

The following diagram illustrates the syntax to run all load or unload jobs in a project.

Running All 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 logfile	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 servername	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

For information on how to run this command for a database server running on Windows, see “Using the onpladm utility on Windows” on page I-1.

Listing all projects

The following diagram illustrates the syntax to list all your projects.

Listing All Projects



Element	Purpose	Key Considerations
-S servername	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



Element	Purpose	Key Considerations
<i>projectname</i>	Identifies the project where the format and map are stored	None
<i>-S 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 17-3 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
INTEGERSIZE 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
<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

Modifying a machine type

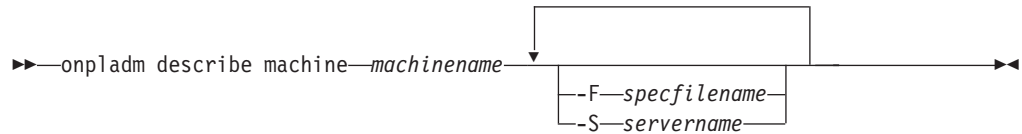
You can only modify a machine type by using a detailed specification file. For information on the machine type specification file, refer to “Creating a machine type” on page 17-36.

Use the syntax shown in “Specification-file conventions” on page 17-3 to modify a machine type.

Describing a machine

The following diagram illustrates the syntax to describe a machine.

Describing a Machine



Element	Purpose	Key Considerations
<code>-F specfilename</code>	Sets the specification file	Additional Information: The default value is the standard output.
<code>machinename</code>	Sets the machine name	None
<code>-S servername</code>	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 17-36.

Listing all existing machine types

The following diagram illustrates the syntax to list the machine types.

Listing All Machines



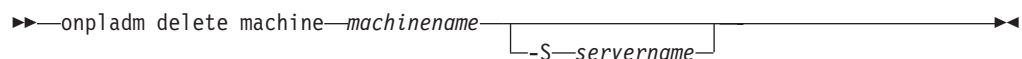
Element	Purpose	Key Considerations
<code>-S servername</code>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

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
-S <i>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.

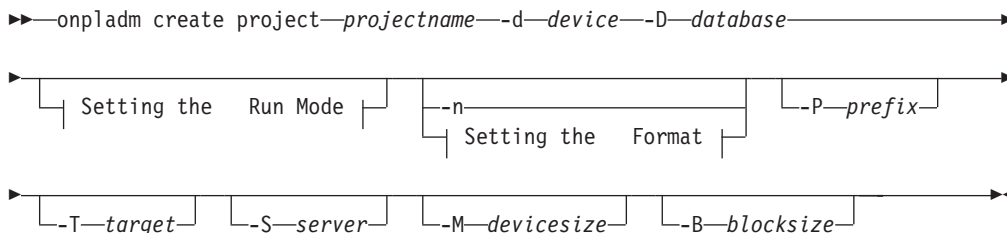
If you do not specify a prefix, the **onpladm** utility creates files of the following format:

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.

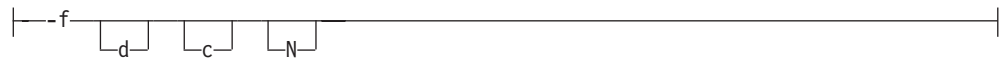
Creating a Database Project



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.

Setting the Run Mode:



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.

Setting the Format:



Element	Purpose	Key Considerations
D	Sets the format to delimited	Additional Information: This is the default value. See “Delimited records” on page 7-12.
FI	Sets the format to fixed internal	Additional Information: See “Fixed-length records” on page 7-2.
FA	Sets the format to fixed ASCII	Additional Information: See “Fixed-length records” on page 7-2.
FB	Sets the format to fixed binary	Additional Information: See “Fixed-length records” on page 7-2.
C	Sets the format to COBOL	Additional Information: See “COBOL records” on page 7-15.
CB	Sets the format to COBOL (byte)	Additional Information: See “COBOL records” on page 7-15.

Configuring target-server attributes

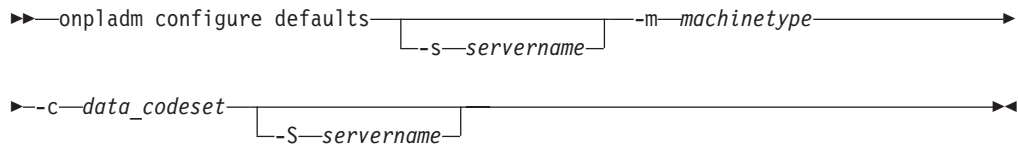
To configure target-server attributes, you must:

- 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.

Setting Target Server Attribute Values



Element	Purpose	Key Considerations
-c <i>data_codeset</i>	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 <i>machinetype</i>	Machine type	None
-s <i>servername</i>	Database server for which defaults are set	Additional Information: If a server is not specified, the default information within the onpload database that describes all database servers is modified.
-S <i>servername</i>	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.

Listing Target Server Defaults

```

▶▶ onpladm list defaults -s servername -S servername

```

Element	Purpose	Key Considerations
-s <i>servername</i>	Database server for which defaults are set	Additional Information: If a server is not specified, the default information within the onpload database that describes all database servers is modified.
-S <i>servername</i>	Sets the onpload database server	Additional Information: The default is the value of the INFORMIXSERVER environment variable.

Appendix A. 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 in the onpload database

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 default values" on page 5-2.

Delimiters table in the onpload database

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 7-19 and Figure 7-22 on page 7-20.

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 in the onpload database

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 6-2.

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 (UNIX only), file, or tape)
file	CHAR(128)	File or device to be accessed by this array element
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 (UNIX only)
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses
header	TEXT	The tape header for a device that DDR uses

Driver table in the onpload database

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 5-4.

Filteritem table in the onpload database

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 10-2.

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(128)	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 in the onpload database

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 Figure 10-2 on page 10-4.

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(128)	The name of the filter
lockflag	CHAR(1)	Flag for locking mechanism used by ipload

Formatitem table in the onpload database

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. Table A-1 on page A-5 lists the possible values for the **fctype** column. For more information, see Figure 7-3 on page 7-4.

Column	Type	Description
formid	INTEGER	Record format identifier (foreign key to the formats table)
seq	INTEGER	Item sequence number for internal organization
fname	CHAR(128)	Name of record field
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

Table A-1. Possible values for the ftype column

ftype Value	Type of Data
1	Character (fixed and delimited)
2	Date
3	Short integer
4	Integer
5	Long Integer
6	Floating-point vale
7	Double floating-point value
8	Unsigned short integer
9	Unsigned integer
10	Unsigned long integer
11	UNIX date
18	Packed Decimal
19	Zoned decimal
20	Comp-1
21	Comp-2
22	Comp-3
23	Comp-4
24	Comp-5
25	Comp-6
26	Comp-X
27	Comp-N
28	Character (COBOL)
34	Blob Length
35	Blob File
36	Blob HexASCII
37	Blob Text
39	INT8
40	SERIAL8
41	BOOLEAN
42	Extended Type String
43	Extended Type HexASCII
44	Extended Type Binary
45	Extended Type StringLength
46	Extended Type BinaryLength
49	BIGINT
50	BIGSERIAL
51	INT64 (64-bit integer)

Formats table in the onpload database

The **formats** table defines the basic information for a record format. Use the Records Format window to modify this table. For more information, see Figure 7-2 on page 7-3.

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(128)	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
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

Language table in the onpload database

The **onpload** utility does not use the **language** table at this time.

Machines table in the onpload database

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 5-6.

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 in the onpload database

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 9-1.

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	VARCHAR(128,0)	Name of database column
fname	CHAR(128)	Name of field in a data-file record

Mapoption table in the onpload database

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 the mapping options” on page 9-8.

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

Column	Type	Description
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
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

If the values of **inputcode** and **storecode** are different, **onpload** converts the contents of the BYTE or TEXT data.

Maps table in the onpload database

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 9-1.

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(128)	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
query	CHAR(128)	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

Note table in the onpload database

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 3-14.

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 in the onpload database

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 4-1.

Column	Type	Description
name	CHAR(128)	Name of object
projectid	SERIAL	Uniquely identifies the project (primary key)
dcreate	DATE	Date that the project was created

Query table in the onpload database

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 8-1.

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(128)	Name of the query
database	CHAR(30)	Name of database being queried
arrayname	CHAR(128)	Not in use
lockflag	CHAR(1)	Flag for locking mechanism that ipload uses
sqlselect	TEXT	SQL statement of the query

Session table in the onpload database

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: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• 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(130)	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: <ul style="list-style-type: none">• 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)
infile	CHAR(160)	Name of the device array (foreign key to the name column of the device table)

Column	Type	Description
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(128)	Filter for screening import data (foreign key to the name column of the filters table)
recordfilter	CHAR(384)	File in which to store filtered records
suspensefile	CHAR(384)	File in which to store records that do not pass conversion
rejectfile	CHAR(384)	File in which to place records that the database server rejected
logfile	CHAR(384)	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
sourctrace	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 block size for accessing device
filetype	INTEGER	Specifies the type of file: tape, array, pipe (UNIX only)
number_records	INTEGER	Specifies the number of records to load
start_record	INTEGER	Specifies the number of the record at which to start loading
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.)

