



IBM Informix Migration Guide



IBM Informix Migration Guide

Note:

Before using this information and the product it supports, read the information in “Notices” on page B-1

First Edition (December 2004)

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

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

© Copyright International Business Machines Corporation 1996, 2004. All rights reserved.

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

Contents

Introduction.	ix
About This Manual	ix
Types of Users	x
Software Dependencies	x
Assumptions About Your Locale	xii
Demonstration Databases	xii
New Features in Database Servers	xii
Documentation Conventions	xiii
Typographical Conventions	xiii
Feature, Product, and Platform	xiii
Syntax Diagrams	xiv
Example Code Conventions	xviii
Additional Documentation.	xix
Installation Guides	xix
Online Notes	xix
Informix Error Messages	xxi
Manuals.	xxii
Online Help	xxii
Accessibility	xxii
IBM Informix Dynamic Server Version 10.0 and CSDK Version 2.90 Documentation Set	xxii
Compliance with Industry Standards	xxv
IBM Welcomes Your Comments	xxvi

Part 1. Overview of IBM Informix Migration

Chapter 1. Database Server Migration	1-1
IBM Informix Database Server Products	1-1
Migration on the Same Operating System.	1-4
Source and Target Database Servers on UNIX	1-4
Source and Target Database Servers on Linux	1-4
Source and Target Database Server on Windows XP	1-5
Source and Target Database Servers on Windows NT	1-5
Source and Target Database Servers on Windows 2000	1-6
Source and Target Database Servers on Windows 95	1-6
Migration on Different Operating Systems	1-6
Database Server Migration Paths.	1-7
Chapter 2. Data Migration	2-1
Migrating a Database or Selected Data.	2-2
Migrating Between Different Versions of a Database Server	2-2
Changing Database Servers, Operating Systems, or GLS Locales	2-2
Distributing a Client Application.	2-2
Importing Non-IBM Informix Data	2-3
Setting Environment Variables Before Using Utilities	2-3

Choosing Data-Migration Tools	2-3
Automatic Data Migration	2-5
The onunload and onload Utilities	2-5
The dbexport and dbimport Utilities	2-7
LOAD, UNLOAD, and dbload	2-9
The dbschema Utility	2-10
External Tables	2-12
The High-Performance Loader	2-12
Nonlogging Raw Tables (IDS 9.x or Later)	2-13
Movement of TEXT and BYTE Data	2-13
Moving Data Between Computers and Dbspaces	2-14
Importing Data from a Non-IBM Informix Source.	2-14
Importing Data with IBM Informix Enterprise Gateway Products	2-14

Part 2. Migration to a Later Version of a Database Server

Chapter 3. Preparing for Migration	3-1
Preparing for Migration.	3-1
Diagnostic Information That You Need Before Upgrading	3-2
Migration Guidelines	3-3
Changes in Dynamic Server 10.0, 9.40, 9.30, and 9.2x	3-4
Storage-Manager Validation and Installation	3-35
Migrating Between 32-bit and 64-Bit Database Servers	3-36
Chapter 4. Migrating to Dynamic Server 10.0 with Enterprise Replication.	4-1
Converting to Dynamic Server 10.0 with Enterprise Replication	4-1
Converting Replication of 9.2x User-Defined Data Types.	4-2
Reverting from Dynamic Server 10.0 with Enterprise Replication	4-3
Chapter 5. Converting to Dynamic Server 10.0	5-1
Converting to Version 10.0.	5-1
Summary of Conversion Steps	5-2
Check and Configure Available Space	5-3
Save Copies of the Current Configuration Files	5-5
Close All Transactions and Shut Down the Source Database Server	5-6
Check for Any Open Transactions	5-6
Verify the Integrity of the Data	5-7
Verify the Database Server Mode	5-8
Disable High-Availability Data Replication	5-8
Make a Final Backup of the Source Database Server	5-9
Verify That the Source Database Server Is Offline	5-9
On UNIX or Linux, Modify Kernel Parameters	5-9
Install Dynamic Server 10.0	5-9
Set Environment Variables	5-10
Customize Configuration Files	5-11
Add Any Communications Support Modules (UNIX/Linux)	5-12
Install and Configure Any DataBlade Modules.	5-12
Initialize Dynamic Server 10.0	5-12
Monitor the Conversion Complete Status	5-13

Upgrade the High-Performance Loader onpload Database	5-14
For ON-Bar, Rename the sm_versions.std File	5-15
Update Statistics	5-15
Verify the Integrity of the Data	5-15
Make an Initial Backup of Dynamic Server 10.0	5-16
Tune Dynamic Server 10.0 for Performance	5-16
Enable HDR	5-16

Chapter 6. Reverting from Dynamic Server 10.0. 6-1

Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24	6-1
Determine Whether Reversion Is Possible.	6-2
Recompile Java UDRs That Have Been Compiled Using JDK 1.4.x	6-7
Revert an onpload Database to the Previous Version	6-8
Check and Configure Available Space	6-8
Save Copies of the Current Configuration Files (UNIX/Linux).	6-8
Verify the Integrity of the Data	6-8
Back Up Dynamic Server 10.0.	6-8
Remove BladeManager Extensions	6-8
Remove Version 10.0 Features.	6-9
Disable HDR	6-9
Run the Reversion Utility	6-9
Modify Configuration Parameters	6-10
Reset Environment Variables.	6-10
Remove Any Communications Support Module Settings (UNIX/Linux)	6-10
Reinstall and Start the Target Database Server	6-10
Update Statistics.	6-10
Verify the Integrity of the Data	6-11
Back Up the Target Database Server	6-11
Return the Target Database Server to Online Mode	6-11
Enable HDR	6-11

Part 3. Migration of Data Between Database Servers

Chapter 7. Migrating Between Database Servers and Operating Systems. 7-1

Choosing a Migration Method	7-1
Adjusting Database Tables for File-System Variations	7-2
Moving Data to a Database Server on a Different Operating System	7-2
Using the Migration Utilities	7-3
Adapting Your Programs for UNIX or Windows	7-3
Completing Migration	7-4
Moving Data Between Dynamic Server and Workgroup Edition on Different Operating Systems	7-4

Part 4. Data Migration Utilities

Chapter 8. The dbexport and dbimport Utilities 8-1

Syntax of the dbexport Command	8-2
Syntax of the dbimport Command	8-7
Simple Large Objects (IDS 9.x or Later)	8-14
Database Locale Changes	8-14

Chapter 9. The dbload Utility	9-1
Syntax of the dbload Command	9-1
Command File for dbload	9-5
Command File to Load Complex Data Types (IDS 9.x or Later)	9-14
Chapter 10. The dbschema Utility	10-1
Syntax of the dbschema Command	10-2
Database Schema Creation	10-3
Server-Specific Information	10-5
User-Defined and Complex Data Types (IDS 9.x or Later)	10-5
Sequence Creation	10-6
Synonym Creation	10-7
Privileges	10-7
Granting Privileges	10-8
Displaying Privilege Information for a Role.	10-8
Table, View, or Procedure Creation.	10-9
Table Information	10-9
Role Creation	10-10
Distribution Information for Tables	10-11
Example Output	10-12
Distribution Description	10-12
Distribution Information.	10-13
Overflow Information	10-13
DB-Access Input from dbschema Output	10-14
Inserting a Table into a Database Example.	10-14
Re-Creating the Schema of a Database	10-14
Chapter 11. The LOAD and UNLOAD Statements	11-1
Syntax of the UNLOAD Statement.	11-1
Syntax of the LOAD Statement	11-2
Using Load and Unload Statements for Locales That Support Multibyte Code Sets	11-2
Chapter 12. The onmode Utility	12-1
Use of the onmode -b Command for Reversion	12-1
Preparation for Reversion.	12-2
Syntax of the onmode -b Command	12-2
Chapter 13. The onunload and onload Utilities.	13-1
How onunload and onload Work	13-1
Syntax of the onunload Command.	13-2
Destination Parameters	13-3
Constraints That Affect onunload	13-3
Database or Table Unloading	13-4
Logging Mode	13-5
Locking During Unload Operation.	13-5
Syntax of the onload Command	13-5
Source Parameters	13-6
Create Options	13-7
Constraints That Affect onload	13-8
Constraints That Affect onload and onunload	13-9

Restrictions That Affect onload and onunload	13-10
Logging During Loading	13-10
Movement of Simple Large Objects to a BlobSpace	13-11
Ownership and Privileges	13-11
Exclusive Locking During Load Operation.	13-11
Steps for Using onunload and onload	13-11

Part 5. Appendixes

Appendix. Accessibility	A-1
Notices	B-1
Index	X-1

Introduction

About This Manual	ix
Types of Users	x
Software Dependencies	x
Assumptions About Your Locale	xii
Demonstration Databases	xii
New Features in Database Servers	xii
Documentation Conventions	xiii
Typographical Conventions	xiii
Feature, Product, and Platform	xiii
Syntax Diagrams	xiv
How to Read a Command-Line Syntax Diagram	xvi
Keywords and Punctuation	xvii
Identifiers and Names	xvii
Example Code Conventions	xviii
Additional Documentation.	xix
Installation Guides	xix
Online Notes	xix
Locating Online Notes	xx
Online Notes Filenames.	xxi
Informix Error Messages	xxi
Manuals.	xxii
Online Manuals	xxii
Printed Manuals	xxii
Online Help	xxii
Accessibility	xxii
IBM Informix Dynamic Server Version 10.0 and CSDK Version 2.90 Documentation Set	xxii
Compliance with Industry Standards	xxv
IBM Welcomes Your Comments	xxvi

In This Introduction

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

About This Manual

This manual describes the migration procedures to convert to Version 10.0 of IBM Informix Dynamic Server, revert to older database servers, and move data between databases, database servers, and computers. This manual also contains information on how to use the **dbexport**, **dbimport**, **dbload**, **dbschema**, **onload**, and **onunload** data-migration utilities and the **LOAD** and **UNLOAD** SQL statements.

For information on migrating to other IBM Informix database server versions, see the Migration Guide that is included in the documentation set for that version of the server.

Migration includes conversion (upgrading) to a later version of a database server, reversion to an earlier version of a database server, and movement of data between databases, database servers on the same operating system, database servers on different operating systems, and different kinds of database servers. Conversion or reversion often involves changing connectivity information in the **sqlhosts** file or registry key, host environment variables, configuration parameters, and other database server features.

Types of Users

This manual is for the following users:

- Database users
- Database administrators
- Database server administrators
- System administrators
- Database-application programmers
- Backup operators
- Performance engineers

To understand this manual, you need to have the following background:

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

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

Software Dependencies

The table below lists IBM Informix database servers with their short database server names.

Table 1. IBM Informix 10.0, 9.x, 7.x, and 5.x Database Servers

Short Database Server Name	Complete Database Server Name
Dynamic Server 10.0	IBM Informix Dynamic Server, Version 10.0
Dynamic Server 9.40	IBM Informix Dynamic Server, Version 9.40

Table 1. IBM Informix 10.0, 9.x, 7.x, and 5.x Database Servers

Short Database Server Name	Complete Database Server Name
Dynamic Server 9.30	IBM Informix Dynamic Server, Version 9.30
Dynamic Server 9.21	IBM Informix Dynamic Server, Version 9.21
Workgroup Edition 9.21	IBM Informix Dynamic Server, Workgroup Edition, Version 9.21
Dynamic Server 9.20	IBM Informix Dynamic Server, Version 9.20
Dynamic Server 7.31	IBM Informix Dynamic Server, Version 7.31
Workgroup Edition 7.31	IBM Informix Dynamic Server, Workgroup Edition, Version 7.31
Dynamic Server, Linux Edition 7.31	IBM Informix Dynamic Server, Linux Edition, Version 7.31
Dynamic Server 7.30	IBM Informix Dynamic Server, Version 7.30
Workgroup Edition 7.30	IBM Informix Dynamic Server, Workgroup Edition, Version 7.30
Dynamic Server, Linux Edition 7.30	IBM Informix Dynamic Server, Linux Edition, Version 7.30
Dynamic Server 7.24	IBM Informix Dynamic Server, Version 7.24
Workgroup Edition 7.24	IBM Informix Dynamic Server, Workgroup Edition, Version 7.24
OnLine 5.1x	IBM Informix OnLine, Version 5.1x
SE 7.25	IBM Informix SE, Version 7.25
SE 7.24	IBM Informix SE, Version 7.24
SE 7.23	IBM Informix SE, Version 7.23
SE 7.22	IBM Informix SE, Version 7.22
SE 5.1x	IBM Informix SE, Version 5.1x

In this guide, a short database server name followed by a version number refers only to that version, or set of versions, of the database server. A version number without an “x” refers to a single version of the database server; for example, Dynamic Server 9.21 refers only to IBM Informix Dynamic Server, Version 9.21. A version number with an “x” can refer to more than one version of the database server; for example, Dynamic Server 9.x refers to all Dynamic Server Versions 9.20, 9.21, 9.30, and 9.40.

The migration can be between operating systems. Each of the Informix database servers can run on one or more of the following operating systems:

- UNIX
- Linux
- Windows 2000
- Windows NT
- Windows XP
- Windows 95

Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All the information related to character set, collation, and representation of numeric data, currency, date, and time is brought together in a single environment, called a GLS (Global Language Support) locale.

The examples in this manual are for the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for date, time, and currency. In addition, this locale supports the ISO 8859-1 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 the database server products, includes one or more of the following demonstration databases:

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

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

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

New Features in Database Servers

For a comprehensive list of new database server features with a short description of each feature, see your *IBM Informix: Getting Started Guide* and your release notes.

For a short summary list of new features, see “Changes in Dynamic Server 10.0, 9.40, 9.30, and 9.2x” on page 3-4.

Documentation Conventions

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

The following conventions are discussed:

- Typographical conventions
- Other conventions
- Syntax diagrams
- Command-line conventions
- Example code conventions

Typographical Conventions

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

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
<i>italics</i> <i>italics</i> <i>italics</i>	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface boldface	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
monospace <i>monospace</i>	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
>	This symbol indicates a menu item. For example, “Choose Tools > Options ” means choose the Options item from the Tools menu.

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

Feature, Product, and Platform

Feature, product, and platform markup identifies paragraphs that contain feature-specific, product-specific, or platform-specific information. Some

examples of this markup follow:

Dynamic Server

Identifies information that is specific to IBM Informix Dynamic Server

End of Dynamic Server

Extended Parallel Server

Identifies information that is specific to IBM Informix Extended Parallel Server

End of Extended Parallel Server

UNIX Only

Identifies information that is specific to UNIX platforms

End of UNIX Only

Windows Only

Identifies information that is specific to the Windows environment

End of Windows Only

This markup can apply to one or more paragraphs within a section. When an entire section applies to a particular product or platform, this is noted as part of the heading text, for example:

Table Sorting (Linux Only)

Syntax Diagrams

This guide uses syntax diagrams built with the following components to describe the syntax for statements and all commands other than system-level commands.


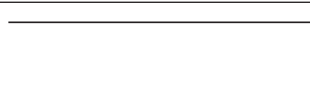
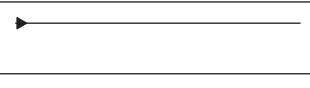
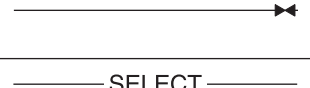
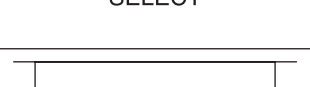

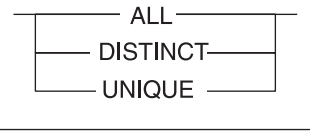
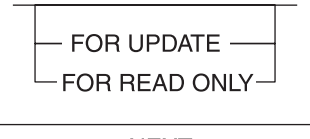
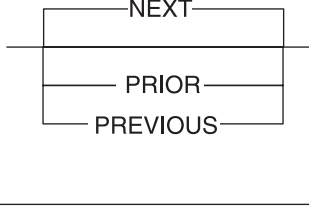
Note: Starting in 2004, syntax diagrams have been reformatted to conform to the IBM standard.

Syntax diagrams depicting SQL and command-line statements have changed in the following ways:

- The symbols at the beginning and end of statements are now double arrows instead of a vertical line at the end.
- The symbols at the beginning and end of syntax segment diagrams are now vertical lines instead of arrows.