Column	Type	Description
runmode	INTEGER	<p>Contains a value that qualifies onpload to perform a load or unload operation.</p> <p>For load operations, the runmode is a combination of the following values:</p> <ul style="list-style-type: none"> • 0x1 = Deluxe mode • 0x2 = Express mode • 0x80 = Data conversion required • 0x100 = No violation generated • 0x1000 = Load is not replicated (CDR) <p>For example, you could have this command for a load operation:</p> <pre>onpladm create job j4 -f1N -D tst -t tab1 -d data.unl</pre> <p>For this command for a load job, the runmode value is:</p> <pre>0x00001081 = Deluxe+Conversion+NoReplication</pre> <p>For unload operations, the runmode is a combination of the following values:</p> <ul style="list-style-type: none"> • 2 = No conversion • 129 = Conversion (128) + dirty read isolation level (1) • 130 = Conversion (128) + committed read isolation level (2) • 131 = Conversion (128) + cursor stability isolation level (3) • 132 = conversion (128) + repeatable read isolation level (4)
loadmode	INTEGER	<p>Type of job:</p> <ul style="list-style-type: none"> • 1 = Load • 2 = Unload
caseconvert	INTEGER	<p>Case conversion type. Convert to:</p> <ul style="list-style-type: none"> • U or u = Uppercase • L or l = Lowercase • P or p = Proper names
commitinterval	INTEGER	<p>Commit interval for committing a load transaction.</p> <p>(The value is specified in the Load Options window, 12-9. The commit interval applies only to deluxe mode.)</p>
socketport	INTEGER	<p>Set by onpload to specify the port number of the connection</p>
numtapes	INTEGER	<p>Number of tapes to load</p>

Tip: Deluxe-mode loads do not support the “no conversion” option and the “with conversion and do not generate violations table” option.

Appendix B. 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`.

HPL configuration parameter descriptions

The description of each configuration parameter has one or more of the following fields (depending on their relevance):

<i>default value</i>	The value that appears in the <code>plconfig.std</code> 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

HPL configuration parameter 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 configuration parameter

<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>	“Assessing information for loading or unloading external data” on page 15-8

The AIOBUFFERS configuration parameter sets the number of buffers used to transport data from converter threads to the AIO handler.

Windows Only

You must set the AIOBUFFERS parameter on Windows computers to a minimum of 8.

End of Windows Only

AIOBUFSIZE configuration parameter

<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>	“Assessing information for loading or unloading external data” on page 15-8

The AIOBUFSIZE configuration 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 6-4). 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 configuration parameter

<i>default value</i>	1
<i>range of values</i>	Minimum: 1 Maximum: depends on computer configuration
<i>refer to</i>	“Assessing information for loading or unloading external data” on page 15-8

The CONVERTTHREADS configuration 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:

CONVERTTHREADS * 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 configuration parameter

<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>	“Assessing information for loading or unloading external data” on page 15-8

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 configuration parameter

<i>default value</i>	0
<i>range of values</i>	0 or 1 0 = The custom conversion or driver function receives three arguments: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• 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
<i>refer to</i>	“The onpload conversion process” on page E-1

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 configuration parameter

<i>default value</i>	<code>\$INFORMIXDIR/lib/ipldd11a.so</code>
<i>range of values</i>	Any valid directory, plus <code>ipldd11a.SOLIBSUFFIX</code> . (<i>SOLIBSUFFIX</i> 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 <code>\$INFORMIXDIR</code> .
<i>refer to</i>	“Rebuilding the shared-library file” on page H-3

The `ipldd11a.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,” on page H-1. For more information on custom conversion functions, see Appendix E, “Custom-conversion functions,” on page E-1.

Previous versions of the database server required the `ipldd11a.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 configuration parameter

<i>default value</i>	Maximum of $(4,2 * \text{CONVERTTHREADS})$
<i>recommended value</i>	Maximum of $(4,2 * \text{CONVERTTHREADS})$
<i>range of values</i>	Integer > 4
<i>refer to</i>	“Assessing information for loading or unloading external data” on page 15-8

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:

`STRMBUFFERS * numdevices`

where *numdevices* is the number of devices in the current array.

STRMBUFFSIZE configuration parameter

<i>default value</i>	64
<i>units</i>	Kilobytes
<i>range of values</i>	Minimum: $2 * \text{operating system page size}$ Maximum: depends on operating system resources
<i>refer to</i>	“Assessing information for loading or unloading external data” on page 15-8

The STRMBUFFSIZE configuration parameter sets the size of a server-stream buffer. Larger buffers are more efficient because moving buffers around requires less overhead.

Appendix C. 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 7-15. 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 9-8.

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

Fills 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

Appendix D. 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>	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. 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.
<i>value</i>	Equals (=) is the default operator. Thus this case is equivalent to =value, except that the characters do <i>not</i> have to be delimited by quotes. 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.
> <i>value</i>	Matches if the data record field is greater than the specified value. 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. Character strings must be delimited by quotes (> "Jones").
< <i>value</i>	Matches if the data record field is less than the specified value. 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. Character strings must be delimited by quotes (< "Jones").
>= <i>value</i>	Matches if the data-record field is equal to or greater than the specified value. For example, if you are matching on a field named Income, the match condition >= 50000 selects all records whose Income field contains an entry of 50,000 or greater. Character strings must be delimited by quotes (>= "Jones").
<= <i>value</i>	Matches if the data-record field is less than or equal to the specified value. For example, if you are matching on a field named Income, the match condition <= 50000 selects all records whose Income field contains an entry of 50,000 or less. Character strings must be delimited by quotes (<= "Jones").
<> <i>value</i>	Matches if the data-record field is not equal to the specified value. Character strings must be delimited by quotes. For example, if you are matching on a field named State, the match condition <>"TX" selects all records whose State field contains an entry other than TX.
between <i>value1</i> and <i>value2</i>	Matches if the data-record field is between the range specified in value 1 and value 2. For example, if you are matching on a field named Income, the match condition between 50000 and 100000 selects all records whose Income field contains an entry between 50,000 and 100,000. Character strings must be delimited by quotes.
and	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. For example, if you are matching a field named Income, the match condition > 5000 and <> 6000 selects all the records with income greater than 5000, but not a record of 6000.
or	Constructs a comparison of two or more items. Matches if the data record fields match <i>any</i> of the comparisons. For example, if you are matching on a field named City, the match condition ="Dallas" or = "Fort Worth" selects all records whose City field contains either the entry Dallas or the entry Fort Worth.
NULL	Matches when all characters are blank or when a character is binary zero (null). For example, you might want to discard any records that have all blanks for a name field.
* (asterisk)	Wildcard match of any number of characters in a string. For example, to match on a field that contains the city name and state, the match condition Dal1* would select records with any of the following entries: <ul style="list-style-type: none">• Dallas-Forth Worth• Dallas, TX• Dallas TX

Operator	Description
?	Matches any single character in a string. For example, to match on a field that contains a last name, the match condition Sm?th would select records with any of the following entries: <ul data-bbox="313 289 407 352" style="list-style-type: none">• Smith• Smyth

Appendix E. 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 9-8.

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 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 that **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 configuration parameter” on page B-3.

- 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 9-8.

```
/*
 * 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 **ipldd11a.SOLIBSUFFIX**, (where **SOLIBSUFFIX** is the shared-library suffix for your platform). Follow the instructions in “Rebuilding the shared-library file” on page H-3.

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

Appendix F. 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 *IBM Informix 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 the database server, with the following exceptions:

```

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

```

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**).

Appendix G. 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:

UNIX Only

- **/usr/include/errno.h**

End of UNIX Only

Windows Only

- **errno.h**, **windsock.h**, and **winsoc2.h** in the **include** subdirectory for Microsoft Visual C++.

End of Windows Only

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 14-5.

How HPL log-file 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.

HPL log-file 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. 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.

HPL log-file messages

Blob conversion error occurred on record *record_num*.

Explanation: The SQLBYTE simple large object data could not be converted to HEXASCII, or the SQLTEXT simple large object data has invalid character data (characters not in the code set).

User response: Remove the invalid characters from the input data.

Cannot access database table *table_name*: **SQL error** *error_num*.

Explanation: The target database table cannot be accessed.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot allocate shared memory.

Explanation: A memory allocation error occurred. Probably the system is out of virtual shared memory.

User response: 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.**

Explanation: System memory cannot be allocated for communications. This situation should only happen if all system resources are consumed.

User response: Note the circumstances and contact Technical Support.

Cannot bind socket connection: errno=*operating-system_error_num*.

Explanation: A TCP socket cannot be opened.

User response: See your **errno.h** file.

Cannot bind TLI connection: t_errno=t_error_num.

Explanation: An error occurred when **onpload** attempted to open a TLI connection.

User response: Check that TLI services are installed on the operating system. See your **tiuser.h** file.

Platform: UNIX Only

Cannot configure driver driver_name.

Explanation: 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.

User response: For information on building a shared library, see "Rebuilding the shared-library file" on page H-3. Make sure your driver is configured correctly for **Fixed**, **Cobol**, or **Delimited**.

Platform: UNIX Only

Cannot connect to message server: Socket error=UNIX_error_num.

Explanation: This message is generated by **ipload** when it cannot connect to the **onpload** socket service.

User response: See **/usr/include/errno.h**.

Platform: UNIX Only

Cannot connect to message server: TLI error=t_error_num, TLI event=t_event_num, errno=error_num.

Explanation: An error occurred when **onpload** attempted to open a TLI connection.

User response: Check that TLI services are installed on the operating system. See **/usr/include/tiuser.h** (**t_error_num**).

Platform: UNIX Only

Cannot connect to server_name: SQL error error_num, ISAM error error_num.

Explanation: The target database server cannot be opened.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot connect worker to server data stream.

Explanation: A possible permissions problem exists for **onpload** or **oninit**.

User response: Note the circumstances and contact Technical Support.

Cannot disable table_name object constraints: SQL error error_num, ISAM error error_num.

Explanation: 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.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot disable primary-key constraint. Child-table references exist.

Explanation: You attempted to use express mode to load a table that has child-table records that refer to it. The 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.)

User response: Perform the load in deluxe mode or remove the constraint in question.

Cannot express load to logged table on HDR server *server_name*.

Explanation: You attempted to use express mode to load an HDR replicated table. The express mode does not support this condition.

User response: Perform the load in deluxe mode.

Cannot filter indexes for table *table_name*: **SQL error** *error_num*, **ISAM error** *error_num*.

Explanation: 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.

User response: 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*.

Explanation: The **ipldd11a.so** shared library path is not set in the **plconfig** file.

User response: 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*.

Explanation: The **onpload** process could not find the required user-defined function in the shared library.

User response: 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*.

Explanation: Cannot access the **systable** table to get dictionary information for the indicated table.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot initialize shared library handling.

Explanation: The **pload** utility cannot start because it failed to initialize the shared library-handling functionality.

User response: 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*.

Explanation: 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.

User response: Check that the file exists. Check the file for permissions.

Cannot load mapping definitions.

Explanation: A memory-allocation error or database-integrity error occurred when **onpload** accessed the **onpload** database.

User response: 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 persist, contact Technical Support.

Cannot load the shared library *library_location*.
Shared library load failed with error message *error_message*.

Explanation: The **onpload** utility could not load the shared library from the *library_location* path and failed with *error_message*.

User response: 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.

Explanation: No delimiter is found when **onpload** scans for an end-of-record delimiter in the load data.

User response: 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.

Explanation: An internal error occurred when **onpload** attempted to open the load or unload file.

User response: Note the circumstances and contact Technical Support.

Cannot open simple large object file:*file_name*, **simple large object not loaded.**

Explanation: The record references a filename that should contain a simple large object, but the file cannot be located.

User response: Check that the simple-large-object file exists.

Cannot open database *database_name*: **SQL error** *error_num*, **ISAM error** *error_num*.

Explanation: The target database cannot be opened.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Cannot open file *file_name*: **error number** *operating-system_error_num*.

Explanation: The file cannot be opened.

User response: See your **errno.h** file.

Cannot open TCP connection for *server_name*: **errno** *operating-system_error_num*.

Explanation: A TCP socket cannot be opened.

User response: See your **errno.h** file.

Cannot perform express mode load on table with pseudo rowid.

Explanation: The load table is fragmented by row ID. The express mode does not support this condition.

User response: Perform the load in deluxe mode.

Cannot perform express-mode load with rowsize=row_length > page_size.

Explanation: The table-row size exceeds page size. The express mode does not support this condition.

User response: Perform the load in deluxe mode.

Cannot read file *file_name*: AIO error code *operating-system_error_num*.

Explanation: 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.

User response: 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*.

Explanation: 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.

User response: Review the information in the **violations** and **diagnostics** tables.

Cannot re-order query statement to align simple large objects or Ext Types.

Explanation: The unload query does not contain a FROM clause.

User response: Rewrite the query so that it contains a FROM clause.

Cannot re-order query statement to align blobs.

Explanation: The unload query does not contain a FROM clause.

User response: 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*.

Explanation: The object constraints are disabled during the load and re-enabled after the load. An error occurred when **onpload** attempted to reset object constraints back to their original state.

User response: 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*.

Explanation: An error occurred when **onpload** attempted to set up the violations table for the load table.

User response: 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*.

Explanation: 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.

User response: 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.

Explanation: The server determined that the query cannot be run in parallel.

User response: 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*.

Explanation: The unload file cannot be accessed.

User response: See your **errno.h** file.

Code-set conversion overflow.

Explanation: 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.

User response: To unload this data, use a delimited format.

Conversion of onpload database failed due to error *error_num*.

Explanation: The **onpload** tried to convert the old database when **onpload** ran for the first time on the new database server. This conversion failed because of the error referenced in the error message.

User response: For more information, use the **finderr** or **Informix Error Messages** utility. Resolve this before you rerun **onpload**.

Conversion of onpload database failed due to error *error_num*, run as user informix.

Explanation: Database conversion fails because the current user running **onpload** does not have sufficient privileges to convert the **onpload** database.

User response: Run the **onpload** job as user **informix** once.

Custom conversion function *function_name* not found in shared library.

Explanation: The custom function specified in a map option was not located in **ipldd11a.so**. The shared library extension is platform-specific; for example, the **.so** extension is specific for Solaris and is probably different on other platforms.

User response: For information on how to configure the custom function library, see Appendix E, "Custom-conversion functions," on page E-1.

Platform: UNIX Only

Discarded *num_bytes* null bytes from end of tape device *device_name*.

Explanation: 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.

User response: If necessary, manually enter the discarded data.

Environment variable *variable_name* expansion would overflow string.

Explanation: 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.

User response: Use a shorter default value, or expand the length of the column.

Error accepting socket connection: *errno=operating-system_error_num*.

Explanation: A TCP socket cannot be accessed.

User response: See your **errno.h** file.

Error accessing *file_name*.

Explanation: An error occurred when **onpload** attempted to open the load or unload file.

User response: Check that the file exists. Check the file for permissions.

Error accessing format: SQL error *error_num*, ISAM error *error_num*.

Explanation: 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.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing map *map_name*: SQL error *error_num*, ISAMerror *error_num*.

Explanation: The requested map for the load or unload does not exist, or a problem exists with the **onpload** database.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing sysmaster: SQL error *error_num*, ISAMerror *error_num*.

Explanation: An access error occurred on the sysmaster database on the target server where **onpload** attempted to perform the load or unload job.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error accessing table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Explanation: The target database table cannot be accessed.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error:AIO buffer size *buffer_size* **is less than required minimum size** *size*.

Explanation: AIO buffer size is less than required size.

User response: Increase the specified buffer size in the PLCONFIG file.

Error *error_num* **closing current database.**

Explanation: A server error occurred when **onpload** closed the **onpload** or target database.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *operating-system_error_num* **closing file** *file_name*.

Explanation: An error occurred when **onpload** closed the load or unload file.

User response: See your **errno.h** file

Error *error_num* **converting record field** *field_name* **to column** *column_name*.

Explanation: A conversion error occurred when **onpload** attempted to convert the record data to the database column type.

User response: 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.

Explanation: Cannot access information about the load table.

User response: Check the validity of the table in the target database.

Error declaring cursor: SQL error *error_num*, ISAM error *error_num*.

Explanation: The **onpload** utility is unable to use the autogenerated formats and maps to create entries in a table in the **onpload** database.

User response: 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*.

Explanation: The unload query cannot be processed.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* **initializing backend connection.**

Explanation: An internal error occurred in **onpload**. Probably the server went down.

User response: Note the circumstances and contact Technical Support.

Error inserting into table *table_name*: SQL error *error_num*, ISAM error *error_num*.

Explanation: The **onpload** utility is unable to use the autogenerated formats and maps to create entries in a table in the **onpload** database.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error listening for socket connection: **t_errno**=*t_error_num* **errno**=*operating-system_error_num*.

Explanation: An error occurred listening on a Socket connection.

User response: See your **errno.h** file.

Error listening for TLI connection: **t_errno**=*t_error_num* **errno**=*UNIX_error_num*.

Explanation: An error occurred listening on a TLI connection.

User response: See **/usr/include/tiuser.h** (*t_error_num*).

Platform: UNIX Only

Error *error_num* **on record** *record_num* **converting column** *column_name* **to record field** *field_name*.

Explanation: A conversion error occurred when **onpload** attempted to convert the column data to the record field type.

User response: 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*.

Explanation: A conversion error occurred in a record.

User response: No action is required.

Error on close of server load session: SQL error *error_num*, ISAM error *error_num*.

Explanation: An internal error occurred in **onpload**. Probably the server went down.

User response: Note the circumstances and contact Technical Support.

Error opening cursor: SQL Error *error_num*.

Explanation: An error occurred when **onpload** attempted to set up an insert cursor on the load table.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error preparing query: SQL error *error_num*.

Explanation: The unload query cannot be processed.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error preparing statement *statement_name*: **SQL error** *error_num*, **ISAM error** *error_num*.

Explanation: An internal error occurred when **onpload** attempted to access the **onpload** database.

User response: 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*.

Explanation: The unload query cannot be processed.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* **reading message queue.**

Explanation: 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.

User response: 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 TLI/socket connection.**

Explanation: An error occurred reading a socket or TLI connection, based on the type of connection specified in ONCONFIG file.

User response: See your **errno.h** file.

Error *error_num* **setting isolation level.**

Explanation: An access error occurred when **onpload** attempted to set the isolation level for an unload job.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Error *error_num* **writing message on message queue.**