- How many times a loop can be repeated is now explained in a diagram footnote instead of a number in a gate symbol.
- Syntax statements that are longer than one line now continue on the next line instead of looping down with a continuous line.
- Product or condition-specific paths are now explained in diagram footnotes instead of icons.

The following table describes syntax diagram components.

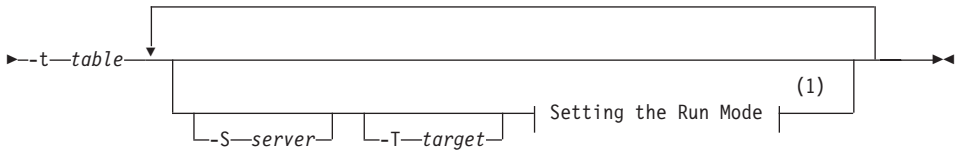
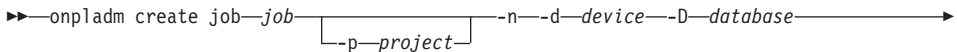
Component represented in PDF	Component represented in HTML	Meaning
	>>-----	Statement begins.
	----->	Statement continues on next line.
	>-----	Statement continues from previous line.
	-----><	Statement ends.
	-----SELECT-----	Required item.
	---+-----+--- '-----LOCAL-----'	Optional item.
	---+-----ALL-----+--- +--DISTINCT-----+ '---UNIQUE-----'	Required item with choice. One and only one item must be present.
	---+-----+--- +--FOR UPDATE-----+ '--FOR READ ONLY--'	Optional items with choice are shown below the main line, one of which you might specify.
	.---NEXT-----. ---+-----+--- +---PRIOR-----+ '---PREVIOUS-----'	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.

Component represented in PDF	Component represented in HTML	Meaning
	<pre> -----,----- V -----+----- +---index_name---+ '---table_name---' </pre>	Optional items. Several items are allowed; a comma must precede each repetition.
	<pre>>>- Table Reference -<<</pre>	Reference to a syntax segment.
<p>Table Reference</p>	<pre>Table Reference ---+-----view-----+--- +-----table-----+ '-----synonym-----'</pre>	Syntax segment.

How to Read a Command-Line Syntax Diagram

The following command-line syntax diagram uses some of the elements listed in the table in the previous section.

Creating a No-Conversion Job

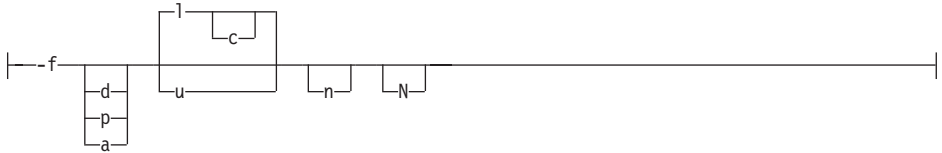


Notes:

1 See page 17-4

The second line in this diagram has a segment named “Setting the Run Mode,” which according to the diagram footnote, is on page 17-4. This segment is shown in the following segment diagram (the diagram uses segment start and end components).

Setting the Run Mode:



To construct a command correctly, start at the top left with the command. Follow the diagram to the right, including the elements that you want. The elements in the diagram are 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.

Your diagram is complete.

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.

Example Code Conventions

Examples of SQL code occur throughout this manual. 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 DB–Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.

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

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

Additional Documentation

For additional information, refer to the following types of documentation:

- Installation guides
- Online notes
- Informix error messages
- Manuals
- Online help

Installation Guides

Installation guides are located in the `/doc` directory of the product CD or in the `/doc` directory of the product's compressed file if you downloaded it from the IBM Web site. Alternatively, you can obtain installation guides from the IBM Informix Online Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/>.

Online Notes

The following sections describe the online files that supplement the information in this manual. Please examine these files before you begin using your IBM Informix product. They contain vital information about application and performance issues.

Online File	Description	Format
TOC Notes	The TOC (Table of Contents) notes file provides a comprehensive directory of hyperlinks to the release notes, the fixed and known defects file, and all the documentation notes files for individual manual titles.	HTML
Documentation Notes	The documentation notes file for each manual contains important information and corrections that supplement the information in the manual or information that was modified since publication.	HTML, text
Release Notes	The release notes file describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. For some products, this file also contains information about any known problems and their workarounds.	HTML, text
Machine Notes	(Non-Windows platforms only) The machine notes file describes any platform-specific actions that you must take to configure and use IBM Informix products on your computer.	text
Fixed and Known Defects File	This text file lists issues that have been identified with the current version. It also lists customer-reported defects that have been fixed in both the current version and in previous versions.	text

Locating Online Notes

Online notes are available from the IBM Informix Online Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/>. Additionally you can locate these files before or after installation as described below.

Before Installation

All online notes are located in the **/doc** directory of the product CD. The easiest way to access the documentation notes, the release notes, and the fixed and known defects file is through the hyperlinks from the TOC notes file.

The machine notes file and the fixed and known defects file are only provided in text format.

After Installation

On UNIX platforms in the default locale, the documentation notes, release notes, and machine notes files appear under the `$INFORMIXDIR/release/en_us/0333` directory.

Dynamic Server

On Windows the documentation and release notes files appear in the **Informix** folder. To display this folder, choose **Start > Programs > IBM Informix Dynamic Server *version* > Documentation Notes** or **Release Notes** from the taskbar.

Machine notes do not apply to Windows platforms.

End of Dynamic Server

Online Notes Filenames

Online notes have the following file formats:

Online File	File Format	Examples
TOC Notes	<i>prod_os_tocnotes_version.html</i>	ids_win_tocnotes_10.0.html
Documentation Notes	<i>prod_bookname_docnotes_version.html/txt</i>	ids_hpl_docnotes_10.0.html
Release Notes	<i>prod_os_relnotes_version.html/txt</i>	ids_unix_relnotes_10.0.txt
Machine Notes	<i>prod_machine_notes_version.txt</i>	ids_machine_notes_10.0.txt
Fixed and Known Defects File	<i>prod_defects_version.txt</i>	ids_defects_10.0.txt client_defects_2.90.txt
	<i>ids_win_fixed_and_known_defects_version.txt</i>	ids_win_fixed_and_known_defects_10.0.txt

Informix Error Messages

This file is a comprehensive index of error messages and their corrective actions for the Informix products and version numbers.

On UNIX platforms, use the **finderr** command to read the error messages and their corrective actions.

Dynamic Server

On Windows, use the Informix Error Messages utility to read error messages and their corrective actions. To display this utility, choose **Start > Programs > IBM Informix Dynamic Server *version* > Informix Error Messages** from the taskbar.

End of Dynamic Server

You can also access these files from the IBM Informix Online Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/>.

Manuals

Online Manuals

A CD that contains your manuals in electronic format is provided with your IBM Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print online manuals, see the installation insert that accompanies your CD. You can also obtain the same online manuals from the IBM Informix Online Documentation site at <http://www.ibm.com/software/data/informix/pubs/library/>.

Printed Manuals

To order hardcopy manuals, contact your sales representative or visit the IBM Publications Center Web site at <http://www.ibm.com/software/howtobuy/data.html>.

Online Help

IBM Informix online help, provided with each graphical user interface (GUI), displays information about those interfaces and the functions that they perform. Use the help facilities that each GUI provides to display the online help.

Accessibility

IBM is committed to making our documentation accessible to persons with disabilities. Our books are available in HTML format so that they can be accessed with assistive technology such as screen reader software. The syntax diagrams in our manuals are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader. For more information about the dotted decimal format, see the Accessibility appendix.

IBM Informix Dynamic Server Version 10.0 and CSDK Version 2.90 Documentation Set

The following tables list the manuals that are part of the IBM Informix Dynamic Server, Version 10.0 and the CSDK Version 2.90, documentation set. PDF and HTML versions of these manuals are available at <http://www.ibm.com/software/data/informix/pubs/library/>. You can order hardcopy versions of these manuals from the IBM Publications Center at <http://www.ibm.com/software/howtobuy/data.html>.

Table 1. Database Server Manuals

Manual	Subject
Administrator's Guide	Understanding, configuring, and administering your database server.
Administrator's Reference	Reference material for Informix Dynamic Server, such as the syntax of database server utilities onmode and onstat , and descriptions of configuration parameters, the sysmasters tables, and logical-log records.
Backup and Restore Guide	The concepts and methods you need to understand when you use the ON-Bar and ontape utilities to back up and restore data.
DB-Access User's Guide	Using the DB-Access utility to access, modify, and retrieve data from Informix databases.
DataBlade API Function Reference	The DataBlade API functions and the subset of ESQL/C functions that the DataBlade API supports. You can use the DataBlade API to develop client LIBMI applications and C user-defined routines that access data in Informix databases.
DataBlade API Programmer's Guide	The DataBlade API, which is the C-language application-programming interface provided with Dynamic Server. You use the DataBlade API to develop client and server applications that access data stored in Informix databases.
Database Design and Implementation Guide	Designing, implementing, and managing your Informix databases.
Enterprise Replication Guide	How to design, implement, and manage an Enterprise Replication system to replicate data between multiple database servers.
Error Messages file	Causes and solutions for numbered error messages you might receive when you work with IBM Informix products.
Getting Started Guide	Describes the products bundled with IBM Informix Dynamic Server and interoperability with other IBM products. Summarizes important features of Dynamic Server and the new features for each version.
Guide to SQL: Reference	Information about Informix databases, data types, system catalog tables, environment variables, and the stores_demo demonstration database.
Guide to SQL: Syntax	Detailed descriptions of the syntax for all Informix SQL and SPL statements.
Guide to SQL: Tutorial	A tutorial on SQL, as implemented by Informix products, that describes the basic ideas and terms that are used when you work with a relational database.
High-Performance Loader User's Guide	Accessing and using the High-Performance Loader (HPL), to load and unload large quantities of data to and from Informix databases.
Installation Guide for Microsoft Windows	Instructions for installing IBM Informix Dynamic Server on Windows.
Installation Guide for UNIX and Linux	Instructions for installing IBM Informix Dynamic Server on UNIX and Linux.

Table 1. Database Server Manuals (continued)

Manual	Subject
J/Foundation Developer's Guide	Writing user-defined routines (UDRs) in the Java programming language for Informix Dynamic Server with J/Foundation.
Large Object Locator DataBlade Module User's Guide	Using the Large Object Locator, a foundation DataBlade module that can be used by other modules that create or store large-object data. The Large Object Locator enables you to create a single consistent interface to large objects and extends the concept of large objects to include data stored outside the database.
Migration Guide	Conversion to and reversion from the latest versions of Informix database servers. Migration between different Informix database servers.
Optical Subsystem Guide	The Optical Subsystem, a utility that supports the storage of BYTE and TEXT data on optical disk.
Performance Guide	Configuring and operating IBM Informix Dynamic Server to achieve optimum performance.
R-Tree Index User's Guide	Creating R-tree indexes on appropriate data types, creating new operator classes that use the R-tree access method, and managing databases that use the R-tree secondary access method.
SNMP Subagent Guide	The IBM Informix subagent that allows a Simple Network Management Protocol (SNMP) network manager to monitor the status of Informix servers.
Storage Manager Administrator's Guide	Informix Storage Manager (ISM), which manages storage devices and media for your Informix database server.
Trusted Facility Guide	The secure-auditing capabilities of Dynamic Server, including the creation and maintenance of audit logs.
User-Defined Routines and Data Types Developer's Guide	How to define new data types and enable user-defined routines (UDRs) to extend IBM Informix Dynamic Server.
Virtual-Index Interface Programmer's Guide	Creating a secondary access method (index) with the Virtual-Index Interface (VII) to extend the built-in indexing schemes of IBM Informix Dynamic Server. Typically used with a DataBlade module.
Virtual-Table Interface Programmer's Guide	Creating a primary access method with the Virtual-Table Interface (VTI) so that users have a single SQL interface to Informix tables and to data that does not conform to the storage scheme of Informix Dynamic Server.

Table 2. Client/Connectivity Manuals

Manual	Subject
Client Products Installation Guide	Installing IBM Informix Client Software Developer's Kit (Client SDK) and IBM Informix Connect on computers that use UNIX, Linux, and Windows.
Embedded SQLJ User's Guide	Using IBM Informix Embedded SQLJ to embed SQL statements in Java programs.

Table 2. Client/Connectivity Manuals (continued)

Manual	Subject
ESQL/C Programmer's Manual	The IBM Informix implementation of embedded SQL for C.
GLS User's Guide	The Global Language Support (GLS) feature, which allows IBM Informix APIs and database servers to handle different languages, cultural conventions, and code sets.
JDBC Driver Programmer's Guide	Installing and using Informix JDBC Driver to connect to an Informix database from within a Java application or applet.
.NET Provider Reference Guide	Using Informix .NET Provider to enable .NET client applications to access and manipulate data in Informix databases.
ODBC Driver Programmer's Manual	Using the Informix ODBC Driver API to access an Informix database and interact with the Informix database server.
OLE DB Provider Programmer's Guide	Installing and configuring Informix OLE DB Provider to enable client applications, such as ActiveX Data Object (ADO) applications and Web pages, to access data on an Informix server.
Object Interface for C++ Programmer's Guide	The architecture of the C++ object interface and a complete class reference.

Table 3. DataBlade Developer's Kit Manuals

Manual	Subject
DataBlade Developer's Kit User's Guide	Developing and packaging DataBlade modules using BladeSmith and BladePack.
DataBlade Module Development Overview	Basic orientation for developing DataBlade modules. Includes an example illustrating the development of a DataBlade module.
DataBlade Module Installation and Registration Guide	Installing DataBlade modules and using BladeManager to manage DataBlade modules in Informix databases.

Compliance with Industry Standards

The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) have jointly established a set of industry standards for the Structured Query Language (SQL). IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

IBM Welcomes Your Comments

We want to know about any corrections or clarifications that you would find useful in our manuals, which will help us improve future versions. Include the following information:

- The name and version of the manual that you are using
- Section and page number
- Your suggestions about the manual

Send your comments to us at the following email address:

`docinf@us.ibm.com`

This email address is reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact IBM Technical Support.

We appreciate your suggestions.

Part 1. Overview of IBM Informix Migration

Chapter 1. Database Server Migration

IBM Informix Database Server Products	1-1
Migration on the Same Operating System.	1-4
Source and Target Database Servers on UNIX	1-4
Source and Target Database Servers on Linux	1-4
Source and Target Database Server on Windows XP	1-5
Source and Target Database Servers on Windows NT	1-5
Source and Target Database Servers on Windows 2000	1-6
Source and Target Database Servers on Windows 95	1-6
Migration on Different Operating Systems	1-6
Database Server Migration Paths.	1-7

In This Chapter

This chapter provides an overview of database server migration, including:

- A list of all IBM Informix database server products and their operating systems
- Information about migration on the same operating system
- Information about migration on different operating systems
- Database server migration paths from previous IBM informix database servers to Version 10.0 of IBM Informix Dynamic Server

For information about how to move data between database servers on different operating systems, see Chapter 7, “Migrating Between Database Servers and Operating Systems,” on page 7-1

IBM Informix Database Server Products

Table 1-1 lists all Version 10.0, 9.x, 8.x, 7.x, and 5.1x IBM Informix database servers with the operating systems on which you can run them. Table 1-1 uses short names for the database servers. For the complete database server names that correspond to these short names, see “Software Dependencies” on page x of the Introduction.