Explanation: 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.

User response: 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 TLI/socket connection.**

Explanation: An error occurred writing a socket or TLI connection, based on the type of connection specified in ONCONFIG file.

User response: See your **errno.h** file.

Error: Stream buffer size *buffer_size* **is less than required minimum size** *size*.

Explanation: Stream buffer size is less than required size.

User response: Increase the specified buffer size in the PLCONFIG file.

Exhausted all attempts to allocate shared-memory key.

Explanation: All the shared-memory keys in the key range tried by **onpload** are currently allocated.

User response: Wait until another **onpload** session finishes. If the problem persists, contact Technical Support.

Fatal error: cannot execute *pipe_name*.

Explanation: An attempt to run the PIPE type device in the device array failed.

User response: Make sure the PIPE entry in the device array is a valid, executable program. Pipes are only supported on UNIX.

Fatal error - cannot load X resource

Explanation: This is an internal error.

Fatal error creating server load session: error *error_num*.

Explanation: Cannot start the load session with the server.

User response: Note the circumstances and contact Technical Support.

Fatal error getting stream buffer from server.

Explanation: An internal error occurred in **onpload**. Probably the server went down.

User response: Note the circumstances and contact Technical Support.

Fatal error in server row processing: SQL error *error_num*, **ISAM error** *error_num*.

Explanation: An internal communication problem exists between the server and **onpload**.

User response: Note the circumstances and contact Technical Support.

File type device file *file_name* **is not a regular (disk) file.**

Explanation: The device array specifies that the file is a disk file, but it is not.

User response: 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.

Explanation: An internal error occurred, or a user sent an interrupt to **onpload**.

User response: If a user did not generate this interrupt, contact Technical Support.

Internal error: cannot initialize AIO library.

Explanation: This critical initialization error probably means that the UNIX kernel does not have enough shared memory or semaphores configured.

User response: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

Internal error: cannot send message.

Explanation: An internal error occurred in **onpload**. The most likely cause is a lack of shared memory.

User response: Note the circumstances and contact Technical Support.

Internal error: *error_num*. Contact Tech Support.

Explanation: A critical internal error occurred.

User response: Note the circumstances and contact Technical Support.

Internal error: invalid message type *error_num*.

Explanation: A critical internal error occurred.

User response: Note the circumstances and contact Technical Support.

Internal error *error_num* reading queue.

Explanation: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

User response: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

Invalid count detected, may be due to abnormal BE shutdown.

Explanation: The count of rows being loaded became corrupt. This message can appear if a deluxe load job aborts with an error.

User response: Contact IBM Technical Support.

Invalid code-set character: cannot convert.

Explanation: The data being loaded or unloaded has invalid character data.

User response: Make sure that you specified the correct data type on the format definition.

Invalid HEXASCII simple large object or extended type representation on record *record_num*.

Explanation: The simple large object or extended type field being loaded was classed as HEXASCII, but the data contains a non-HEXASCII character.

User response: Fix the data.

Invalid HEXASCII simple large object representation in *fieldname*, record *record_num*.

Explanation: The simple large object data field being loaded was classed as HEXASCII, but the data contains a non-HEXASCII character.

User response: Fix the data.

Invalid project name *project_name* entered.

Explanation: Incorrect project name was specified for **onpload**.

User response: Check the given project name and restart **onpload**.

Invalid reject count detected, may be due to abnormal BE shutdown. Using last known reject count and proceeding.

Explanation: The count of rows being loaded became corrupt. This message can appear if a deluxe load job aborts with an error.

User response: Contact IBM Technical Support.

Invalid session ID *id_number*.

Explanation: 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.

User response: Make sure the session ID on the command line matches the correct session ID in the **session** table.

Invalid Tape Header Expecting -> *tape_name*.

Explanation: Incorrect tape was mounted.

User response: Mount the correct tape.

Map *map_name* type is not a load map.

Explanation: Incorrect map was specified to **onpload**. You must use a load map for a load job and an unload map for an unload job.

User response: Verify that you are using the correct map type.

Method not supported by current driver.

Explanation: An internal error occurred in **onpload**.

User response: Note the circumstances and contact Technical Support.

MT cannot bind to vpid.

Explanation: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

User response: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

MT internal failure.

Explanation: This critical initialization error probably means that the operating system kernel does not have enough shared memory or semaphores configured.

User response: On UNIX, increase shared memory or semaphores. On Windows, repeat the operation.

If the condition persists, contact Technical Support.

MT failure putting CPU online.

Explanation: This critical initialization error probably means that the UNIX kernel does not have enough shared memory or semaphores configured.

User response: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

No insert permission on table *table_name*.

Explanation: You cannot load the indicated table because the DBA has not granted permission for you to do so.

User response: Make sure that you have insert permissions on the table.

No mapping to simple large object field *field_name*.

Explanation: The record format specifies a simple large object or extended type, but no column from the query is mapped to the record field.

User response: 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.**

Explanation: User tried to run **onpload** on a host computer other than the one that has the target database.

User response: Run **onpload** on the host specified in the error message.

onpload terminated by signal.

Explanation: Either an internal error occurred or a user sent **onpload** a termination signal.

User response: If the signal is not SIGKILL, SIGTERM, or SIGQUIT, note the circumstances and contact Technical Support.

Platform: UNIX Only

Pipe type device file *file_name* **is not a regular file.**

Explanation: The device array specifies that the file is a pipe (executable program) file, but it is not.

User response: 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/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

Explanation: The **onpload** utility requires that the special columns (simple large column and user-defined types) appear at the end of the select list. The **onpload** utility will reorder simple SELECT statements but is unable to re-order the select list because of expressions, aggregates, or both.

User response: Reorder 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 simple large object column *column_name*: **cannot proceed.**

Explanation: The unload query is extracting a simple large object or extended type column that is not mapped to the record field.

User response: Modify the unload query so that it does not reference the simple large object or extended type column, or map it to a field in the record format.

Query for unload is not a select query.

Explanation: The unload query does not contain a SELECT statement.

User response: 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*.

Explanation: The record size exceeds the size of the **onpload** buffers (AIOBUFSIZE). This error can occur when a delimited record contains simple-large-objects or extended types, and a format specification for a field is missing, which causes a simple large object or an extended type to be treated as a regular field.

User response: 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.

Explanation: An **onload**/server interface error occurred.

User response: Note the circumstances and contact Technical Support.

SQL error *error_num* executing statement *statement_name*.

Explanation: An internal error occurred when **onload** accessed the **onload** database.

User response: For more information, use the **finderr** or **Informix Error Messages** utility.

Simple large object or extended type conversion error occurred on record *record_num*.

Explanation: The SQLBYTE simple large object or extended type data could not be converted to HEXASCII, or the SQLTEXT simple large object or extended type has invalid character data (characters not in the code set).

User response: 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*.

Explanation: A start record was specified for the load, but fewer records are in the input file than the indicated number of records to skip.

User response: Specify the start-record number again.

Successfully loaded the shared library *library_location*.

Explanation: The shared library was loaded successfully.

User response: 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.

Explanation: After an express-mode load, a level-0 archive is needed to make the table available for update.

User response: Perform a level-0 archive.

Tables with BLOBS cannot be loaded in High Performance Mode.

Explanation: You attempted to use express mode to load a table that contains BLOB data types. The express mode does not support this condition.

User response: Perform the load in deluxe mode.

Tables with BLOBS or extended types cannot be loaded in Express mode.

Explanation: You attempted to use express mode to load a table that contains BLOB or extended data types. The express mode does not support this condition.

User response: Perform the load in deluxe mode.

Tables with simple large objects or extended types cannot be processed with no conversion (-fn).

Explanation: You attempted a no-conversion load on a table with simple large object or extended type columns. This action is not allowed.

User response: 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.*

Explanation: A tape header size is too large to fit into a memory buffer.

User response: 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.

Explanation: The device array specifies that the file is a tape device, but it is not.

User response: 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.

Explanation: The specified column has a NOT NULL constraint, but in the definition of the load map, no field is mapped to the column.

User response: Correct the load map or drop the NOT NULL constraint.

Unable to load locale categories for locale *locale_name*: error *error_num*.

Explanation: 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.

User response: 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.

Explanation: A SELECT statement contains a column name that might not be unique.

User response: 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.

Explanation: The **onpload** command-line option **-Z** enabled write to and read from the tape until the end of the device.

User response: No action is required.

Write to device (tape or pipe) *device_name* failed; no space left on device. AIO error *error_num*.

Explanation: The write to tape is failing on a tape device due to lack of space.

User response: Increase the space on the tape device or replace the device and restart the load or unload job.

HPL log-file pop-up messages

Cannot attach to server shared memory.

Explanation: If the server is on, a permissions problem exists.

User response: 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**.

Cannot create shared-memory message queue: error *error_num*.

Explanation: A critical initialization error occurred. Probably the UNIX kernel does not have enough shared memory or semaphores configured.

User response: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

Platform: UNIX Only

Cannot create shared-memory pool: errno *UNIX_error_num*.

Explanation: The operating system shared-memory system cannot be accessed.

User response: See your **errno.h** file.

Platform: UNIX Only

Cannot initialize multithreaded library.

Explanation: A critical initialization error occurred. Probably the UNIX kernel does not have enough shared memory or semaphores configured.

User response: Increase shared memory or semaphores. If the condition persists, contact Technical Support.

Platform: UNIX Only

Cannot initialize shared memory: errno *operating-system_error_num*.

Explanation: The operating system shared-memory system cannot be accessed.

User response: See your **errno.h** file.

Cannot load X resource.

Explanation: The **ipload** utility attempted to display a full-color splash screen image but another process was already using the resources needed for color display.

User response: Use the **ipload -n** option, which does not display a splash screen.

Platform: UNIX Only

Cannot open.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Explanation: An internal error occurred when **onpload** attempted to open the load or unload file.

User response: 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*.

Explanation: The log file for the job cannot be opened.

User response: See your **errno.h** file.

Cannot start I/O.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Explanation: An internal error occurred when **onpload** attempted to open the load or unload file.

User response: 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.

Explanation: 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.

User response: 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.

Explanation: Upgrade of **onpload** database failed.

User response: Look in **\$INFORMIXDIR/etc/conpload.out** for information on conversion errors.

Press 'r' when ready, 'c' to shutdown device or 'q' to quit.

Explanation: The tape device is full.

User response: 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.

Explanation: The full absolute path of the **ipldd11a.so** shared library is not set.

User response: 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.

Explanation: The express mode cannot load tables that contain BLOB data types.

User response: Use Deluxe mode.

Write error.

Enter (r)etry, (c)ontinue, (q)uit job when ready

Explanation: An internal error occurred when **onpload** attempted to open the load or unload file.

User response: 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.

Appendix H. 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-1.
- 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-3.

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 5-5.
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, “An example of a custom driver” on page H-6. 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-9 and “Available API support functions” on page H-10.

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-1.
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 **ipldd11a.SOLIBSUFFIX** shared-library file with your custom-code files. (SOLIBSUFFIX is the shared-library suffix for your operating system.)

After you rebuild **ipldd11a.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 = ipldd11a.${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
#

SO0BJS = plcsrdrv.o plcsrcnv.o your_custom_driver.o
your_custom_function.o

soilib: $(SO0BJS)
        $(LD) -o $(PLCDLIBSO) $(LDSOFLAGS) $(SO0BJS)

##### end of makefile #####

```

2. Include the following files in the makefile.

File	Description
plcstdrv.c	Prepares the custom driver tables
plcstcnv.c	Prepares the custom-conversion function tables
<i>your_custom_driver.c</i>	Contains your user-specific driver functions
<i>your_custom_functions.c</i>	Contains your user-specific conversion functions

Appendix E, “Custom-conversion functions,” on page E-1 describes the **plcstcnv.c** and the *your_custom_functions.c* files.

3. Run **make** using the makefile.

The makefile builds the shared-library file **ipldd11a.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 **ipldd11a.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 **ipldd11a**, look for **ipldd11b** or **ipldd11c**.

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-10. 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 run 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 **onload** to run less efficiently.

An example of a custom driver

The example in this topic shows how to code a custom driver that completely 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 **onload** 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 ran successfully.
PL_RTN_FAIL	Method did not succeed. Stop processing.
PL_RTN_EOF	Method reached the end of the file.

The PL_MTH_OPEN function

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 path name to open
Output arguments:	None	
Return values:	PL_RTN_OK, PL_RTN_FAIL	

The PL_MTH_CLOSE function

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.

The pl_inherit_methods(driver, methodtable) function

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-5.

The `pl_set_method_function(methodtable, method, function)` function

This function inserts the passed function into the method chain.

Function Information		Description of Arguments
Function type:	int	
Input arguments:	void *methodtable int method int (*funcptr)()	Method table passed to PL_MTH_CONFIGURE Method ID Function to insert into method chain
Return values:	PL_RTN_OK, PL_RTN_FAIL	

The `pl_driver_inherit(method)` function

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.

The `pl_get_recordlength()` function

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 PL_RTN_FAIL	Function succeeded Function failed

The `pl_set_informix_conversion(flag)` function

When this function is set to 1, it disables data conversion during the load. The data must be formatted in Informix format (as opposed to another type of format that would have to be converted before loading the data into the database).

When this function is set to 0, data conversion is enabled.

Function Information		Description of Arguments
Function type:	int	
Input argument:	int flag	1 = disable, 0 = enable
Return values:	PL_RTN_OK, PL_RTN_FAIL	

The `pl_lock_globals()` function

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	

The `pl_reset_inherit_chain(method)` function

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	

Appendix I. Running load and unload jobs on a Windows computer

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.

If you use the **onpladm** utility to start a project on Windows, the utility starts each **onpload** job in a new command window because, by default, the `INTERACTIVE_DESKTOP_OFF` environment variable is set to 0. If you want the jobs to start in the background, set the `INTERACTIVE_DESKTOP_OFF` environment variable to 1.

The setting of the `INTERACTIVE_DESKTOP_OFF` environment variable influences the behavior of all stored procedures in an instance.

For general information on the **onpladm** utility, see Chapter 17, "The onpladm utility," on page 17-1.

Running the run job or run project commands

To run 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 Development Kit. For more information on the **SETNET32** utility, see the *IBM Informix Client Products Installation Guide*.

If you do not enter the user name 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 -r1 -fb" in SPL routine cannot be executed because user
"username" did not connect with a password.
```

To run other commands with the **onpladm** utility, you do not have to enter a user name and password.

Running onpladm on UNIX with the database server running on Windows

You can run 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 run the 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," on page 17-1.

8. Prepare load and unload jobs with the **onpladm** utility.

For more information, see "Creating onpladm jobs" on page 17-4.

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 with the **ipload** utility on Windows computers

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 path name 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
```

WINservername Name of the database server on the Windows computer

WINname Name of the Windows computer

servicename 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: Onpload 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.

Appendix J. Conversion and reversion scripts for HPL database migration

When you convert or revert to different versions of the database server, you can use conversion and reversion scripts to manually upgrade or revert your **onpload** database. You must use these scripts if you are required to upgrade between the same server versions, for example, between Versions 9.40.UC1 to 9.40.UC3.

Alternatively, you can use the `IFX_ONPLOAD_AUTO_UPGRADE` environment variable with the **ipload** or **onpladm** utilities to automatically upgrade the **onpload** database the first time that you invoke an HPL utility after you migrate to a new version of the database server. You cannot use the `IFX_ONPLOAD_AUTO_UPGRADE` environment variable with the **onpload** utility.

If you invoke an HPL utility before upgrading the **onpload** database, you receive the following error, which informs you that the **onpload** database must be converted:

```
Incorrect database version. Make sure that it is upgraded properly. To
upgrade please run the $INFORMIXDIR/etc/conv/conpload.sh script manually.
For automatic upgrade please set the IFX_ONPLOAD_AUTO_UPGRADE variable.
```

Upgrading the High-Performance Loader onpload database

When you upgrade to a new version of the database server, you must also upgrade the **onpload** database.

If you are upgrading from a version of the database server that is prior to Version 9.40, you must run a conversion script. For instructions for running this script, see your *IBM Informix Migration Guide*.

Starting with Version 9.40.xC3, the database server has a new version of the **onpload** database with longer column lengths. The **onpload** database now requires slightly more disk space than it did in previous versions. The following table shows the relationship between the database server version and the **onpload** database schema version.

Database Server Version	onpload Database Version
Pre 7.3	0 or not available
7.31	1
9.2, 9.3, 9.40.xC1, 9.40.xC2	2
9.40.xC3 and later	3

Reverting from the current onpload database

Starting with Version 10.00 of the database server, reversion of the **onpload** database is automatic.

Appendix K. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility Features

The following list includes the major accessibility features in IBM Informix. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Tip: The IBM Informix Information Center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard Navigation

This product uses standard Microsoft Windows navigation keys.

Related Accessibility Information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software. The syntax diagrams in our publications are available in dotted decimal format. For more information about the dotted decimal format, go to "Dotted Decimal Syntax Diagrams."

You can view the publications for IBM Informix in Adobe Portable Document Format (PDF) using the Adobe Acrobat Reader.

IBM and Accessibility

See the *IBM Accessibility Center* at <http://www.ibm.com/able> for more information about the commitment that IBM has to accessibility.