Table 1-1. Operating Systems on Which to Run Informix Database Servers

Database Server	Operating Systems
Dynamic Server 10.0	UNIX Linux Windows 2000 Windows XP

Table 1-1. Operating Systems on Which to Run Informix Database Servers (continued)

Database Server	Operating Systems
Dynamic Server 9.40	UNIX Linux Windows 2000 Windows XP
Dynamic Server 9.30	UNIX Linux Windows 2000 Windows NT
Dynamic Server 9.21	UNIX Linux Windows 2000 Windows NT
Workgroup Edition 9.21	UNIX Linux Windows 2000 Windows 95
Dynamic Server 9.20	UNIX Linux Windows 2000 Windows NT
Extended Parallel Server 8.40	UNIX
Extended Parallel Server 8.32	UNIX
Extended Parallel Server 8.31	UNIX
Extended Parallel Server 8.30	UNIX
Dynamic Server with AD and XP Options 8.21	UNIX Linux Windows NT
Dynamic Server 7.31	UNIX Windows 2000 Windows NT Windows 95
Workgroup Edition 7.31	UNIX Windows 2000 Windows NT Windows 95
Dynamic Server, Linux Edition 7.31	Linux UNIX
Dynamic Server 7.30	UNIX Linux Windows NT Windows 95

Table 1-1. Operating Systems on Which to Run Informix Database Servers (continued)

Database Server	Operating Systems
Workgroup Edition 7.30	UNIX Windows NT Windows 95
Dynamic Server, Linux Edition 7.30	Linux UNIX
Dynamic Server 7.24	UNIX Windows NT Windows 95
Workgroup Edition 7.24	UNIX Windows NT Windows 95
OnLine 5.1x	UNIX Linux
SE 7.25	UNIX
SE 7.24	UNIX Linux Windows NT
SE 7.23	UNIX Windows NT
SE 7.22	UNIX Windows NT
SE 5.1x	UNIX

If you have Dynamic Server Version 9.40, 9.30, 9.2x, 7.3x, or 7.24, you can migrate directly to Dynamic Server Version 10.0, the current database server.

You can also migrate to the current database server from the following database servers after you migrate to an intermediary database server:

- IBM Informix Universal Server, Version 9.14 (Universal Server 9.14)
- IBM Informix OnLine Dynamic Server, Version 7.22 through Version 7.23 (OnLine Dynamic Server 7.22 through 7.23)

For information on migration paths, see “Database Server Migration Paths” on page 1-7.

To become more familiar with the Informix client-server environment, read *IBM Informix: Getting Started with Database Server Products*. This book contains information on the differences between Informix database servers plus information on network and server configurations. Also, read the *IBM Informix: Getting Started Guide* for your database server.

Migration on the Same Operating System

This section provides tables that show migration paths between database servers on the same operating system.

Source and Target Database Servers on UNIX

The table below lists the source and target versions for migration of database servers on UNIX.

Table 1-2. Migrating Between Database Servers on UNIX

Source Version	Target Version													
	10.0	9.40	9.30	9.20-21	9.14	8.40	8.31-32	8.30	8.21.UD4	8.21.U	7.30-31	7.25.U	7.24	5.1x
10.0	x	x	x	x	x						x		x	
9.40	x	x	x	x	x						x		x	
9.30	x	x	x	x	x						x		x	
9.21	x	x	x	x	x						x		x	
9.20	x	x	x	x	x						x		x	
9.14			x	x	x						x		x	
8.40						x	x	x	x		x		x	
8.32						x	x	x	x		x		x	
8.31						x	x	x	x		x		x	
8.30						x	x	x	x		x		x	
8.21.UD4								x	x	x				x
8.21.U									x	x				x
7.31	x	x	x	x	x	x	x	x			x		x	x
7.30	x	x	x	x	x	x	x	x			x		x	x
7.25												x	x	x
7.24	x	x	x	x	x	x	x	x	x		x	x	x	x
7.23.U				x	x						x	x	x	x
7.22.U				x	x						x	x	x	x
5.1x.U											x	x	x	x

Source and Target Database Servers on Linux

Table 1-3 lists the source and target versions for migration of database servers on Linux.

Table 1-3. Migrating Between Database Servers on Linux

Source Version	Target Version				
	10.0	9.40	9.30	7.31	7.30
10.0	x	x	x	x	x
9.40	x	x	x	x	x
9.30	x	x	x	x	x
9.21	x	x	x	x	x
9.20	x	x	x	x	x
7.31	x	x	x	x	x
7.30	x	x	x	x	x

Source and Target Database Server on Windows XP

Currently, only Dynamic Server Versions 9.4 and 10.0 are available on Windows XP.

Source and Target Database Servers on Windows NT

Important: Dynamic Server Version 10.0 is not available on Windows NT.

Table 1-4 lists the source and target versions for migration of previous database server versions on Windows NT.

Table 1-4. Migrating Between Database Servers on Windows NT

Source Version	Target Version						
	9.30.T	8.21.T	7.31.T	7.30.T	7.24.T	7.23.T	7.22.T
9.30.T	x		x	x	x		
9.21.T	x		x	x	x	x	x
9.20.T	x		x	x	x	x	x
9.14.T	x		x	x	x	x	x
8.21.T		x			x	x	x
7.31.T	x	x	x	x	x	x	x
7.30.T	x	x	x	x	x	x	x
7.24.T	x	x	x	x	x	x	x
7.23.T		x	x	x	x	x	x
7.22.T		x	x	x	x	x	x

Source and Target Database Servers on Windows 2000

Table 1-5 lists the source and target versions for migration of Informix database servers on Windows 2000.

Table 1-5. Migrating Between Database Servers on Windows 2000

Source Version	Target Version			
	10.0.T	9.40.T	9.30.T	7.31.T
10.0.T	x	x	x	x
9.40.T	x	x	x	x
9.30.T	x	x	x	x
9.21.T	x	x	x	x
7.31.T	x	x	x	x

Source and Target Database Servers on Windows 95

Table 1-6 lists the source and target versions for migration of Informix database servers on Windows 95.

Table 1-6. Migrating Between Database Servers on Windows 95

Source Version	Target Version		
	7.31.T	7.30.T	7.24.T
7.31.T	x	x	x
7.30.T	x	x	x
7.24.T	x	x	x

Migration on Different Operating Systems

If you are using Dynamic Server on Windows NT and migrating to Dynamic Server Version 10.0, you also need to migrate to Windows 2000 or Windows XP.

For specific information on migrating between operating systems, see Chapter 7, "Migrating Between Database Servers and Operating Systems," on page 7-1.

Table 1-7 lists the operating systems supported by each version of Dynamic Server.

Table 1-7. Migrating Between Operating Systems

Server Version	Operating System					
	UNIX	Linux	Windows XP	Windows NT	Windows 2000	Windows 95
10.0	x	x	x		x	
9.40	x	x	x		x	
9.30	x	x		x	x	
9.21	x	x		x	x	
9.20	x	x		x		
9.14	x			x		
8.40	x					
8.31	x					
8.30	x					
8.21	x			x		
7.31	x	x		x	x	x
7.30	x	x		x		x
7.25	x					
7.24	x			x		x
7.23	x			x		
7.22	x			x		
5.1x	x					

Database Server Migration Paths

If you have Dynamic Server Version 9.40, 9.30, 9.2x, 7.3x, or 7.24, you can migrate directly to Dynamic Server Version 10.0. Table 1-8 lists migration paths for converting from an existing source database server to a newer target database server. The **Reference** column shows where to find information about how to convert.

If you are converting from OnLine 5.1x, Dynamic Server 7.22, Dynamic Server 7.23, or Dynamic Server 7.24 (with Enterprise Replication), you must convert to an intermediate, slightly older version of Dynamic Server before you convert to Version 10.0. Figure 1-1 shows the paths for converting from these older database servers to Dynamic Server Version 10.0.

Table 1-8. Converting to a Newer Database Server

Source Database Server	Target Database Server	Reference
Dynamic Server 9.40	Dynamic Server 10.0	Chapter 3, "Preparing for Migration," on page 3-1 and Chapter 5, "Converting to Dynamic Server 10.0," on page 5-1
Dynamic Server 9.30		
Dynamic Server 9.20 or Dynamic Server 9.21		
Dynamic Server 7.3x		
Dynamic Server 7.24 or Workgroup Edition 7.24		
Universal Server 9.14	Dynamic Server 9.30	The Version 9.30 Migration Guide
OnLine Dynamic Server 7.23	Dynamic Server 7.31	The Version 7.31 Migration Guide
OnLine Dynamic Server 7.22		
OnLine 5.1x		

Figure 1-1 shows the paths for converting from specific older database servers to Dynamic Server Version 10.0.

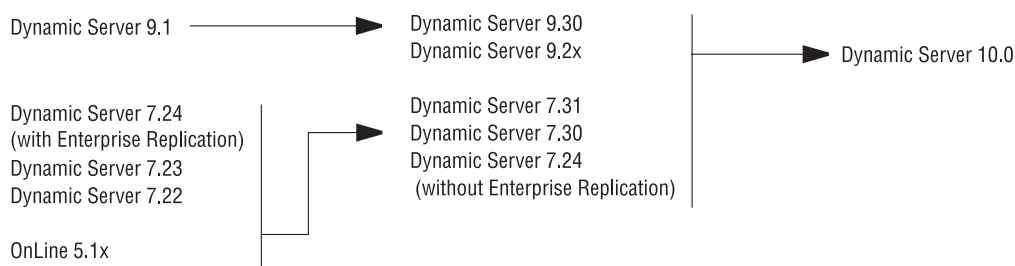


Figure 1-1. Migration Paths to Dynamic Server 10.0

For information on migrating directly to Dynamic Server Version 10.0 from a 9.40, 9.30, 9.2x, 7.3x, or 7.24 database server, see Chapter 3, "Preparing for Migration," on page 3-1 and Chapter 5, "Converting to Dynamic Server 10.0," on page 5-1.

If you need to migrate to another version of Dynamic Server before you migrate to Version 10.0, see the Migration Guide that is included in the documentation set for that database server. For example, if you need to migrate to Version 7.31 before migrating to Version 10.0, see the Version 7.31 *IBM Informix: Migration Guide* for information on migrating to Version 7.31.

You must follow the steps in that guide before you follow the instructions in the current guide for migrating to Version 10.0.

Table 1-9 shows the database servers to which you can revert from Dynamic Server 10.0. After you install Version 10.0, you cannot revert from Version 10.0 to any IBM Informix database server that is not shown in the list. The **Page** column shows where to find information in this manual about how to revert.

Table 1-9. Reverting to an Older Database Server

Source	Target	Page
Dynamic Server 10.0	Dynamic Server 9.40 Dynamic Server 9.30 Dynamic Server 9.2x Dynamic Server 7.3x Dynamic Server, Linux Edition Version 7.3x Workgroup Edition 7.3x Dynamic Server 7.24 Dynamic Server, Linux Edition Version 7.3x Workgroup Edition 7.24	Chapter 6, "Reverting from Dynamic Server 10.0," on page 6-1

Chapter 2. Data Migration

Migrating a Database or Selected Data	2-2
Migrating Between Different Versions of a Database Server	2-2
Changing Database Servers, Operating Systems, or GLS Locales	2-2
Distributing a Client Application.	2-2
Importing Non-IBM Informix Data	2-3
Setting Environment Variables Before Using Utilities	2-3
Choosing Data-Migration Tools	2-3
Automatic Data Migration	2-5
The onunload and onload Utilities	2-5
Constraints on Using onunload and onload	2-6
Restrictions on Using onunload and onload	2-6
The dbexport and dbimport Utilities	2-7
Destination Options	2-8
Location Options	2-8
Database Server Options	2-9
LOAD, UNLOAD, and dbload	2-9
The dbschema Utility	2-10
Guidelines for Using dbschema	2-11
DB-Access Input from dbschema Output.	2-11
Object Modes and Violation Detection	2-11
External Tables	2-12
The High-Performance Loader	2-12
HPL Tools	2-12
Performance Advantage of the HPL	2-12
Nonlogging Raw Tables (IDS 9.x or Later)	2-13
Movement of TEXT and BYTE Data	2-13
Moving Data Between Computers and Dbspaces	2-14
Importing Data from a Non-IBM Informix Source.	2-14
Importing Data with IBM Informix Enterprise Gateway Products	2-14

In This Chapter

This chapter provides an overview of data migration and compares the IBM Informix migration utilities. This chapter covers the following topics:

- Migrating a database or selected data
- Setting environment variables before using utilities
- Choosing the most effective data migration tool
- Moving data between computers and dbspaces

Migrating a Database or Selected Data

You might need to perform data migration on a database or selected data to complete any of the following tasks:

- Migrating between different versions of a database server
- Changing database servers, operating systems, or GLS locales
- Distributing a client application
- Importing non-Informix data

Migrating Between Different Versions of a Database Server

When you convert to a later version of a database server or revert to an earlier version, you need to consider the following data migration issues:

- Changes in the configuration parameters and environment variables
- Amount of memory and dbspace space required
- Organization of the data

Later chapters discuss these issues in detail.

Changing Database Servers, Operating Systems, or GLS Locales

You might want to move between database servers, to a different operating system, or to a different GLS locale. You also might want to change the database schema to accommodate more information, to provide for growth, or to enhance performance.

For information about how to move data to a different operating system, see Chapter 7, “Migrating Between Database Servers and Operating Systems,” on page 7-1.

Global Language Support

For information about how to move to a different GLS locale, see the *IBM Informix: GLS User's Guide*.

End of Global Language Support

Distributing a Client Application

After you convert a database server on the same operating system or move the database server to another compatible computer, review the client applications and **sqlhosts** file or registry-key connections. You might need to recompile or modify client applications, or update the **sqlhosts** file or registry key.

Verify that the client-application version you use is compatible with your database server version. Update the **sqlhosts** file or registry key for the client applications with the new database server information.

For more information about interactions between client applications and different database servers, refer to a client manual.

Importing Non-IBM Informix Data

You can use the **dbimport** and **dbload** utilities, the High-Performance Loader (HPL), the IBM Informix Enterprise Gateway products, or external tables to import data from non-Informix sources.

Setting Environment Variables Before Using Utilities

Before you use any data migration utility, you must set your `PATH`, `INFORMIXDIR`, and `INFORMIXSERVER` environment variables. For information about environment variables, see the *IBM Informix: Guide to SQL Reference*.

Choosing Data-Migration Tools

This section provides brief descriptions of the tools, utilities, and SQL statements that you can use to move data from one database to another.

When you migrate to the current version of the database server, you can use these data-migration tools:

- The **onunload** and **onload** utilities (only between database servers of the same version)
- The **dbexport** and **dbimport** utilities
- `UNLOAD` and `LOAD` statements and the **dbload** utility
- The **dbschema** utility
- The High-Performance Loader (HPL)
- Nonlogging raw tables

For more information about the utilities and statements listed above, see the chapters in the last section of this guide.

In some previous versions of the database server, you could use IBM Informix Enterprise Command Center (IECC) to move data from one database to another.

If you need to migrate to another version of Dynamic Server before you migrate to Version 10.0 and you want information on the migration tools and utilities that can be used for migration to an intermediary version of Dynamic Server, see the Migration Guide that is included in the documentation set for that database server.

The table below contains a list of the tools you can use for each IBM Informix database server. An asterisk (*) after the utility name identifies a tool that was used in a previous version of Dynamic Server, but is no longer used.

Table 2-1. Utilities for Moving Data

Utility	IDS 10.0	IDS 9.40 and 9.30	IDS 7.3x or 7.24	SE 7.2x and 5.1x	OL 5.1x
dbexport/ dbimport	x	x	x	x	x
dbload	x	x	x	x	x
HPL	x	x	x		
onunload/ onload	x	x	x		
UNLOAD/ LOAD	x	x	x	x	x
IECC*			x		

The best method for moving data depends on your operating system and whether you want to move an entire database, selected tables, or selected columns from a table. The table below summarizes the characteristics of the methods for loading data and the constraints and advantages of each method.

Table 2-2. Comparison of Tools for Loading Data

Utility	dbexport/ dbimport	dbload	HPL	onunload/ onload	UNLOAD/ LOAD
Granularity of Data	Database only	Partial or complete table	Partial or com-plete table	Table or database	Partial or complete table
Performance	Moderate	Slow	Fast	Fast	Moderate
Source of Data	Usually prod-uced by dbexport	Any data in the format specified by the input file	Any ASCII or COBOL data. User can create custom read capabilities.	Must be prod-uced by on-unload	Any data in the specified format, usually produced by UNLOAD
Database Schema	Can modify	Can modify	Can modify	Cannot modify	Can modify
Location of Data	Disk or tape	Disk only	Disk, tape, or pipe	Disk or tape	Disk only
Type of File	Text	Text	Text	Binary	Text
Logging Status	Logging optional	Logging optional	Logging optional	Logging must be turned off	Logging optional

Table 2-2. Comparison of Tools for Loading Data (continued)

Utility	dbexport/ dbimport	dbload	HPL	onunload/ onload	UNLOAD/ LOAD
Move Data Between Operating Systems?	Yes	Yes	Yes, or from a non-Informix data-base	No	Yes
Ease of Use	Moderate	Moderate	Most difficult	More difficult	Easiest

The following sections provide guidelines on how to choose the appropriate migration tool or tools.

Automatic Data Migration

Data migration is automatic. Automatic data migration means that when you migrate from one database server to another, you do not need to use any data migration tools to move the data. The data migrates automatically from the source database server to the target database server after you bring up the target database server.

Data migration can also be automatic when you move between different versions of a database server in the same operating system.

The onunload and onload Utilities

The **onunload** and **onload** utilities provide the fastest way to move data, but they do not let you modify the database schema or move from one operating system or database server version to another. The **onunload** utility unloads data from the specified database or table onto a tape or a file on disk in disk-page-sized units, making this utility more efficient than **dbexport**. The **onload** utility takes a tape or a file that the **onunload** utility creates and re-creates the database or the table. The **onunload** and **onload** utilities are faster than **dbimport**, **dbload**, or **LOAD** but are much less flexible.

Because the data is written in page-sized units, you can use **onunload** and **onload** only when certain constraints are met. For example, you cannot use **onunload** and **onload** to move data between UNIX and Windows. You can, however, use the **onunload** and **onload** utilities to move data between computers that use the same database server on the same platform.

You can only use **onunload** and **onload** if your answer to each of the following questions is *yes*. If your answer is *no*, you cannot use **onunload** and **onload**.

Use onunload and onload If Your Answer To Each Question Is Yes
Do you want to move to another database server of the same version?

Use onunload and onload If Your Answer To Each Question Is Yes
Do you want to move data between GLS and non-GLS databases?
Do you want to modify the database schema?
Do you want to move an entire database or an entire table?
Are the page images compatible? Are the numeric representations the same? Is the target database server on the same hardware platform?

Constraints on Using onunload and onload

The **onunload** and **onload** utilities are the fastest way to unload and load data, but you can use them only when all the following criteria are the same for the source and target computers:

- Page size
- Representation of numeric data
- Byte alignment for structures and unions
- Informix database server version

You cannot use **onunload** and **onload** to move data between UNIX or Linux and Windows because they use different page sizes. For example, the page size is two kilobytes on some UNIX systems and four kilobytes on Windows.

You can use the **onunload** and **onload** utilities to unload from and load data into the following database servers between computers that have the same operating system:

- Dynamic Server 10.0, 9.40, 9.30, or 9.2x (only databases that do not contain extended data types)
- Dynamic Server 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24

For example, your site purchases a more powerful UNIX computer to allow faster access for users. You need to transfer existing databases to the new database server on the new computer. Use **onunload** to unload data from the first database server and then use **onload** to load the data into the second database server. Both database servers must have the same version number, or they must have compatible version numbers. You can move an entire database or selected tables only, but you cannot modify the database schema.

Restrictions on Using onunload and onload

The **onunload** and **onload** utilities have the following restrictions:

Do not use **onunload** and **onload** to move data between two Dynamic Server 10.0, 9.40, 9.30, or 9.2x databases if either database contains an extended data type.

Use the HPL instead to move the data.

Global Language Support

- You cannot use **onunload** and **onload** to move data between non-GLS and GLS locales.

End of Global Language Support

You can use **onunload** and **onload** to move data between databases if the NLS and GLS locales are identical. For example, if user A has a French locale NLS table on server A and tries to load data into a German locale GLS table on server B, **onload** and **onunload** report errors. However, if both the NLS and GLS tables were created with the same French locale, **onload** and **onunload** would work.

The tape that **onload** reads contains binary data that is stored in disk-page-sized units. For this reason, the computers where the original database resides (where you use **onunload**) and where the target database will reside (where you use **onload**) must have the following characteristics:

- The same page size
- The same representation of numeric data
- The same byte alignment for structures and unions

If the page sizes are different, **onload** fails. If the alignment or numeric data types on the two computers are different (for example, with the most-significant byte last instead of first or different float-type representations), the contents of the data page could be misinterpreted.

Important: You cannot use the **onload** and **onunload** utilities to move data from one version of a database server to another. You also cannot use these utilities to move data between different types of database servers.

For additional constraints and restrictions, see Chapter 13, “The onunload and onload Utilities,” on page 13-1.

The dbexport and dbimport Utilities

If you cannot use **onunload** and **onload**, you have three other methods to choose from:

- The **dbload** utility (to load data)

- The **dbexport** and **dbimport** utilities
- The UNLOAD and LOAD SQL statements

All these methods enable you to modify the database schema. The **dbexport** and **dbimport** utilities provide some flexibility, but you must move an entire database.

If you cannot use **onunload** and **onload** to export and import data, you can unload your data to text files. You can use the **dbexport** utility to unload data to tape from any of the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x, or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE

The UNLOAD statement lets you manipulate the data as you unload it, but it requires that you unload to files on disk instead of to tape. If you unload to disk files, you might need to use UNIX, Linux, or Windows utilities to load those files onto tape.

The **dbexport** utility unloads a database into text files and creates a schema file. You can use the schema file with **dbimport** to re-create the database schema in another IBM Informix environment. You can edit the schema file to modify the database that **dbimport** creates. The **dbexport** utility supports Dynamic Server 10.0, 9.40, 9.30, and 9.2x data types.

Destination Options

The **dbexport** utility supports the following destination options:

- Unload a database and its schema file to disk
- Unload a database and its schema file to tape
- Unload the schema file to disk and unload the data to tape

The **dbimport** utility creates a database and loads it with data from text files. The input files consist of a schema file that is used to re-create the database and data files that contain the database data. Normally, you generate the input files with the **dbexport** utility, but you can use any properly formatted input files. The **dbimport** utility supports new data types in Dynamic Server 10.0, 9.40, 9.30, and 9.2x.

Location Options

The **dbimport** utility can use files from the following location options:

- All input files are located on disk
- All input files are located on tape

- The schema file is located on disk, and the data files are located on tape

Database Server Options

The **dbimport** utility supports the following options for a new IBM Informix database server:

- Create an ANSI-compliant database (includes unbuffered logging).
- Establish transaction logging for a database (unbuffered or buffered logging).
- Specify the dbspace where the database will reside.

LOAD, UNLOAD, and dbload

The LOAD statement is moderately fast and easy to use, but it can only accept specified data formats. You usually use LOAD with data that is prepared with an UNLOAD statement.

You can use the UNLOAD statement in DB–Access to unload selected rows from a table into a text file.

To load tables, use LOAD or **dbload**. To manipulate a data file that you are loading or to access a database while it is loading, use the **dbload** utility. The cost of the flexibility is the time you spend creating the **dbload** command file and slower execution. When possible, use the LOAD statement, which is faster than **dbload**.

Figure 2-1 summarizes questions to help you choose among these methods.

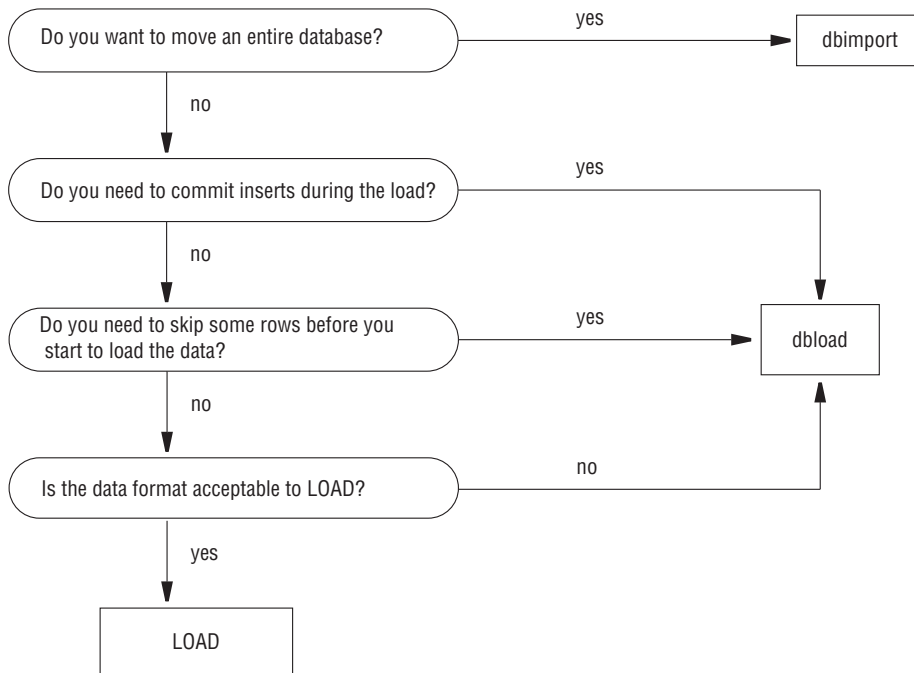


Figure 2-1. Choosing Among *dbimport*, *dbload*, and *LOAD*

The **dbload** utility gives you a great deal of flexibility, but it is not as fast as the other methods, and you must prepare a command file to control the input. You can use **dbload** with data in a variety of formats.

The **dbload** utility offers the following advantages over the **LOAD** statement:

- You can use **dbload** to load data from input files that were created with a variety of format arrangements. The **dbload** command file can accommodate data from entirely different database management systems.
- You can specify a starting point in the load by directing **dbload** to read but ignore *x* number of rows.
- You can specify a batch size so that after every *x* number of rows are inserted, the insert is committed.
- You can limit the number of bad rows read, beyond which **dbload** ends.

The cost of **dbload** flexibility is the time and effort spent creating the **dbload** command file, which is required for **dbload** operation. The input files are not specified as part of the **dbload** command line, and neither are the tables into which the data is inserted. This information is contained in the command file.

The **dbschema** Utility

You can use the **dbschema** utility for the following purposes:

- To display the SQL statements (the *schema*) that are required to replicate a database or a specific table, view, or procedure
- To display the schema for the Information Schema views
- To display the distribution information that is stored for one or more tables in the database
- To display information on user-defined data types and row types

Guidelines for Using `dbschema`

Global Language Support

When the GLS environment variables are set correctly, as the *IBM Informix: GLS User's Guide* describes, **dbschema** can handle foreign characters in Dynamic Server 9.40, 9.30, 9.2x, 7.3x, or 7.24; Dynamic Server, Linux Edition 7.3x; or Workgroup Edition 7.3x or 7.24.

End of Global Language Support

You can use delimited identifiers with the **dbschema** utility. It detects database objects that are keywords, mixed case, or that have special characters, and places double quotation marks around them.

DB-Access Input from `dbschema` Output

You can use the **dbschema** utility to get the schema of a database and redirect the **dbschema** output to a file. Later, you can feed this file to DB-Access to re-create the database.

Object Modes and Violation Detection

The **dbschema** output supports object modes and violation detection, as follows:

- The output shows the names of not-null constraints after the not-null specifications.
You can use the output of the utility as input to create another database. If the same names were not used for not-null constraints in both databases, problems could result.
- The output shows the object mode of objects that are in the disabled state. These objects can be constraints, triggers, or indexes.
- The output shows the object mode of objects that are in the filtering state. These objects can be constraints or unique indexes.
- The output shows the violations and diagnostics tables that are associated with a base table (if violations and diagnostics tables were started for the base table).

For more information about object modes and violation detection, see the SET, START VIOLATIONS TABLE, and STOP VIOLATIONS TABLE statements in the *IBM Informix: Guide to SQL Syntax*.

External Tables

You can use external tables to unload and load data when no other data migration tool is available.

For detailed information on external tables, see the description of the CREATE EXTERNAL TABLE statement in the *IBM Informix: Guide to SQL Syntax*.

The High-Performance Loader

The High-Performance Loader (HPL) utility uses parallel processing to perform fast data loading and unloading. The HPL is available with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x, and 7.24
- Workgroup Edition 7.3x and 7.24

The HPL requires significant preparation time but it is fast. Use the HPL for large migration jobs. The HPL can load data from any ASCII or COBOL file that meets certain format requirements.

You can use the HPL to load from large ASCII or COBOL databases. COBOL is supported up to Version 7.3.

HPL Tools

In addition to the advantage of speed, the following HPL features provide powerful tools for handling data from non-Informix sources:

- Drivers to handle different database types
- Filters and functions to manipulate data
- Code-set conversion
- The **ipload** GUI for UNIX
- The **onpladm** command-line utility for UNIX and Windows

Performance Advantage of the HPL

For extremely large databases, the HPL has a performance advantage over other IBM Informix data-migration utilities because it performs I/O and code-set conversions in parallel. The user, however, must invest significant preparation time before using the HPL, and the HPL program has a significant start-up time. Therefore, use the HPL only for large databases, for which the time savings in the actual loading or unloading of data makes the preparation time worthwhile.

For more information about the HPL, refer to the *IBM Informix: High-Performance Loader User's Guide*.

Nonlogging Raw Tables (IDS 9.x or Later)

You can use nonlogging raw tables in a logging database to speed up the initial loading and validation of data in Dynamic Server 9.40, 9.30, or 9.2x, which creates standard tables that use logging by default. Data warehousing and other applications can have very large tables that take a long time to load. Nonlogging tables are faster to load than logging tables.

To create a nonlogging table for loading, you can use the CREATE RAW TABLE statement, or you can use the ALTER TABLE statement to change the table type from STANDARD to RAW. Tables of type RAW do not allow indexes or referential constraints, so the initial loading is faster than with tables of type STANDARD. After the loading of a raw table is complete, you can change it to a logging table (in a logging database) by changing the table type to STANDARD. Then you can use ALTER TABLE statements to add referential constraints to the table and CREATE INDEX statements to add indexes. For more information on these SQL statements, see the *IBM Informix: Guide to SQL Syntax*.

To load raw tables, you can use any data loading utility, such as **dbimport** or HPL in express mode. After you load data, perform a level-0 backup. Before you modify any data in a raw table or use it in a transaction, change the table type to STANDARD.

If an error or failure occurs during the loading of a raw table, the resulting data is whatever was on the disk at the time of the failure.

The **dbexport** and **dbschema** utilities support the CREATE RAW TABLE and ALTER TABLE...TYPE (RAW) statements.

For more information on nonlogging tables, see your *IBM Informix: Administrator's Guide*. For more information on how to improve the performance of loading very large tables, see your *IBM Informix: Performance Guide*. For more information on the ALTER TABLE statement, see the *IBM Informix: Guide to SQL Syntax*.

Movement of TEXT and BYTE Data

An Informix database server scans TEXT and BYTE data into an existing table in one of the following ways.

- SQL LOAD statement
- The **dbload** utility
- Informix ESQL/C program
- The HPL
- External table

Informix database servers do not have any mechanisms for compressing TEXT and BYTE data after the data has been scanned into a database.

Moving Data Between Computers and Dbspaces

This section discusses moving data between different computers and importing data from non-Informix environments. Except when you use the HPL or external tables, you must unload your data to ASCII files before you move the data to another computer.

If you are moving to an Informix database server on another computer, you can use the **dbimport** and **dbload** utilities to load the data that you exported.

If you are moving data to a non-Informix application, you might need to use the UNLOAD statement because it lets you specify the delimiter that is used in the data files.

Importing Data from a Non-IBM Informix Source

The **dbimport** and **dbload** utilities can import data from any ASCII file that is properly formatted. Most applications that produce data can export the data into files that have a suitable format for **dbimport**. If the format of the data is not suitable, use UNIX, Linux, or Windows utilities to reformat the data before you import it into one of the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x, or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE

In addition to **dbimport** and **dbload**, the IBM Informix Enterprise Gateway products and the HPL provide ways to access information from non-Informix sources.

Importing Data with IBM Informix Enterprise Gateway Products

IBM Informix Enterprise Gateway with DRDA lets you query databases that conform to the DRDA protocol published by IBM. You can use this Gateway product to query a DRDA database and then insert the results into an Informix database. For example, to import data, execute a SELECT statement to select data from the non-Informix database and then execute an INSERT statement to insert data into the Informix database. For more information, refer to the *IBM Informix: Enterprise Gateway with DRDA User Manual*.