Dotted Decimal Syntax Diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines

2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area or you can include none. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line 6.1+ data-area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. As for the * symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

.flt file 15-5

A

Accessibility K-1
 dotted decimal format of syntax diagrams K-1
 keyboard K-1
 shortcut keys K-1
 syntax diagrams, reading in a screen reader K-1
Active Job window 2-19, 2-20, 11-5, 12-6, 13-4, 13-5, 13-9
AIO error code 27 12-6
AIOBUFFERS configuration parameter 15-12, B-2
 affecting onpload processes 15-7
 defined B-2
 example 15-9
AIOBUFSIZE configuration parameter 15-12, B-2
 affecting onpload process 15-7
 example 15-9
Alter table schema 15-8
ALTER TABLE statement
 format to use 13-6
Assigning records to devices 6-1
Autogenerate Unload Components window 2-24, 13-2, 13-3

B

Binary data
 format 5-6
Binary type, of computer A-6
Block size of tape 16-7, 17-6, 17-39
Browse option
 defined 14-2
 log file 14-6
Browsers menu
 defined 3-3
 Logfile option 14-5
 Record option 14-1
 Violations option 14-4
Buffer size
 I/O, with onpload 16-7
 server stream buffer 16-8
BYTE data
 amount to transfer 9-10
 order of binary information 5-3, 5-6
 size of data type 7-6
 size of variables 5-7
 specification of order A-6

C

cadiload threads 1-10
Carriage returns, in fixed format 7-7
Case conversion 9-9, 16-10, A-12
Changing
 unload job options 12-8
CHAR data type 15-11
Character
 case conversion 16-10

Character (*continued*)
 set, modifying 7-19
COBOL format 7-17
 byte
 setting with onpladm 17-8, 17-21, 17-40
 used with generate 13-7
 creating 7-16
 definition window 7-16
 records 7-15
 setting with onpladm 17-8, 17-21, 17-40
 used with generate 13-7
Code sets
 conversion 10-7
 data file 5-3
 database 5-3
 defaults table A-1
 delimited formats 7-19
 Fixed and COBOL formats 7-18
 Global Language Support (GLS) 5-3, 7-18, 7-19, 10-7
 modify 7-19
 setting with onpladm 17-26, 17-27, 17-28, 17-41
Column Selection window 8-4, 8-5
Columns
 characteristics 9-12
 default values 9-10
 drop, add, modify 15-8
 offset, in mapping options 9-10
Commit
 interval
 load job 12-8
 onpload database A-12
Components menu
 defined 3-3
 devices 6-2
 filter 10-2
 formats 7-2
 generate 13-8
 maps 9-4, 9-5
 query 8-1, 8-7
Computer
 configuration, reorganize 15-8
 modifying 5-6
Configuration file
 conventions B-1
 HPL 16-11, B-1
 onpload 1-7
Configuration parameters
 AIOBUFFERS B-2
 AIOBUFSIZE B-2
 CONVERTTHREADS B-2
 CONVERTVPS B-3
 HPL parameters defined B-1
 HPL_DYNAMIC_LIB_PATH B-4, H-3
 HPLAPIVERSION B-3
 STRMBUFFERS B-4
 STRMBUFFERSIZE B-4
 thread control 15-6
Configure menu 3-3
Configuring
 ipload utility 5-1
Confirm delete window 3-14

- Confirmation window
 - file overwrite 8-9
- Connect Server window 5-1, 5-2
- Constraints
 - checking 15-3
 - table-level referential ones when using onpladm 17-16
 - violations 15-5
- Conversion functions, custom E-1
- Converter threads 1-10, 15-11, 16-11
- CONVERTTHREADS configuration parameter B-2
 - affecting onpload process 15-7
 - example 15-9
- CONVERTVPS configuration parameter B-3
 - affecting onpload process 15-7
 - example 15-9
- Copy Data window
 - illustration 3-13
 - using 3-12
- Copy existing format 3-12
- cron
 - job 11-2, 12-2
- Custom conversion functions E-1
- Custom driver
 - adding 5-5
 - creating H-1
- Custom file handling software 5-4
- Custom input to pipes 5-4

D

- Data
 - unload using onpload 16-5
- Data conversion
 - onpload 16-5
- Data file
 - formats supported by ipload 7-2
 - structure 7-1
- Data masking 9-10
- Data source, for onpload 16-5
- Data types
 - COBOL 7-15
 - fixed format 7-5
 - values in onpload database A-5
- Database servers
 - limitations 12-1
 - listing attributes with onpladm 17-41
 - selecting 5-1
 - setting attributes with onpladm 17-40
 - target server 11-2, 12-1
- Database Views window 8-10
- Databases
 - create for example 2-2
 - creating project with onpladm 17-38
 - name, override in onpload 16-10
 - unloading records 11-2
- DATE data type 15-11
- DB_LOCALE environment variable 5-3
- DBDELIMITER environment variable 7-20
- DBONPLOAD environment variable 1-7, 5-2
 - defined 1-7
- Debugging information A-11
- DECIMAL data type 15-11
- Decimals 7-6
- Defaults
 - name of log file 14-5
 - name of rejects file 14-5
 - values, example 2-3
- defaults table, in onpload database A-1
- Defaults window 3-10, 5-3
- Define format
 - delimited records 7-12, 7-16
 - editing a format 7-6
 - fixed-length records 7-2
 - modifying a format 7-18
- Definition window 7-17
- Delete existing format 3-13
- Delimited format
 - creating 7-12, 7-16
 - in formats table A-6
 - modifying 7-19
 - setting with onpladm 17-8, 17-21, 17-40
 - using a .csv file 7-20
 - using simple large objects 7-13
- Delimited Format window 2-11, 7-14, 7-15
- Delimited record, defined 7-12
- Delimiter characters
 - changing 7-19
 - defined 7-12
- Delimiter Options window 7-20
- delimiters table, in onpload database A-1
- Deluxe mode
 - choosing from load options 12-8
 - compare to express 15-4
 - defined 15-2
 - in High-Performance Loader, defined 1-4
 - INSERT statements 15-2
 - mentioned 15-2
 - without replication 15-2
- Device array
 - defined 6-1
 - device types 6-1, 6-3
 - editing 6-4
 - elements of, in onpload database A-2
 - example 2-7
 - improving performance 15-8
 - onpladm
 - creating 17-17
 - deleting 17-19
 - describing 17-18
 - listing all in a project 17-18
 - modifying 17-18
 - using 17-6, 17-12
 - onpload
 - using 16-5
 - speed 15-8
 - tape parameters 6-3
- Device Array Definition window 2-8, 3-5, 6-3, 6-4
- Device Array Selection window 2-7, 3-4, 6-2
- device table of onpload database A-2
- Dirty Read isolation level 11-7
- Disabilities, visual
 - reading syntax diagrams K-1
- Disability K-1
- Dotted decimal format of syntax diagrams K-1
- Driver
 - class, format type 5-5
 - creating custom driver H-1
 - modifying 7-19
 - name of custom driver 5-5
- Drivers window
 - using 5-5, 5-7

E

- EBCDIC data, generating 13-7
- Environment variables
 - DB_LOCALE 5-3
 - DBDELIMITER 7-20
 - DBONPLOAD 1-7
 - IFX_ONPLOAD_AUTO_UPGRADE 1-7, J-1
 - INFORMIXDIR 1-6
 - INFORMIXSERVER 1-6
 - LD_LIBRARY_PATH 1-6
 - ONCONFIG 1-6
 - PDQPRIORITY 1-7
 - PLCONFIG 1-7
 - PLOAD_LO_PATH 1-7, 1-8
 - PLOAD_SHMAT 1-7
 - PLOAD_SHMBASE 1-7, 1-8
- Error code 27 12-6
- Errors
 - constraint violations 15-6
 - maximum number allowed 11-7, 12-8, 16-8
 - onpladm utility 17-4
- Exporting a query 8-8
- Express mode 15-2
 - choosing from load options 12-8
 - compare to deluxe 15-4
 - foreign key constraints 15-4
 - in High-Performance Loader, defined 1-4
 - level-0 backup 12-6
 - load example 15-10
 - page size limitation 15-3
 - sequence of events 15-3

F

- Fast format 7-18
- Fast job
 - defined 7-18
- Fast Job Startup window 13-9
- Field
 - set minimum and/or maximum 9-10
 - set offset 9-10
- Figure
 - extracting data from a table 8-1
 - foreign-key constraints 15-4
 - load and unload modes 15-2
 - maps found by a search 9-15
 - view indicator 9-11
- File descriptor, COBOL 7-17
- Files
 - .flt 15-5
 - COBOL 7-15
 - configuration for HPL B-1
 - default onpload configuration 1-7
 - import/export queries 8-8
 - ipldd11a.so B-4, E-3, H-3
 - onpload.std 1-7
 - path name for I/O 16-3
 - plconfig 16-11
- Fill character, mapping options 9-10
- Filter
 - conversion of code set 10-7
 - creating 10-2
 - defined 10-1
 - editing 10-5, 10-6
 - example 10-1
 - match conditions 10-4