UNIX/Linux Only

IBM Informix Enterprise Gateway provides a single, standards-based gateway to multiple data sources. Gateway Manager connects the Informix

environment with that of any shared-library ODBC Level 2-compliant driver manager and driver on UNIX or Linux. For instance, you can use Gateway Manager with the IBM Informix Enterprise Gateway driver products to access UNIX or Linux database server products such as SYBASE SQL Server 10 and ORACLE7 Server. For more information, refer to the *IBM Informix: Enterprise Gateway Manager User Manual*.

_____ **End of UNIX/Linux Only** _____

Part 2. Migration to a Later Version of a Database Server

Chapter 3. Preparing for Migration

Preparing for Migration.	3-1
Diagnostic Information That You Need Before Upgrading	3-2
Migration Guidelines	3-3
Changes in Dynamic Server 10.0, 9.40, 9.30, and 9.2x	3-4
Environment Variables	3-4
Configuration Parameters	3-6
SQL Reserved Words	3-12
System Catalog and sysmaster Changes	3-14
Feature Changes.	3-19
New Features.	3-23
Storage-Manager Validation and Installation	3-35
Migrating Between 32-bit and 64-Bit Database Servers	3-36

In This Chapter

This chapter contains the information you need to know before migrating to Dynamic Server 10.0.

You can convert to Dynamic Server 10.0 from the following database servers:

- Dynamic Server 9.40
- Dynamic Server 9.30
- Dynamic Server 9.21
- Dynamic Server 9.20
- Dynamic Server 7.31
- Dynamic Server 7.30
- Dynamic Server 7.24 (without Enterprise Replication)

If necessary, you can revert to the database server that you converted from.

Important: To convert from another database server, including Universal Server 9.14, you need to convert to an intermediate database server first. For information on which intermediate database servers to use, see “Database Server Migration Paths” on page 1-7.

Preparing for Migration

This section contains:

- Details about the diagnostic information that you need to gather before you upgrade to another version of Dynamic Server

- General guidelines that you must follow when migrating between database servers.
- Information on new features that might affect migration.

Diagnostic Information That You Need Before Upgrading

Before you upgrade to a newer version of Dynamic Server, get the diagnostic information shown in Table 3-1. The information that you gather will be useful if you have problems or issues after the upgrade and need help from Technical Support. If you have problems, the information that you gather can be compared to information obtained after the upgrade.

You can print the pages containing Table 3-1 and then use the table as a checklist. After you get the information specified in each row, check the second column of the row.

Table 3-1. Checklist of Information to Get Before Upgrading

Information That You Need to Get	Check Here to Indicate You Have the Information
Get the SQL query plans for all regularly used queries, especially complex queries, using SET EXPLAIN ON.	____ Yes, I did this.
Run dbschema -d -hd for all critical tables. The output contains distribution information.	____ Yes, I did this.
Get oncheck -pr output that dumps all of the root reserved pages.	____ Yes, I did this.
Make a copy of the ONCONFIG configuration file. A copy of this file is useful because oncheck -pr does not dump all of the configuration parameters. In addition, a copy of the ONCONFIG file is essential if you need to revert to an earlier version of Dynamic Server.	____ Yes, I did this.
Prepare a list of all the environment variables that are set using the env UNIX command.	____ Yes, I did this.
During times of peak usage, obtain: <ul style="list-style-type: none"> • an online.log snippet, with some checkpoint durations in it • onstat -aF, -g all, and -g stk all 	____ Yes, I did this.

Table 3-1. Checklist of Information to Get Before Upgrading (continued)

Information That You Need to Get	Check Here to Indicate You Have the Information
<p>During times of peak usage, run the following onstat commands repeatedly with the -r repeat option for a period of about three to five minutes:</p> <ul style="list-style-type: none"> • onstat -u, to see the total number of sqlexecs used • onstat -p, for read and write cache rates, to detect deadlocks and the number of sequential scans • onstat -g nta, a consolidated output of -g ntu, ntt, ntm and ntd • onstat -g nsc, -g nsd, and -g nss for the status of shared memory connections • onstat -P, -g tpf, and -g ppf • vmstat, iostat and sar, for cpu utilization <p>Run these repeatedly for a period of about 3 to 5 minutes. Check pages for usage.</p> <ul style="list-style-type: none"> • timex of all queries that you regularly run 	<p>_____ Yes, I did this.</p>

Note: For 32- to 64-bit engine upgrades, change SHMBASE and STACKSIZE according to the **onconfig.std** configuration file for the new version. Also, look in Table 3-3 on page 3-6 or in the look in **onconfig.std** file for configuration parameters that are introduced with the new version of Dynamic Server.

Migration Guidelines

Observe the following guidelines when you migrate to Dynamic Server 10.0:

- Check the release notes for information about the correct operating-system release and any patches that you need for successful installation and operation of the database server.

The release notes are in one of the following directories:

- The **Informix** folder on Windows
 - To display the release notes, choose **Start > Programs > Informix Dynamic Server 10.0 > Release Notes** from the taskbar.
- **\$INFORMIXDIR/release/en_us/0333** on UNIX or Linux
- On UNIX or Linux, retain both versions of the IBM Informix product software on disk, if you have enough disk resources. You cannot retain both versions of the IBM Informix product software on disk on Windows.
- Retain the installation media from both versions of the IBM Informix product software.

- Before you convert to the target database server from the source database server, make sure that no open transactions exist in the source database server.

Fast recovery will fail when rolling back open transactions during the conversion. For information about how to close the source database server properly prior to conversion, see “Close All Transactions and Shut Down the Source Database Server” on page 5-6.

- **Important:** Before migration, you must perform a level-0 backup of all storage spaces with the source database server. After you complete the migration, perform another level-0 backup with Dynamic Server 10.0. You need to make a backup with each database server because you can restore a backup only with the same version of the database server on which you did the backup.
- It is recommended that you use a test instance of your database server to test the new version of the database server.
- Verify storage-manager validation for the target database server. For details, see “Storage-Manager Validation and Installation” on page 3-35.
- After you migrate and start using Version 10.0 consider the reversion issues described in Chapter 6, “Reverting from Dynamic Server 10.0,” on page 6-1, in case you need to revert. For example, any new in-place alter that you perform in using Version 10.0 must be completed before you revert.

For additional installation information and guidelines, refer to your *IBM Informix: Installation Guide* and your *IBM Informix: Getting Started Guide*.

Changes in Dynamic Server 10.0, 9.40, 9.30, and 9.2x

This section describes changes in Dynamic Server 10.0:

- New and changed environment variables
- New and changed configuration parameters
- New SQL reserved words
- System catalog and **sysmaster** changes
- Feature changes
- New features

Environment Variables

Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20 introduced new environment variables that might affect your installation. You might also need to adjust the values of existing environment variables. For more information on environment variables, see the *IBM Informix: Guide to SQL Reference* and your *IBM Informix: Dynamic Server Administrator's Guide*.

Table 3-2 lists the new environment variables in Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20.

Table 3-2. New Environment Variables

Version	Environment Variable	Description
10.0	IFX_EXTDIRECTIVES	A client-side external optimizer directive to use as a temporary solution to problems when you do not want to change SQL statements in queries
10.0	IFX_NO_TIMELIMIT_WARNING	A variable that supports time-limited license
10.0	IFX_ONPLOAD_AUTO_UPGRADE	A variable that enables you 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
10.0	STDIO	A TAPEDEV configuration parameter variable that improves the speed of HDR setup
9.40	CDR_LOGDELTA	Determines when spooling of the Enterprise Replication queue occurs, based on the percentage of the logical log size. Use as directed by Technical Support.
9.40	CDR_PERFLOG	Enables Enterprise Replication queue tracing. Use as directed by Technical Support.
9.40	CDR_ROUTER	Determines whether intermediate processing for Enterprise Replication is allowed in a hierarchal topology. Use as directed by Technical Support.
9.40	CDR_RMSCALEFACT	Sets the maximum number of Enterprise Replication DataSync threads per CPU VP. Use as directed by Technical Support.
9.40	USETABLENAME	Disallows the use of a synonym of the table in certain SQL statements.
9.30	IFX_DEF_TABLE_LOCKMODE	Specifies the default lock mode for database tables.
9.21	AFDEBUG	Forces the database system to hang when a failure occurs.
9.21	JAR_TEMP_PATH	Specifies a non-default local file system location for temporary .jar files of the Java virtual machine.
9.21	JAVA_COMPILER	Disables JIT compilation.

Table 3-2. New Environment Variables (continued)

Version	Environment Variable	Description
9.21	JVM_MAX_HEAP_SIZE	Sets a non-default upper limit on the size of the heap for the Java virtual machine.
9.20	IFX_LONGID	Determines whether a given client application is capable of handling long identifiers.
9.20	IFX_UPDDESC	Allows the execution of a DESCRIBE of an UPDATE statement.
9.20	STMT_CACHE	Controls the use of the shared statement cache on a session.

In Dynamic Server 9.30, the environment variable **DELIMIDENT** must be set before a client starts to manipulate a table with an SQL DELETE statement that omits the FROM keyword.

Configuration Parameters

Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20 introduced new configuration parameters that might affect your installation. You might also need to adjust the values of existing parameters. For more information on configuration parameters, see the *IBM Informix: Dynamic Server Administrator's Reference* and your *IBM Informix: Dynamic Server Administrator's Guide*.

Table 3-3 lists the new configuration parameters in Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20. All parameters are located in the ONCONFIG file, unless otherwise noted.

Table 3-3. New Configuration Parameters

Version	New Configuration Parameter	Description
10.0	ALRM_ALL_EVENTS	Specifies whether ALARMPROGRAM runs for all events that are logged in the MSGPATH or only specified noteworthy events.
10.0	BUFFERPOOL	Specifies configuration information for a buffer pool for each different page size used by a dbspace.
10.0	CDR_SUPPRESS_ATSRISWARN	Enterprise Replication configuration parameter that specifies whether comma-separated error and warning numbers are suppressed from ATS and RIS files.

Table 3-3. New Configuration Parameters (continued)

Version	New Configuration Parameter	Description
10.0	DR_IDXAUTO	Determines how a secondary database server reacts to a high-availability data-replication failure.
10.0	DS_NONPDQ_QUERY_MEM	Enables you to increase the amount of sort memory that is available for a query that is not a PDQ query.
10.0	EXT_DIRECTIVES	An external optimizer directive that provides a temporary solution to problems when you do not want to change SQL statements in queries
10.0	FAST_RESTART_CKPT_FUZZYLOG	Enables the flushing of dirty-pages table (DPT) records to the physical log on fuzzy checkpoints during the roll-forward phase of recovery. Like FAST_RESTART_PHYSLOG, this parameter decreases recovery time.
10.0	FAST_RESTART_PHYSLOG	Enables the database server to perform physical logging on fuzzy checkpoints during the roll-forward phase of recovery.
10.0	IFX_EXTEND_ROLE	Enables a database server administrator (DBSA) to prevent unauthorized users from registering DataBlade user-defined routines (UDRs).
10.0	LISTEN_TIMEOUT	Sets the incomplete connection timeout period. The default value is 10 seconds.
10.0	MAX_INCOMPLETE_CONNECTIONS	Restricts the number of incomplete requests for connections.
10.0	ONLIDX_MAXMEM	Limits the amount of memory that is allocated to the <i>preimage</i> log pool and to the <i>updater</i> log pool in shared memory. You can use this configuration parameter if you plan to complete other operations on a table column while executing the CREATE INDEX ONLINE statement on the column.
10.0	TBLTBLFIRST	Specifies the first extent size of tblspace tblspace in kilobytes.
10.0	TBLTBLNEXT	Specifies the next extent size of tblspace tblspace in kilobytes.
9.40	ASF_SOCTCP_BACKLOG	Specifies the number of connections queuing when using Sockets (SOCTCP).

Table 3-3. New Configuration Parameters (continued)

Version	New Configuration Parameter	Description
9.40	CDR_DBSPACE	Defines the default dbspace for the Enterprise Replication syscdr database.
9.40	CDR_ENV	Sets Enterprise Replication environment variables CDR_LOGDELTA, CDR_PERFLOG, CDR_ROUTER, and CDR_RMSCALEFACT.
9.40	CDR_MAX_DYNAMIC_LOGS	Specifies the number of dynamic log file requests that Enterprise Replication can make in one server session.
9.40	ENCRYPT_CDR	Enables and sets the level of network encryption for Enterprise Replication.
9.40	ENCRYPT_CIPHER	Specifies the ciphers to use for encryption for Enterprise Replication.
9.40	ENCRYPT_MAC	Specifies the level of message authentication coding to use for Enterprise Replication.
9.40	ENCRYPT_MACFILE	Specifies MAC key files for Enterprise Replication.
9.40	ENCRYPT_SWITCH	Defines the frequency at which ciphers and secret keys are re-negotiated for Enterprise Replication.
9.40	HPL_DYNAMIC_LIB_PATH	For the High-Performance Loader, sets the location of the shared-library file containing custom-code functions. Located in the plconfig file.
9.40	HPLAPIVERSION	For the High-Performance Loader, sets whether custom-code functions can use different input and output data lengths. Located in the plconfig file.
9.40	PLOG_OVERFLOW_PATH	Sets the location of the temporary space to extend the physical log during fast recovery.
9.30	CDR_QHDR_DBSPACE	Specifies the dbspace Enterprise Replication uses to store the transaction record headers spooled from the send and receive queues.
9.30	CDR_QDATA_SBSpace	Specifies the sbspace Enterprise Replication uses to store spooled transaction row data.
9.30	DEF_TABLE_LOCKMODE	Sets the lock mode to row for every newly created table.

Table 3-3. New Configuration Parameters (continued)

Version	New Configuration Parameter	Description
9.30	DYNAMIC_LOGS	Controls dynamic log allocation. This parameter is not in the default ONCONFIG file.
9.30	S BSPACETEMP	Specifies the name of the default temporary sbspace for storing temporary smart large objects.
9.21	DS_HASHSIZE	Specifies the number of hash buckets in the data-distribution cache for statistics generated by the UPDATE STATISTICS statement.
9.21	DS_POOLSIZE	Specifies the maximum number of entries in each hash bucket for statistics generated by the UPDATE STATISTICS statement.
9.21	STMT_CACHE_HITS	Specifies the number of references to a statement before it is fully inserted in the SQL statement cache.
9.21	STMT_CACHE_NOLIMIT	Controls whether to insert qualified statements into the SQL statement cache after its size is greater than the STMT_CACHE_SIZE value.
9.21	STMT_CACHE_NUMPOOL	Specifies the number of memory pools for the SQL statement cache.
9.20	ALLOW_NEWLINE	Controls whether the database server allows the newline character (\n) in a quoted string.
9.20	DD_HASHMAX	Specifies the maximum number of tables in each hash bucket in the data-dictionary cache.
9.20	DD_HASHSIZE	Specifies the number of hash buckets or lists in the datadictionary cache.
9.20	OPT_GOAL	Specifies optimization goals.
9.20	PC_HASHSIZE	Specifies the number of hash buckets in the caches that the database server uses.
9.20	PC_POOLSIZE	Specifies the maximum number of entries in several memory caches that the database server uses.
9.20	STMT_CACHE	Determines whether the database server uses the SQL statement cache.

Table 3-3. New Configuration Parameters (continued)

Version	New Configuration Parameter	Description
9.20	STMT_CACHE_SIZE	Specifies the size of the SQL statement cache.
9.20	SYSSBSPACENAME	Specifies the name of the sbspace for statistics collected by the UPDATE STATISTICS statement for certain user-defined data types.

Table 3-4 lists the configuration parameters that were altered or removed in Dynamic Server 10.0, 9.40, and 9.30. No changes were made to configuration parameters in Dynamic Server 9.21 and 9.20.

Table 3-4. Altered Configuration Parameters

Version	Altered Configuration Parameter	Description of Change
10.0	BUFFERS	Removed. Information now specified with the BUFFERPOOL configuration parameter.
10.0	LRUS	Removed. Information now specified with the BUFFERPOOL configuration parameter.
10.0	LRU_MAX_DIRTY	Removed. Information now specified with the BUFFERPOOL configuration parameter.
10.0	LRU_MIN_DIRTY	Removed. Information now specified with the BUFFERPOOL configuration parameter.
9.40	ALARMPROGRAM	Can be set to the alarmprogram.sh file to enable event alarms.
9.40	CDR_QDATA_SBSpace	Can accept up to 32 sbspaces.
9.40	CDR_QDATA_SBFLAGS	Removed. Enterprise Replication always uses the default log mode of the sbspace for spooling row data.
9.40	DBSERVERALIASES	Can accept up to 32 server alias values.
9.40	LTAPEBLK	The default value is 32 KB.
9.40	LTAPESIZE	Can accept a value of 0 to read or write to the end of the tape device.
9.40	LRU_MAX_DIRTY	Can accept a value of type INTEGER or FLOAT. (This configuration parameter was removed in Version 10.0.)

Table 3-4. Altered Configuration Parameters (continued)

Version	Altered Configuration Parameter	Description of Change
9.40	LRU_MIN_DIRTY	Can accept a value of type INTEGER or FLOAT. (This configuration parameter was removed in Version 10.0.)
9.40	OPTICAL_LIB_PATH	Is valid for both UNIX and Windows. Must be set to the location of the storage manager library.
9.40	TAPEBLK	The default value is 32 KB.
9.40	TAPESIZE	Can accept a value of 0 to read or write to the end of the tape device.
9.30	AFF_NPROCS	Removed; superseded by the VPCLASS configuration parameter.
9.30	AFF_SPROC	Removed; superseded by the VPCLASS configuration parameter.
9.30	CDR_LOGBUFFERS	Removed.
9.30	CDR_LOGDELTA	Removed.
9.30	CDR_NIFRETRY	Removed.
9.30	CDR_NUMCONNECT	Removed.
9.30	JDKVERSION	New default value of 1.3.
9.30	JVPJAVAHOME	New default value of /usr/informix/extend/krakatoa .
9.30	JVPJAVALIB	New default value that is platform dependent.
9.30	JVPJAVAVM	New default value that is platform dependent.
9.30	LBU_PRESERVE	Removed; configured an obsolete utility.
9.30	LOGSMAX	Removed.
9.30	LTXHWM	Removed from the default ONCONFIG file, but is still valid.
9.30	LTXEHWM	Removed from the default ONCONFIG file, but is still valid.
9.30	NOAGE	Removed; superseded by the VPCLASS configuration parameter.
9.30	NUMAIOVPS	Removed; superseded by the VPCLASS configuration parameter.
9.30	NUMCPUVPS	Removed; superseded by the VPCLASS configuration parameter.

SQL Reserved Words

Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20 support new SQL keywords that are reserved words and might affect migration of your applications. Although you can use almost any word as an SQL identifier, syntactic ambiguities might occur if you use an SQL reserved word. An ambiguous statement might not produce the results you want.

The list below shows new SQL reserved words. For more information about SQL reserved words, see the *IBM Informix: Guide to SQL Syntax*.

Version 10.0:

- ACTIVE
- AVERAGE
- DATASOURCE
- DIRECTIVES
- ENCRYPTION
- EXTEND
- FALSE
- HINT
- INACTIVE
- INLINE
- INOUT
- NONE
- ONLINE
- OPTCOMPIND
- PARTITION
- PASSWORD
- REUSE
- SAVE
- STORAGE
- TEMPLATE
- TEST
- TRUE
- TRUNCATE
- TYPEID
- TYPENAME
- TYPEOF
- UPPER
- XADATASOURCE

- XID

Version 9.40

- COLLATION
- CROSS
- FULL
- INSTEAD
- RESTART
- RIGHT

Version 9.30

- AVOID_EXECUTE
- AVOID_SUBQF
- USE_SUBQF

Version 9.21

- RAW
- STANDARD

Version 9.20

- AGGREGATE
- CACHE
- COSTFUNC
- ITEM
- NAME
- REF
- SELCONST

Version 9.20 and 7.31

- INNER
- JOIN
- LEFT
- LOCKS
- RETAIN

Version 9.20 and 7.30

- ALL_ROWS
- CASE
- CRCOLS

- DECODE
- FIRST_ROWS
- MEMORY_RESIDENT
- NON_RESIDENT
- NVL
- REPLICATION
- SUBSTR
- SUBSTRING

System Catalog and sysmaster Changes

The system catalog tables and **sysmaster** database are different from those for database servers earlier than Dynamic Server 9.20, which includes changes to some column widths, data types, and treatment of null values. Also, some tables have additional columns, and some tables were added or deleted.

Remote Queries on System Catalog Tables Between 7.x and 9.x or later:

Certain system catalog tables use data types in Dynamic Server 9.x or later that are not supported in Dynamic Server 7.x. Remote queries that issue a `SELECT *` statement on these system catalog tables from Dynamic Server 7.x to Dynamic Server 9.x or later will fail.

For example, the following queries that originate on a Dynamic Server 7.x fail when executed on Dynamic Server 9.x or later:

```
SELECT * FROM dbname@remoteserver:sysindices;
SELECT * FROM dbname@remoteserver:sysindexes;
```

Instead of using an asterisk as the Projection clause, specify the required column names explicitly. You cannot specify any columns that have user-defined types.

Difference in sysindexes: In Version 7.x, **sysindexes** is a table. In Dynamic Server 10.0, 9.40, 9.30, and 9.2x, **sysindexes** is a view. The `ALTER TABLE` statement fails for **sysindexes** because this statement is not valid for altering a view.

Column-Width Changes: Version 9.20 and later versions of Dynamic Server provide long identifiers. All identifiers in the system catalog tables and the **sysmaster** database reflect these new limits on identifier length. The *IBM Informix: Guide to SQL Syntax* defines *identifiers*, which specify the names of database objects.

The column widths for identifiers that refer to database objects and other identifiers changed from `CHAR(18)` to `VARCHAR(128,0)` in the following system catalog tables:

sysaggregates	sysfragauth	sysroutinelangs
sysams	sysfragments	sys synonyms
sysattrtypes	sysindexes	sys syntable
sysblobs	sysindices	sys tabamdata
syscasts	sysobjstate	sys tables
syscolattribs	sysopclasses	sys tracemsgs
syscolumns	sysopclstr	sys triggers
sysconstraints	sysprocedures	sys xdtypes
sysdomains		

Identifiers changed from CHAR(18) to CHAR(128) in the following **sysmaster** database tables:

arc_dbSPACE	syscrtadt	syslocks
arc_dbSPACE_set	sysdatabases	sysopendb
arc_phys_dev	sysdbslocale	sysprc
arc_rep_table	sysdbspaces	sysproccache
arc_replicate	sysdbspartn	sysptprof
arc_server	sysdbstab	sys sdblock
arc_version	sysdic	sys sqlcurall
arc_vset	sysdiccache	sys sqlstat
arc_vset_view	sysdistcache	sys sqlstat
flags_text	sysdsc	sys tabnames
syscfgtab	sys extents	sys trans
sysconfig	sys extspaces	sys xptab

Column widths for user login identifiers changed from CHAR(8) to CHAR(32) in some system catalog tables and **sysmaster** database tables. The following system catalog tables changed:

sysaggregates	sysindices	sys synonyms
sysams	syslangauth	sys syntable
syscasts	sysobjstate	sys tabauth
syscolauth	sysopclasses	sys tables
sysconstraints	sysopclstr	sys triggers
sysdomains	sysprocauth	sys users
sysfragauth	sysprocedures	sys xdtypeauth
sysindexes	sysroleauth	sys xdtypes

The following **sysmaster** database tables changed:

sysaudit	sysdiccache	sysrstcb
sysdatabases	sysdistcache	syssslst
sysdbspaces	sysdsc	sysessions
sysdbspartn	sysextspaces	systabnames
sysdbstab	sysprc	sysuserthreads
sysdic	sysproccache	

Columns that include pathnames or other values changed from CHAR(128) to CHAR(256) in the following **sysmaster** database tables:

sysadinfo	sysrctadt	sysmchktab
syschktab	sysdrbc	
syschunks	sysdri	

The path for a physical device changed from CHAR(128) to CHAR(260) in the following **sysmaster** database table:

arc_phys_dev

Columns widths changed from CHAR(20) to CHAR(128) for longer object names in the following **sysmaster** database tables:

sysdrbc
sysdri

Column widths changed from CHAR(37) to CHAR(257) in the following **sysmaster** database tables:

sysdistcache	sysprc
sysdsc	sysproccache

Column widths changed from DECIMAL(16,0) to DECIMAL(32,0) in the following **sysmaster** database table:

sysesprof

The **tabauth** column of the **systabauth** system catalog table is now CHAR(9) instead of CHAR(8). The 9th character indicates the Under privilege.

Data Type Changes: The preceding section on column-width changes lists columns that have changed from the CHAR data type to the VARCHAR data type.

One or more columns changed from the SMALLINT data type to the integer data type in the following **sysmaster** database tables:

sysdbspaces	sysdic	sysddbblock
sysdbstab	sysrstcb	

The CHAR data type changed to the STAT data type in the following system catalog table:

sysdistrib

Changes in Treatment of Null Values: No nulls allowed changed to nulls allowed for some columns in the following **sysmaster** database tables:

arc_ae_view	arc_pendreq_view	arc_volume_view
arc_db_file_view	arc_req_vset_view	arc_vset_user_view
arc_directory_view	arc_request_view	arc_vset_view
arc_file_copy_view	arc_save_set_view	
arc_file_view	arc_vol_lock_view	

Columns Added: One or more columns have been added to the following system catalog tables:

sysams	sysprocedures
sysdistrib	sysroutinelangs

Several columns have been added to the following **sysmaster** database table:

sysdbstab

Tables Added or Deleted: The following tables have been added to the **sysmaster** database:

logmessage	syscdrctl_txn	syscdrrecv_txn
syscdrack_buf	syscdrprog	syscdrtx
syscdrack_txn	syscdrq	
syscdrctl_buf	syscdrrecv_buf	

The following table has been deleted from the **sysmaster** database:

arc_change_log

Changes for Dynamic Server 10.0: The **sysdirectives** system catalog table was added.

The **sysbufpool** system-monitoring interface (SMI) table was added, and the following changes were made to other SMI tables:

- The **sysfragments** table contains a **Partition** column and the **Flags** column now tells you if the fragmentation scheme has partitions.
- The **sysusers** table contains a **defrole** column.
- The **sysams** table contains an **am_truncate** column.
- The **sysprocedures** table contains a **rtnparameters** column for information on INOUT parameters.
- The **syspaghdr** table has a **pg_page_size** column.
- The **sysptnhdr** table has a **page_size** column.
- The **sysptnhdr** table has a **bpoolindx** column that indicates which buffer pool the buffer is in
- The **sysbufhdr** table has a **bufsize** column, which indicates the buffer page size
- The **sysdbstab** and **syschktab** tables have **page_size** columns.
- The views **syschunks** and **sysdbspaces** tables have a **page_size** columns.
- The views **sysstabinfo** table has a **ti_page_size** column.
- The views **sysstabpaghdrs** and **sysphyspaghdrs** tables have **pg_page_size** columns.

In addition, tables added to the **syscdr** database are removed.

Changes for Dynamic Server 9.40: The following new system catalog table was added:

syssequences

A new **collation** column has been added to the following system catalog tables:

sysconstraints	sysprocplan	systrigbody
sysindices		

Changes for Dynamic Server 9.30: The following tables were deleted from the **sysmaster** database in Version 9.30:

arc_ae_view	arc_pendreq_view	arc_server
arc_db_file_view	arc_phys_dev	arc_version
arc_dbspace	arc_rep_table	arc_vol_lock_view
arc_dbspace_set	arc_replicate	arc_volume_view
arc_directory_view	arc_req_vset_view	arc_vset
arc_file_view	arc_request_view	arc_vset_user_view
arc_file_copy_view	arc_save_set_view	arc_vset_view

Feature Changes

The following feature changes are in Dynamic Server 10.0:

- Removal of the BUFFERS, LRUS, LRU_MIN_DIRTY, and LRU_MAX_DIRTY configuration parameters
- Addition of the BUFFERPOOL configuration parameter, which allows you to specify the buffer pool size as well as information previously specified with the BUFFERS, LRUS, LRU_MIN_DIRTY, and LRU_MAX_DIRTY configuration parameters
- Changes to the ALARMPROGRAM, SHMADD and SHMVIRTSIZE configuration parameters

In addition, the DRAUTO configuration parameter, which was removed from Dynamic Server in Version 9.3, is again included in the ONCONFIG file. For a description of this parameter, see “The DRAUTO Configuration Parameter” on page 3-19.

The following feature changes are in Dynamic Server 10.0 as well as Dynamic Server versions 9.40, 9.30, and 9.2x:

- Detached indexes
- Changes in ON-Bar commands
- Chunk format
- Libraries no longer installed in the /usr/lib directory
- VARCHAR column limit
- Stored procedure parameter limit
- Limitation on using routines in distributed transactions
- Case-sensitive name space
- New administration tools
- Management of the SQLHOSTS connectivity information on Windows
- Maximum index key size difference between 9.x or later and 7.3

The DRAUTO Configuration Parameter: The DRAUTO configuration parameter was present in some earlier versions of Dynamic Server, including Version 7.3. It was removed from Version 9.3, but is included in Version 10.0. This parameter determines how a secondary database server reacts to a high-availability data-replication (HDR) failure.

If the DRAUTO configuration parameter in the ONCONFIG file of the secondary database server in an HDR pair is set to 1 (RETAIN_TYPE) or 2 (REVERSE_TYPE) and the primary database server in the pair fails, the secondary database server automatically becomes the standard database server. It first rolls back any open transactions and then comes into on-line mode as the standard database server. For more information on using this parameter, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Detached Indexes: By default, all new indexes that the CREATE INDEX statement creates in Dynamic Server 10.0, 9.40, 9.30, or 9.2x are detached and stored in separate tablespaces from the data (unless the deprecated "IN TABLE" syntax is specified). Indexes created in Version 7.x are attached until you rebuild them.

You cannot revert detached indexes to Version 7.x. To enable reversion to Version 7.x, retain the Version 7.x attached index behavior by setting the environment variable **DEFAULT_ATTACH** in the application environment. You can attach only B-tree indexes that are nonfragmented and that are on nonfragmented tables (the Version 7.x behavior). All other indexes, including extensibility related indexes such as R-trees and UDT indexes, must be detached.

Changes in ON-Bar Commands: Between Dynamic Server 7.x and Dynamic Server 9.x and later, the syntax of the following ON-Bar commands changed:

Dynamic Server 7.x: **onbar -r -v**

Dynamic Server 9.x or later: **onbar -v**

Dynamic Server 7.x: **onbar -l**

Dynamic Server 9.x or later: **onbar -b -l**

Chunk Format: Dynamic Server 9.40 introduced for the following chunk features that cannot be reverted:

- Chunks greater than 2 GB
- Chunk offsets greater than 2 GB
- Greater than 2047 chunks

To facilitate backward compatibility, these features are not enabled by default. To enable these features, use the **onmode -BC** command.

The **onmode -BC** command has the following options that allow you to stage your migration to these features:

- **onmode -BC 1**

Allows chunks and offsets of greater than 2 GB to be created, but dbspaces and blobspaces that do not contain large chunks remain in the Version 9.3 format. After a chunk larger than 2 GB is added to a dbspace or blobspace, then all chunks added or altered in that dbspace or blobspace are in the new format. To revert, drop any dbspaces or blobspaces using the new chunk features.

- **onmode -BC 2**

Enables the Version 9.4 chunk features for all dbspaces and blobspaces. Any chunk or offset added or modified has the new format. All page writes are in the new format, including those to chunks created prior to Version 9.4. Reversion is no longer possible.

Libraries No Longer Installed in the /usr/lib Directory: Dynamic Server 9.40 and 10.0 do not include the following libraries in the **/usr/lib** directory on UNIX:

- Optical storage manager library

If you use an optical storage manager, set the `OPTICAL_LIB_PATH` configuration parameter in the `ONCONFIG` file to the location of the optical storage manager library. For more information, see the *IBM Informix: Optical Subsystem Guide*.