- Filter (*continued*)
 - mode, with constraints 15-6
 - onpladm
 - creating 17-32
 - deleting 17-34
 - describing 17-33
 - listing all in project 17-33
 - modifying 17-33
 - onpload database A-3
 - rejected records 15-5
- Filter Views window 10-7
- filteritem table, in onpload database A-3
- filters table, in onpload database A-3
- Filters window 10-3
- Find button 9-11
- Find Node window 9-12
- Fixed binary format
 - format type group 13-7
 - setting with onpladm 17-8, 17-21, 17-40
- Fixed format
 - data types 7-5
 - defined 7-2
 - in formats table A-6
 - setting with onpladm 17-8, 17-21, 17-40
 - using carriage returns 7-7
 - using simple large objects 7-8, 7-10
- Fixed Format Definition window 7-4, 7-8, 7-9, 7-11
- Fixed Format edit window 7-3
- Fixed Format Options window 7-19
- Fixed internal format
 - mentioned 7-18
 - setting with onpladm 17-8, 17-21, 17-40
 - used in generate 13-6
- Fixed length 7-2
- FLOAT data type 15-11
- Foreign key constraint 15-4
- Format
 - COBOL 7-15, 13-7
 - copy 3-12
 - create 7-2
 - defined 7-1
 - delete 3-13
 - delimited 7-2
 - fast job 7-18
 - fast, defined 7-18
 - file structure supported in HPL 7-2
 - fixed internal 13-6
 - onpladm
 - creating 17-26
 - deleting 17-29
 - describing 17-28
 - listing all in project 17-29
 - modifying 17-28
 - performance 15-10
 - steps for editing 7-6, 7-7
 - testing 14-1
 - types supported by HPL 7-2
 - types used by generate 13-6
- Format Views window 2-9, 3-9, 7-22
- formatitem table, in onpload database A-3
- formats table, in onpload database A-6
- Fragmenter threads 1-10
- Functions
 - custom conversion E-1
 - user-defined in mapping options 9-10

G

Generate

- assumptions 13-7
- defined 13-1
- EBCDIC data 13-7
- format types 13-6
- from unload job 13-3
- no-conversion job 13-8
- types of load and unload tasks 13-1

Generate window 13-6, 13-8

Global Language Support (GLS) 5-3, 7-18, 7-19, 10-7

H

Help

- High-Performance Loader menu description 3-3
- using online help 3-18

High-Performance Loader

- components 1-4
- configuration file 16-11, B-1
- configuring it 5-1
- data-load process 1-2
- data-unload process 1-3
- features 1-1
- ipload utility 1-5, 2-2, 3-1, 4-1, I-3
- main window 2-3
- managing 15-1
- modes 1-4, 2-18, 15-2
- onpladm utility 1-5, 17-2, I-1
- onpload database 1-5, 4-1, 5-2, A-1, J-1
- onpload utility 1-4, 1-8, 5-4, 16-1
- overview 1-1
- prerequisites 1-6
- scripts for database migration J-1
- usage tasks 15-8

HPL_DYNAMIC_LIB_PATH configuration parameter B-4, H-3

HPLAPIVERSION configuration parameter B-3

I

I/O

- buffer size 16-7
- number of tapes to load 16-9
- tape block size 16-7

IFX_ONPLOAD_AUTO_UPGRADE environment variable 1-7, J-1

Import/Export File Selection window 8-8

Importing query 8-8

INFORMIXDIR environment variable 1-6

INFORMIXSERVER environment variable 1-6

Input

- starting record 16-9

INSERT statements 15-2

INT data type 15-11

INTERACTIVE_DESKTOP_OFF I-1

Internal format

- limitations 7-18
- use with generate 13-8

ipldd11a.so file B-4, E-3, H-3

ipload utility

- configuring 5-1
- creating a project 4-1, 4-4
- defined 1-1, 1-5
- GUI 3-2
- Projects window 4-3

ipload utility (*continued*)

- purpose 1-5
- starting 2-2, 3-1
- using 2-2, 3-2

Isolation level

- committed 11-7
- cursor stability 11-7
- Repeatable Read 11-7
- setting with onpladm 17-10
- unload option 11-7

J

Jobs

- conversion 17-4
- menu, defined 3-3
- no-conversion 17-4

Justification of data in mapping options 9-9

L

language table, in onpload database A-6

LD_LIBRARY_PATH environment variable 1-6

Level-0 backup

- express mode 2-20, 15-4

Limitations, database server 12-1

Load and unload session

- maximum errors 16-8

Load data

- onpladm 17-7, 17-12
- onpload 16-5

Load job

- changing options 12-8
- commit interval 12-8
- components 12-1
- creating 12-3, 12-4
- defined 12-1
- device-array speed 15-8
- editing 12-9
- example 2-4
- from the command line 16-1
- generate violations records 12-8
- log file 14-5
- maximum errors 12-8
- mode options 12-8
- multiple jobs 11-2, 12-2
- number of records 12-8

onpladm

- conversion 17-5
- creating for every table 17-38
- creating no-conversion 17-10
- deleting 17-16
- describing 17-14
- listing all in project 17-14
- modifying 17-13
- running 17-15
- running on Windows I-1
- onpload database A-10
- preview records 14-1
- run example 2-19, 2-25
- running 12-6, 12-9
- server considerations 11-2, 12-1
- starting record 12-8
- status log 12-5
- tapes, number of 12-8
- using cron 11-2, 12-2

- Load Job Select window
 - command line information 12-7
 - illustration 2-5, 12-4
- Load Job window 2-6, 2-12, 2-18, 3-7, 12-5
- Load log
 - examining 14-5
- Load map
 - how to create 9-3
- Load Options window 2-19, 12-9
- Load Record Maps window 2-14
- Log files
 - created by ipload 12-3
 - for load job 14-5
 - messages 12-6, G-1
 - sample entry 14-6
 - setting 11-4
- Lowercase conversion 9-9, 16-10

M

- Machine type
 - High-Performance Loader default 5-3
 - modifying 5-6, 7-19
 - onpladm
 - creating 17-36
 - deleting 17-37
 - describing 17-37
 - listing all 17-37
 - modifying 17-36
- machines table, in onpload database A-6
- Machines window
 - illustration 5-6
 - using 5-7
- Map
 - columns and fields of same name 9-3, 9-6
 - defined 9-1
 - in-line blobs 7-9
 - onpladm
 - creating 17-19
 - deleting 17-24
 - describing 17-24
 - listing all in project 17-25
 - modifying 17-25
- Map Views window 2-13, 9-13, 9-14
- Map-definition window 2-15, 2-16, 2-17, 9-2, 9-5, 9-7, 9-8, 9-11
- Map-edit window
 - defined 9-3
 - purpose 9-3
 - using the find button 9-11
- mapitem table, in onpload database A-7
- mapoption table, in onpload database A-7
- Mapping options
 - bytes to transfer 9-10
 - case conversion 9-9
 - column offset 9-10
 - default column value 9-10
 - defining 9-8
 - field minimum and/or maximum 9-10
 - field offset 9-10
 - fill character 9-10
 - function, user-defined 9-10
 - justification 9-9
 - picture format 9-10
 - steps to define 9-8
 - summary 9-8
 - symbol 9-9
- Mapping Options window 9-9

- maps table, in onpload database A-8
- Masking data 9-10
- Match conditions
 - defined 10-1
 - filter 10-4
 - WHERE clause 8-7
- Maximum
 - number of errors 11-7
- Message log
 - categories of messages G-2
- Message log file
 - path name in onpload 16-11
- Message window 3-10
- Mode options, load job 12-8
- Modify format
 - COBOL 7-18
 - delimited 7-19
 - fixed 7-18
- MONEY data type 15-11
- Multiple load or unload jobs 11-2, 12-2

N

- No-conversion job
 - changing computer configuration 15-8
 - defined 7-18
 - load example 15-9
 - option 13-8
 - restrictions 7-18
 - run fast job 13-8
 - running the job 13-8
 - using onpladm 17-4
 - using onpload 16-5
 - with hidden columns 15-9
- Non-printable field delimiter values A-1
- NOT NULL violation 15-5
- Notes window 3-14
- null UDT data 7-5
- Number of conversion threads 16-11
- Number of records in a load job 12-8
- Number of records to process
 - assigned in onpload 16-8
- Number of tapes to load 16-9

O

- ONCONFIG environment variable 1-6
- Online help
 - High-Performance Loader menu 3-3
 - how to use 3-18
- onpladm command
 - B option 17-6
 - f option 17-6
 - F option 17-3
 - M option 17-6, 17-39
 - n option 17-6
 - P option 17-39
 - R option 17-14
 - S option 17-3, 17-6
 - T option 17-6
 - z option 17-7, 17-20, 17-39
 - Z option 17-16
 - setting format type 17-7, 17-20, 17-39
 - setting run mode 17-6
 - specification file syntax 17-3