- High-Performance Loader custom-code shared library

If you use custom-code files with the High-Performance Loader, set the `HPL_DYNAMIC_LIB_PATH` configuration parameter in the `plconfig` file to the location of the shared library. The default location of this library is `$INFORMIXDIR/lib/ipldd09a.so`. For more information, see the *IBM Informix: High-Performance Loader User's Guide*.

VARCHAR Column Limit: If you are migrating from Dynamic Server 7.3x, the number of VARCHAR columns per table for Dynamic Server 10.0, 9.40, 9.30, or 9.2x has been reduced from 231 to 195, within a row size of 32,762 bytes and based on a page size of 2 kilobytes. Before you upgrade to Dynamic Server 10.0, ensure that your tables obey this restriction.

Stored Procedure Parameter Limit: Dynamic Server 10.0, 9.40, 9.30, or 9.2x have a limit of 341 parameters for each stored procedure.

Limitation on Using Routines in Distributed Transactions: User-defined routines and built-in routines, such as `round()`, cannot be used in distributed transactions involving Dynamic Server 9.2x or 9.30. For Dynamic Server 9.40 and 10.0, routines in distributed transactions are only supported among Version 9.40 and 10.0 database servers.

Case-Sensitive Name Space: If you have case-insensitive schemas, you might need to revise them because Dynamic Server 10.0, 9.40, 9.30, or 9.2x have a case-sensitive name space. This can affect the resolution of blobspaces and SPL names.

New Administration Tools: Dynamic Server 10.0, 9.40, 9.30, and 9.21 do not support IECC. The IECC functionality for these database servers comes from the following sources:

- Server Studio JE
- IBM Informix Server Administrator (ISA)

Server Studio JE: Included in the IBM Informix Dynamic Server bundle is a CD containing Server Studio JE (Java Edition). Server Studio JE is the result of an extensive collaboration between IBM and Advanced Global Systems LTD (AGS).

Before using any Server Studio features, you must first register with AGS. When you use Server Studio to build the first connection to a database server, you will find information on how to register. Once registered, you will receive an email with a certificate to install. You must install this certificate to access Server Studio features.

Once you have registered, you can use the following features for an unlimited period of time:

- Object Explorer
- Schema Editor
- SQL Editor

In addition, during the first 30 days of use, several other modules are enabled on a trial basis. After the 30-day-trial period, the unlicensed add-on modules will not work and will generate expiration messages. To continue using the add-on modules after 30 days, you must obtain a license for those modules directly from Advanced Global Systems LTD (AGS).

Some of the add-on modules that are available inside Server Studio JE are:

- Enhanced SQL Editor
- Database Difference Analyzer
- Dependency Analyzer
- Permission Manager
- XPS 8.3 connectivity
- OnLine 5 connectivity

For a complete list of the currently available add-on modules visit the AGS website at www.agsltd.com.

IBM Informix Server Administrator: IBM Informix Server Administrator (ISA) is a browser-based tool that provides Web-based system administration for all Informix 10.0, 9.40, 9.30, 9.2x, 8.40, 8.3x, 8.21, 7.3x, and 7.24 database servers. ISA provides access to database server command-line functions and presents the output in an easy-to-read format.

With ISA, you can use a browser to perform these common database-server administration tasks:

- Check dbspaces, database catalogs, logs, and other objects.

- Change configuration parameters temporarily or permanently.
- Manage logical and physical logs.
- Examine memory use and add free memory segments.
- Read the message log.
- Change the database server mode between online and offline and intermediate states.
- Back up and restore dbspaces and logical logs.
- Run **onstat** commands to monitor performance.
- Enter SQL statements and examine database schemas.
- Examine and manage user sessions.
- Examine and manage Virtual Processors (VPs).
- Add and remove storage spaces, including spaces that are unique to Dynamic Server database servers.

The database server CD-ROM distributed with your product includes ISA. For information on how to install ISA, see the following file on the CD-ROM.

Operating System	File
UNIX or Linux	/SVR_ADM/README
Windows	\SVR_ADM\readme.txt

Management of the SQLHOSTS Connectivity Information on Windows:

On Windows, IBM Informix products store the SQLHOSTS information in registry keys. Windows provides the **regedt32** utility, which enables you to manage registry keys, but it is recommended that you do not use **regedt32**.

If you have used **Setnet32** to manage SQLHOSTS information, you can continue to do so; however, **Setnet32** does not enable you to assign a database server to a database server group.

For Enterprise Replication, use ISA.

Maximum index key size difference between 9.x or later and 7.3: Before reverting to Dynamic Server 7.3, you must drop any index whose key size is greater than 254. Dynamic Server 9.4 supports a maximum index key size of 255 for VARCHAR and NVARCHAR types. However, Dynamic Server 7.3 only supports a maximum index key size of 254.

New Features

This section lists new features in Dynamic Server 10.0, 9.40, 9.30, 9.21, and 9.20. Read your release notes and documentation notes for late-breaking information on new features.

New Features in Dynamic Server 10.0: Dynamic Server 10.0 introduces the following new features:

- Security enhancements
 - Server utilities check for a secure environment before starting on UNIX or Linux
 - External authentication on Windows, as well as UNIX and Linux operating systems
 - Column-level encryption
 - Multiple listener threads to use to reduce the risk of a hostile-denial of service attack
 - Configuration parameter for the Database Server Administrator (DBSA) to use to restrict users who can register DataBlade user-defined routines
- Database server usability enhancements
 - A new mode, single-user mode, for maintenance tasks
 - Default roles that can be assigned on a per-database level
 - Support for standard or temporary dbspaces with different page sizes, with an increase for the maximum key size (The configurable page size feature is off unless you enable it by creating larger pages. Before you create a larger page, you must use **onmode -BC** to enable the Dynamic Server large chunk feature.)
 - Additional functionality for managing the tblspace **tblspace**, including functionality for specifying the first and next extent sizes of tblspace **tblspace**.
 - Functionality for renaming dbspaces
 - An new alarm configuration parameter for specifying whether the event alarm program operates for specified events or all events that are logged in the MSGPATH
 - Functionality for specifying that segments for shared memory can be as large as your operating system platform or the SHMMAX parameter allows
 - Functionality for creating multiple partitions of a table or index within a single dspace for fragmented tables that use expression-based or round-robin distribution schemes
 - Functionality for setting up High-Availability Data Replication (HDR) using standard ON-Bar or **ontape** commands
 - Functionality for resending an index that became corrupt on the secondary server
 - A new **-version** option for all server utilities
 - Functionality for using the Internet Protocol Version 6 (IPv6) format for IP addresses
- Performance enhancements

- Functionality for specifying the amount of memory that is allocated for non-PDQ queries
- Ability to create, save, and reuse external optimizer directives
- New configuration parameters that enable the database server to perform physical logging on fuzzy checkpoints during the roll-forward phase of recovery
- A new command, `SET ENVIRONMENT OPTCOMPIND`, to use to set or modify the `OPTCOMPIND` value within a session
- New SQL statements, `CREATE INDEX ONLINE` and `DROP INDEX ONLINE`, to use to create or drop an index in an online environment without having an access lock on the table
- Interoperability enhancements
 - Functionality for running IBM Informix ESQL/C applications with DB2 servers and databases
- Enterprise Replication (ER) enhancements
 - Functionality for creating a replicate as a master replicate
 - A replicate template option to use when setting up a replication system
 - Functionality for performing an initial synchronization on data
 - New commands to use to alter replicated tables
 - An event alarm program that detects ER event alarms
- Backup and Restore enhancements
 - The **archecker** utility to use to recover specific tables from an archive
 - Functionality for viewing logical logs that were backed up by ON-Bar
 - Functionality for changing the ON-Bar debugging level
 - Functionality in the **ontape** utility to specify the use of standard I/O instead of a tape device or disk file
 - Functionality to use the **ontape** utility for external backup and restore procedures
- Storage enhancements
 - Support for longer object names in the **onpload** and **onpladm** High-Performance Loader (HPL) utilities
 - Inclusion of Tivoli Storage Manager XBSA with Dynamic Server
- Extensibility enhancements
 - Functionality for using built-in opaque data types in remote queries involving databases residing on the same database server
 - Functionality for creating user-defined routines that get information about triggers and are used in trigger action statements
- Installation enhancements
 - The display of a License Agreement that must be accepted

- A **/doc** directory that contains release notes, machine notes, documentation notes, and the *IBM Informix: Installation Guide* in PDF format
- The IBM Informix Server Instance Manager utility, which has an option to change the name of a Dynamic Server instance on Windows platforms
- The IBM Informix ClusterIT utility, which enables you to create the primary node for a Dynamic Server cluster and install and configure Dynamic Server as a secondary node in a cluster
- A custom installation option for choosing which Dynamic Server components to install
- Application Development enhancements
 - Support for updated JDBC 3.0 specification
 - The .NET Provider, which provides support for .NET applications

New Features in Dynamic Server 9.40: Dynamic Server 9.40 introduces the following new features:

- Security enhancements
 - Encryption for networked transactions
 - Authentication
- Database server usability enhancements
 - Chunks up to 4 TB
 - Chunk offsets up to 4 TB
 - New chunk limit of 32,766
 - Greater than 2 GB file size
 - The default size of the TAPEBLK and LTAPEBLK configuration parameters is 32 KB
 - Chunks can be added when the root chunk is full by storing metadata in extended chunk reserve pages allocated from non-root chunks in the root dbspace
 - Buffer cleaning can be tuned by setting the LRU configuration parameters to a value of type FLOAT (This configuration parameter was removed in Version 10.0.)
 - Event alarms can be set with the **alarmprogram.sh** modifiable script
 - Ability to specify up to 32 database server aliases with the DBSERVERALIASES configuration parameter
 - During fast recovery, the physical log space is extended if the physical log overflows
 - Microsoft Transaction Server/XA support
- Performance enhancements
 - PDQ is enabled for hold cursors

- B-tree cleaner improves transaction processing for logged databases when rows are deleted from a table with indexes
- Improved priority management for the buffer manager
- Reliability, availability, and supportability features
 - Ability to monitor queries dynamically using the **onmode -Y** command
 - Print the session control block address with the **onstat -g ses** command
 - Display the setting and values of environment variables with the **onstat -g env** command
 - Ability to specify the number of pages to print, whether to print just the page headers, and to print pages from chunks that are online with the **oncheck** utility
 - Display the types and values of host variables in SQL statements, show the stored procedure stack, and show the current SQL statement in a stored procedure using the **onstat -g sql** command
- SQL enhancements
 - Ability to specify an expression or a column in the ORDER BY clause of a SELECT statement that is not listed in the projection of the SELECT statement
 - Change the collation used by the session for comparisons and sorts on NCHAR and NVARCHAR objects
 - Functional indexes can be created on 102 parameters for C UDRs, or 341 parameters for SPL and Java UDRs
 - The LVARCHAR(*m*) data type can be set to sizes larger than 2 KB
 - INSTEAD OF triggers on views
 - Sequence objects
 - Additional in-place alters on clauses of the ALTER TABLE statement
 - DESCRIBE INPUT and DESCRIBE OUTPUT statements to return information about multiple input and output parameters
 - The SET EXPLAIN statement AVOID EXECUTE option displays the query plan without executing the query
 - ANSI join syntax support for cross, right outer, and full outer joins
 - Unions allowed in subqueries of SELECT statements
 - USETABLENAME environment variable to invalidate synonyms in ALTER TABLE and DROP TABLE statements
- GLS enhancements
 - Session-level non-default collation
 - Unicode and Unicode collation support
 - Support for the Chinese locale 18030-2000
- Enterprise Replication enhancements

- Encrypted transactions implemented with configuration parameters
- Support for replicating row data types and collection data types
- Faster queue recovery
- Replication during queue recovery
- Support for large transactions up to 4 TB
- Improved availability by combining High-Availability Data Replication (HDR) and Enterprise Replication
- Dynamic log file
- A **brief** option for the **cdr list replicate** command to display a summary of participants for all replicates
- The **cdr remove** command to remove Enterprise Replication from an HDR server
- The **CDR_QDATA_SBSPACE** configuration parameter now accepts up to 32 sbspace names. Enterprise Replication uses all sbspaces in round-robin order.
- The **CDR_DBSPACE** configuration parameter to specify the dbspace of the **syscdr** table
- The **CDR_ENV** configuration parameter to set Enterprise Replication environment variables
- The **CDR_LOGDELTA** environment variable to determine when the send and receive queues are spooled to disk
- The **CDR_PERFLOG** environment variable to enable queue tracing
- The **CDR_ROUTER** environment variable to disables intermediate acknowledgements of transactions in hierarchical topologies
- The **CDR_RMSCALEFACT** environment variable to set the number of DataSync threads started for each CPU VP
- Extensibility enhancements
 - Ability to create user-defined selectivity functions and to calculate the cost of using an R-tree index so as to allow the optimizer to make accurate decisions about which index to use
 - HDR support for extended data types and UDRs
 - Using an iterator function in the FROM clause of a SELECT statement and returning the result set using a virtual-table interface
 - Naming the return parameters of a UDR
 - Using multiple OUT parameters and statement local variables
- DataBlade API enhancements
 - Ability to return the value of the current database server locale with the **mi_get_db_locale()** function
 - Ability to return the ID of the current transaction with the **mi_get_transaction_id()** function

- Ability to change the size of an existing memory block with the **mi_realloc()** function
- Ability to determine whether the current user stack has the specified amount of free space with the **mi_stack_limit()** function
- Ability to execute operating system commands in a separate thread with the **mi_system()** function
- Stream support for files larger than 2 GB
- High-Performance Loader enhancements
 - Use the full capacity of a storage device
 - Ability to set the location of the custom-code shared library file with the **HPL_DYNAMIC_LIB_PATH** configuration parameter
 - Ability to use a different length for data in the input and output arguments of custom-code functions by setting the **HPLAPIVERSION** configuration parameter
- Backup and restore enhancements
 - Renaming chunks to a different path and offset during a cold restore
 - Using the full capacity of a storage device with **ontape**
- Installation enhancements
 - Components previously installed in **/usr/lib** are now installed in **\$INFORMIXDIR/lib**.
 - On UNIX, the installation program prompts the user to avoid overwriting existing client or GLS files that are more recent than those included with the database server.

New Features in Dynamic Server 9.30: Dynamic Server 9.30 has the following new features:

- The ability to display the maximum number of connections
- DataBlade API enhancements
 - New **PER_STMT_EXEC** and **PER_STMT_PREP** memory durations
 - The ability to use **mi_lo** routines without a connection
 - New **mi_collection_card()** function to return the cardinality for a collection (number of items in the collection)
 - Access to files on a client computer one buffer at a time
 - New **mi_transaction_state()** function to return the current transaction state (none, implicit, or explicit)
- Enterprise Replication enhancements
 - Improvements to parallel processing
 - External Enterprise Replication conversion
 - **SERIAL** column primary keys
 - Replicate sets

Pre-Version 9.30 replicate groups are not supported in Version 9.30. Before you migrate to Version 9.30, you must remove any replicate groups. For instructions, see Chapter 4, “Migrating to Dynamic Server 10.0 with Enterprise Replication,” on page 4-1.

- Replicating only changed columns
- Spooling changes
- In-place processing of ALTER statements to add or drop shadow columns
- Command-line changes to support new features
- Extensibility enhancements
 - New **deepcopy()** function for multirepresentational data types
 - Nearest neighbor queries in R-trees
 - Temporary sbspaces
 - Temporary smart large objects
 - Improved partitioning of user data and metadata in sbspaces
- Java Virtual Machine (JVM) 1.3 support
- Performance enhancements
 - The **onstat -g stm** option
 - The ability to display the query plan without executing the query
 - Optimizer directives for subqueries
 - Dynamic addition of logical logs
 - Performance improvement for smart large objects
- SQL enhancements
 - Configurable default lock modes
 - REVOKE...AS USER statement
 - DELETE statement that does not require the FROM keyword

New Features in Dynamic Server 9.21: Dynamic Server 9.21 introduced the following new features:

- Database server features
 - The **onpladm** utility for High-Performance Loader (HPL) jobs
 - SQL statement cache to store SQL statements for re-execution
 - Access to synonyms on remote 7.x database servers through DB–Access
- Extensibility features
 - C++ support for writing user-defined routines (UDRs) with fewer restrictions
 - DataBlade API functions for controlling the virtual processor environment
 - The **mi_fp_funcname()** function to get the SQL name of a function

- Java features
 - JVM 1.2 support
 - Changes in default values of Java configuration parameters
 - GLS support for J/Foundation
 - The **update_jars.sql** script
 - Run-time environment variables
 - JVP virtual-processor classes dropped dynamically
 - Support for variable-length opaque types and opaque-type send/receive, import/export, and importbin/exportbin functions
- Support for MaxConnect and new network protocols: **ontliimc** and **onsocimc**

New Features in Dynamic Server 9.20: Dynamic Server 9.20 introduced the following new features:

- Extensibility enhancements
 - Dynamic lock allocation
 - SQL enhancements
 - Embedded newline characters in quoted strings
 - Nested dot expressions for row types
 - Triggers on SELECT statements
 - Enhancements to smart large objects
 - Round-robin fragmentation for smart large objects
 - ALTER TABLE for smart large objects
 - Data type conversion: BYTE to BLOB and TEXT to CLOB
 - Change of sbspace characteristics (**onspaces -ch**)
 - Immediate deletion of smart large objects
 - Enhancements to collections
 - Collection constructors that use arbitrary expression elements
 - Collection-derived tables
 - Collection subqueries
 - Enhancements to row types
 - Serial types in row types
 - GRANT/REVOKE UNDER on row types
 - Enhancements to UDRs
 - Ability to write UDRs in the Java language
 - GRANT/REVOKE on UDR external languages
 - ALTER FUNCTION/PROCEDURE/ROUTINE statements
 - User-defined aggregates

- Extensions to the DataBlade API
 - Ability to obtain database server environment values with the **mi_get_serverenv()** function
 - Ability to obtain database connection information with the **mi_get_connection_option()** function
 - Ability to obtain a function descriptor from a routine identifier with the **mi_funcdesc_by_typeid()** function
 - Ability to obtain a DATETIME or INTERVAL qualifier from a type descriptor with the **mi_type_qualifier()** function
 - Ability to access a collection subquery with the **mi_collection_open_with_options()** function
 - Ability to delete a smart large object immediately with the **mi_lo_delete_immediate()** function
 - Removal of restrictions on callback extensions
- Extensions to the ON-Bar suite
 - onmsync** tool for object expiration and synchronization
 - onbar** tools: override (with -O option), progress feedback, restartable restore, and external backup and restore
- The **oncheck** and **onlog** utilities for R-tree indexes
- Enhancements to R-tree indexes
 - A support interface for bulk loading of R-tree indexes
 - A new SQL UDR that lets users access the bounding box of the root page of an R-tree index
- Performance improvements
 - Fuzzy checkpoints
 - Parallel statement-local variables (SLVs)
 - SQL statement cache
 - Compiled expressions
 - Improvements for smart large objects
 - Incremental backup support
 - Lightweight I/O
 - Metadata compaction
 - Byte-range locking
 - Dirty read
 - Improvements for UDRs
 - Expensive-function optimization
 - Parallel UDRs
 - Support for parallel UDR execution in HPL

- User-defined statistics routines
 - Parallel scan for Virtual-Table Interface (VTI) and Virtual-Index Interface (VII)
 - Set read and write for VTI and VII
- Enterprise Replication enhancements
 - Improvements in management of storage queues
 - Faster processing for large transactions
 - Better management to avoid long transactions
 - Improvements in the log reader
 - Updates to the command-line interface
 - Hierarchical routing; direct connections no longer required
 - Support for database servers with intermittent connections
 - Reduced global catalog available
 - New CONNECT and DISCONNECT functions in the Enterprise Replication Manager
- Special features
 - Long identifiers: 128-byte identifiers and 32-byte user names
 - Ability to retain update locks

Version 9.20 Features from Universal Server 9.14: Dynamic Server 9.20 includes the following features that were first released in Universal Server 9.1x:

- Lightweight I/O
- Dynamic addition of locks
- Nonroot execution of virtual processors (VPs)
- EXECUTE FUNCTION in FOREACH EXECUTE statement
- Re-entrant triggers
- Limit of SET SESSION AUTHORIZATION scope to the current database
- **INFORMIXCONTIME** environment variable, which needs to be set to a large number to make the DCE CSM work (not necessary for non-CSM environments)
- TP/XA support added
- ON-Bar support for GLS
- **oncheck** enhancement to display access-method data
- **onstat** enhancement to display the access-method cache
- New interface routines added to the Virtual Table Interface and Virtual Index Interface:
 - **mi_string *mi_qual_funcname(MI_AM_QUAL_DESC *qd)**
 - **mi_integer mi_scan_nprojs(MI_AM_SCAN_DESC *sd)**

- `mi_smallint *mi_scan_projs(MI_AM_SCAN_DESC *sd)`
- `MI_UPDATE_STAT_MODE`
`mi_tab_update_stat_mode(MI_AM_TABLE_DESC *td)`
- DataBlade API functions `mi_current_command_name()` , `mi_file_errno()` ,
and `mi_get_id()`
- The `superstores_demo` database
- The `ifxrltree.1.00` DataBlade module

Version 9.20 Features from Dynamic Server 7.30: Dynamic Server 9.20 also has features first released in Dynamic Server 7.30:

- ALTER FRAGMENT ATTACH/DETACH enhancements
- In-place ALTER TABLE MODIFY/DROP (for built-in data types)
- External backup and restore and restartable restore
- Performance enhancements, including new optimizer directives, select first *n* rows, SET OPTIMIZATION statement enhancements, memory-resident tables, correlated subquery enhancements, and key-first index scan
- Features for a database server on Windows
 - Multiple residency
 - Raw device support
 - High-Performance Loader (HPL)
 - ON-Bar XBSA certification
 - ON-Bar parallelism
 - Nondomain Administrator install
 - Microsoft cluster support
 - Local-connection support with named pipes
 - Three gigabytes of shared memory
- Application migration
 - UPPER, LOWER, and INITCAP functions for case-insensitive search (for built-in data types)
 - REPLACE, SUBSTR, LPAD, and RPAD functions for string manipulation (for built-in data types)
 - UNION operator in CREATE VIEW statement
 - CASE expression
 - NVL and DECODE functions
 - TO_CHAR and TO_DATE date-conversion functions (for built-in data types)
 - `IFX_UPDDESC` environment variable to describe an UPDATE statement
 - EXECUTE PROCEDURE syntax to update triggering columns

- New arguments to the **dbinfo()** function to obtain the hostname and version of the database server
- ISM to manage the storage devices and media that contain backups
- Additional information for **onsnmp** Management Information Bases
- Connectivity features
 - Greater network security available through support of third-party security services
 - Client interfaces (ESQL/C, CLI, C++, Java, JDBC, GLS) supported in a single package
 - Compatible with Data Director 3.6 and higher
- Enterprise Replication features
 - Hierarchical routing; direct connections no longer required
 - Additional support and performance enhancements for database servers with intermittent connections
 - Reduced global catalog available
 - Enhanced Global Language Support (GLS) to replicate multiple locales within a single replication environment
 - Command-line utility
 - Scripting view in the Replication Manager graphical interface
 - CONNECT and DISCONNECT functions in the Enterprise Replication application programming interface
- Optical Subsystem shared-library support
- Additional options for the **oncheck** utility
 - The **oncheck -w** option to check and print an index without placing a shared lock on the table
 - The **oncheck -R** option to check the reserved pages, physical-log pages, and logical-log pages

Storage-Manager Validation and Installation

When you convert or revert an IBM Informix database server, the Storage Manager that you used on the old version might not be validated for the version to which you are migrating. Verify that the storage manager is validated for the target database server version and platform at <http://www.ibm.com/software/data/informix/pubs/smv/smv.html>

If the storage manager is not validated, you need to install a certified storage manager before you perform backups with ON-Bar.

Before you convert to a later version of the database server, save a copy of your current **sm_versions** file, which should be in the **\$INFORMIXDIR/etc** directory on UNIX or Linux or the **%INFORMIXDIR%\etc** directory on Windows. If you are using a different directory as **INFORMIXDIR** for the

new database server, copy **sm_versions** to the new **\$INFORMIXDIR/etc** or **%INFORMIXDIR%\etc** directory, or copy **sm_versions.std** to **sm_versions** in the new directory, and then edit the **sm_versions** file with appropriate values before starting the conversion. If you are migrating from Version 7.3x before 7.31.xC5 or from Version 7.24, create the **sm_versions** file by unloading the information from the **sysutils:bar_version** table.

When you convert to the new database server version, install the storage manager *before* you bring up the database server. That way if you have automatic log backup set up on the database server, ON-Bar can start backing up the logs when the database server comes online.

Warning: If you migrate ISM Version 1.0 catalogs to Version 2.0 using the **ism_catalog** utility, the catalogs become corrupted. When the ISM 2.0 server is restarted after catalog migration, error messages occur in various logs.

Dynamic Server 9.40 uses ISM 2.2. When setting up ISM, you might need setup information about the following features:

- ISMData or ISMLogs name change
- NSRADMIN utility
- Year 2000 compliant status
- ISM installation and certification during migration

Important: If you are using NetWare *IPX/SPX*, it should be installed on the same computer as the ISM server.

For information on how to install and upgrade the storage manager, see the *IBM Informix: Storage Manager Administrator's Guide*.

Migrating Between 32-bit and 64-Bit Database Servers

To migrate to or from a 64-bit version of Dynamic Server 10.0 and an earlier database server (64-bit as well as 32-bit), follow the migration process described in Chapter 5, "Converting to Dynamic Server 10.0," on page 5-1 and Chapter 6, "Reverting from Dynamic Server 10.0," on page 6-1.

If you are migrating from a 32-bit version of Dynamic Server 10.0 to a 64-bit version of Dynamic Server 10.0 or reverting from 64-bit version of Dynamic Server 10.0 to a 9.40, 9.30, or 9.2x 32-bit version, you might need to follow additional steps to update certain internal tables. These steps are documented in the platform-specific machine notes that are provided with your database server.

All UDRs and DataBlade modules that were built in 32-bit mode need to be recompiled in 64-bit mode because they will not work with the 64-bit database server. If you have any UDRs that were developed in a 32-bit mode, make sure that proper size and alignment of the data structures are used in order to work correctly on a 64-bit computer after recompilation in 64-bit mode. For more information, refer to the machine notes.

Chapter 4. Migrating to Dynamic Server 10.0 with Enterprise Replication

Converting to Dynamic Server 10.0 with Enterprise Replication	4-1
Converting Replication of 9.2x User-Defined Data Types.	4-2
Reverting from Dynamic Server 10.0 with Enterprise Replication	4-3

In This Chapter

This chapter describes how to convert to and revert from Dynamic Server 10.0 if you are running Enterprise Replication.

All of the conversion and reversion operations must be performed as user **informix**.

Converting to Dynamic Server 10.0 with Enterprise Replication

You can only use the following procedures to convert to Dynamic Server 10.0 from Dynamic Server 9.40, 9.30, 9.2x, or 7.31.

To convert from an earlier database server, you need to convert to an intermediate database server first. For information on which intermediate database server to use, see “Database Server Migration Paths” on page 1-7. For information on how to convert to the intermediate database server, see the Migration Guide that is included in the IBM Informix documentation for that database server.

To prepare for conversion to Dynamic Server 10.0 with Enterprise Replication:

1. If you have replicate groups, which Dynamic Server 10.0 does not support, remove all of them.
2. Stop applications that are performing replicable transactions.
3. Make sure that control and TRG send queues are empty:
 - Run **onstat -g grp** to ensure that grouper does not have any pending transactions.
 - Run **onstat -g rqm** to check for queued messages.
4. Replace old event class identifiers (IDs) used in the ALARMPROGRAM with the new event class IDs.
5. Shut down Enterprise Replication with the following command:
`cdr stop`

To convert to Dynamic Server 10.0 with Enterprise Replication:

1. Perform the conversion tasks described in “In This Chapter” on page 5-1, including starting Dynamic Server 10.0.
2. If the `CDR_QDATA_SBSPACE` configuration parameter is not set, set it in the `ONCONFIG` file to the sbspaces for Enterprise Replication to use for storing spooled row data.
3. Back up the `syscdr` databases with the `dbschema` or the `UNLOAD` statement.
4. Make sure that no replicatable transactions occur before Enterprise Replication starts.
5. Run the conversion script, named `concdr.sh`, in the `$INFORMIXDIR/etc/conv` directory on UNIX, or `concdr.bat`, in the `%INFORMIXDIR%\etc\conv` directory on Windows:

```
% sh concdr.sh from_version 10.0
```

Valid `from_version` values are: 9.40, 9.30, 9.21, 9.20, and 7.31

6. Wait for one of the following messages:

```
'syscdr' conversion completed successfully.  
'syscdr' conversion failed.
```

For details about the conversion, see the following file:

`$INFORMIXDIR/etc/concdr.out` or `%INFORMIXDIR%\etc\concdr.out`

7. If conversion fails, resolve the problem reported in the `concdr.out` file, restore the `syscdr` database from a backup, and then attempt conversion again.
8. After successful conversion, start Enterprise Replication:

```
% cdr start
```

Warning: After you convert to Dynamic Server 10.0 with Enterprise Replication from Dynamic Server 9.30, 9.2x, or 7.31, do not drop the `syscdr` database. If `syscdr` is dropped, then you cannot revert from Dynamic Server 10.0 to the older database server with Enterprise Replication because the data required to carry out the reversion is stored in the `syscdr` database.

Converting Replication of 9.2x User-Defined Data Types

Dynamic Server 9.2x has limited support for the replication of user-defined data types (UDTs). To take advantage of 10.0 UDT replication, the user-defined routines (UDRs) for a UDT must contain `streamwrite()` and `streamread()` functions. After you migrate to Dynamic Server 10.0, implement the `streamwrite()` and `streamread()` functions for any currently replicated UDTs on all database servers within the enterprise.

Reverting from Dynamic Server 10.0 with Enterprise Replication

You can only use the following procedures to revert from Dynamic Server 10.0 to Dynamic Server 9.40, 9.30, 9.2x, or 7.31.

When you revert from Version 10.0 to an earlier version of Dynamic Server with Enterprise Replication:

- Master replicates become standard replicates and tables that were added to the **syscdr** database are removed.
- Tables created with templates are dropped.
- The table containing replicated table-version information, which was created during conversion to Version 10.0, is dropped.

The procedure for reverting to Dynamic Server 9.40 or 9.30 is slightly different than the procedure for reverting to Dynamic Server 9.2x or 7.31. To revert to Dynamic Server 7.31, the database server must be an Enterprise Replication root server (the uppermost level in a hierarchically organized set of database servers).

To revert to Dynamic Server 9.40 or 9.30 from Dynamic Server 10.0 with Enterprise Replication:

1. Stop applications doing replicatable transactions.
2. You cannot revert if Enterprise Replication is in Alter mode, so make sure Enterprise Replication is not in Alter Mode. Use **onstat -g cat repls** to see if Enterprise Replication is in Alter mode. If it is in Alter mode, change the mode.
3. Delete shadow replicates.
4. Make sure that control and TRG send queues are empty:
 - Run **onstat -g grp** to ensure that grouper does not have any pending transactions.
 - Run **onstat -g rqm** to check for queued messages.
5. Shut down Enterprise Replication with the following command:

```
cdr stop
```

6. Back up the **syscdr** databases with **dbschema** or UNLOAD.
7. Run the reversion script, named **revcdr.sh**, in the **\$INFORMIXDIR/etc/conv** directory on UNIX, or **revcdr.bat**, in the **%INFORMIXDIR%\etc\conv** directory on Windows:

```
% sh revcdr.sh 10.0 9.30
```

This script does a reversion test followed by the actual Enterprise Replication reversion.

8. If the reversion test or actual reversion fails, then check the file **\$INFORMIXDIR/etc/revtestcdr.out** or **revcdr.out**, respectively. Attempt reversion after resolving problems reported.
9. Perform database server reversion tasks, as described in “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
10. Run **onmode -l** and **onmode -c**. If you do not do this after reversion and before starting Enterprise Replication, the database server might fail when you start Enterprise Replication.
11. Start Enterprise Replication:


```
% cdr start
```

To revert to Dynamic Server 9.2x or 7.31 from Dynamic