- onpladm utility
 - creating no-conversion jobs 17-10
 - creating onpload database 17-2
 - database project, creating 17-38
 - defined 1-1, 1-5, 17-2
 - defining jobs 17-4
 - device arrays, defining 17-17
 - error handling 17-4
 - features of 17-2
 - filters, defining 17-32
 - formats, defining 17-26
 - machine types, defining 17-36
 - maps, defining 17-19
 - projects, defining 17-34
 - queries, defining 17-30
 - specification file conventions 17-3
 - target server attributes, listing 17-41
 - target server attributes, setting 17-40
 - using on Windows I-1
- Onpladm utility
 - using when tables have constraints 17-16
- onpload command
 - d option 16-6
 - i option 16-9
 - M option 15-11
 - Z option 16-4
 - generated for Load Job 12-7
 - generated for Unload Job 11-6
 - starting 16-2
 - syntax 16-2
 - usage 16-1
 - using with collection columns 16-12
- onpload database
 - connection to 5-1
 - creating 5-2
 - creating with onpladm 17-2
 - creation of 2-3
 - default settings 5-2
 - defined 1-1, 1-5
 - multiple 1-7
 - purpose of 4-1
 - reverting J-1
 - select server 5-1
 - table
 - defaults A-1
 - delimiters A-1
 - device A-2
 - filteritem A-3
 - filters A-3
 - formatitem A-3
 - formats A-6
 - language A-6
 - machines A-6
 - mapitem A-7
 - mapoption A-7
 - maps A-8
 - progress A-9
 - query A-10
 - session A-10
 - table descriptions A-1
 - upgrading J-1
- onpload utility
 - architecture 1-8
 - configuration file 1-7
 - defined 1-1, 1-4, 16-1
 - deluxe mode 1-9
 - express mode 1-10

- onpload utility (*continued*)
 - filename size limits on UNIX 16-1
 - flag for a fast job 7-18
 - load data 16-5
 - specifying defaults 5-4
 - starting 16-2
 - syntax 16-2
 - using with collection columns 16-12
- onpload.std file 1-7
- onstat utility 15-7
- Options
 - load job 12-8
 - unload job 11-7
- Options symbol 9-9
- Organization of a Record that Includes In-Line TEXT Data 7-10

P

- Page size in express mode 15-3
- PDQPRIORITY environment variable 1-7, 15-11
- Performance
 - converter threads 15-11
 - hints for HPL 15-10
 - improving in HPL 15-6
 - VPs 15-11
- Picture description, COBOL 7-17
- Picture format, in mapping options 9-10
- Pipe
 - device arrays 5-4
 - in a device array 6-3
 - starting command 6-1
 - use with onpladm 17-7, 17-12
 - use with onpload 16-5
- pl_wkr threads 1-10
- plconfig configuration file
 - HPL value override 16-11
- PLCONFIG environment variable 1-7
- PLCONFIG file
 - override I/O buffer size 16-7, 16-8, 16-11
- PLOAD_LO_PATH environment variable 1-7, 1-8
- PLOAD_SHMAT environment variable 1-7, 1-8
- PLOAD_SHMBASE
 - avoiding shared memory collision 1-8
- PLOAD_SHMBASE environment variable 1-7, 1-8
- Print button 3-15
- Privileges 12-2
- Problems during a load job 11-5, 12-6
- Project name, in onpload 16-3, 17-29, 17-34
- project table, in onpload database A-9
- Projects
 - creating a new project 4-4
 - onpladm
 - creating 17-34
 - creating for all tables 17-38
 - deleting 17-35
 - listing all 17-35
 - running all jobs 17-34
- Projects window 4-3
- Proper-name case conversion 16-10
- Proper-name conversion 9-9

Q

- Query
 - defined 8-1

- Query (*continued*)
 - export to a file 8-8, 8-9
 - for unload map 9-5
 - import from a file 8-8
 - onpladm
 - creating 17-30
 - deleting 17-31
 - describing 17-30
 - listing all in project 17-31
 - modifying 17-30
 - steps for defining 8-1
 - using the Table button 8-3
 - query table, in onpload database A-10
 - Query window 8-2
 - Query-definition window 8-3, 8-6, 8-7
 - Quiet, suppress output A-11

R

- Raw load and unload 7-18, 13-8
- Reading to end of device 12-6, 16-4, 17-16
- Record Browser window 14-2
- Record filter 16-10
- Record Formats window 2-10, 3-15, 7-3
- Record map, assigned by onpload 16-3
- Records, number to process 16-8
- Rejected records 15-5
 - conversion errors 15-5
 - filter conditions 15-5
 - reviewing 14-3
- Reorganize computer configuration 15-8
- Repeatable read isolation level 11-7

S

- Schema, of database table 13-1
- Screen reader
 - reading syntax diagrams K-1
- sdriver threads 1-10
- SELECT clause, preparing 8-2
- Server name, High-Performance Loader default 5-3
- session table, in onpload database A-10
- SET CONSTRAINTS statement
 - DISABLED 15-4
 - ON 15-6
- setrw threads 1-10
- Shortcut keys
 - keyboard K-1
- Simple large objects
 - as inline data 7-8, 7-10
 - BYTE data type 15-3
 - in delimited records 7-13
 - in fixed format 7-8, 7-10
 - in separate files 7-9, 7-11
 - TEXT and BYTE 15-3
- Single CPU, performance B-3
- Smart large objects 15-3
- SMFLOAT data type 15-11
- Specification file for onpladm
 - conventions 17-3
 - conversion job, creating 17-8
 - device arrays, creating 17-17
 - device arrays, modifying 17-18
 - filters, creating 17-32
 - filters, modifying 17-33
 - formats, creating 17-26

- Specification file for onpladm (*continued*)
 - formats, modifying 17-28
 - jobs, modifying 17-13
 - machine types, creating 17-36
 - machine types, deleting 17-37
 - machine types, modifying 17-36
 - maps, creating 17-21
 - maps, modifying 17-25
 - no-conversion jobs, creating 17-12
 - queries, creating 17-30
 - queries, modifying 17-30
 - syntax 17-3
- Specifications window 9-12
- Specs button 3-11, 9-12
- SQL statements, use in HPL 8-1
- sqlhosts file 5-1
- Start record
 - input 16-9
 - load job 12-8
- Stream threads 1-10
- STRMBUFFERS configuration parameter 15-12, B-4
 - example 15-9
- STRMBUFFERS parameter
 - affecting onpload process 15-7
- STRMBUFFERSIZE configuration parameter 15-12, B-4
 - affecting onpload process 15-7
 - example 15-9
- Suppress message output A-11
- Swap bytes 16-7, A-11
- Symbol, mapping options 9-9
- Synonyms 8-3, 9-4
- Syntax
 - onpload utility 16-2
- Syntax diagrams
 - reading in a screen reader K-1

T

- Table
 - create for example 2-2
- Tape I/O threads 1-9
- Tape parameters, specifying 6-3
- Tape size
 - setting 6-3
- Tapes
 - block size 16-7
 - number of 12-8, A-12
 - number to load 16-9
 - reading to end 12-6, 16-4, 17-16
 - writing to end 11-6, 16-4, 17-16
- Target server 5-1, 11-2, 12-1
- Testing formats 14-1
- TEXT data type 15-3
- Threads
 - cadiload 1-10
 - convert 1-10
 - fragmenter 1-10
 - pl_wkr 1-10
 - sdriver 1-10
 - setrw 1-10
 - stream 1-10
 - tape I/O 1-9
 - ulstrm 1-12
 - unload-stream 1-12
 - worker 1-10
- Trace level A-11
- Transfer bytes, in mapping options 9-10

Troubleshooting
HPL load errors 15-5

U

ulstrm threads 1-12
Unload data
 using onpload 16-5, 17-7, 17-12
Unload job
 changing the options 11-7
 components of 11-1
 creating 11-2
 defined 11-1
 example 2-21
 from the command line 16-1
 generate option 13-3
 log file 14-5
 multiple jobs 11-2, 12-2
 onpladm
 conversion 17-5
 creating for every table 17-38
 creating no-conversion 17-10
 deleting 17-16
 describing 17-14
 listing all in project 17-14
 modifying 17-13
 running 17-15
 running on Windows I-1
 onpload database A-10
 options 11-7, 12-8
 using cron 11-2
Unload Job Select window
 command line information 11-6
 illustration 11-3
Unload Job window 2-23, 2-25, 11-4, 13-4
Unload map
 defined 9-5
 how to create 9-5
 steps to create 9-5
Unload Options window 11-7
Unload Record Maps window 9-6
Uppercase conversion 9-9, 16-10
Usage description, COBOL 7-17
Usage models for HPL 15-8

V

VARCHAR data type 15-11
View icon
 defined 3-16
View indicator, figure 9-11
Views 8-3
Violations
 when loading records from a data file 15-5
Violations table
 generate from load job 12-8
 viewing 14-4
Violations Table Browser window 14-4, 14-5
Visual disabilities
 reading syntax diagrams K-1
VPs, performance 15-11

W

WHERE clause
 match conditions 8-7

WHERE clause (*continued*)
 preparation 8-6
White space in configuration file B-1
Window
 Delimited Format definition 7-14
 Delimiter Options 7-20
 Device Array Selection 6-2
 device-array definition 6-3
 Filter Views 10-7
 Filters 10-3
 Fixed Format 7-3
 Fixed Format definition 7-8, 7-9
 Fixed Format definition window 7-4
 HPL main window 2-3
 Load Job 2-6, 2-12
 Load Job Select 2-5
 Load Options 2-19
 Machines 5-6
 map definition 2-17
 query definition 8-3
 Record Browser 14-2
 Record Format 7-3
Windows, using onpladm utility I-1
Worker threads 1-10
Writing to end of device 11-6, 16-4, 17-16



Printed in USA

SC27-3811-00



Spine information:

IBM Informix **Version 11.50**

IBM Informix High-Performance Loader User's Guide

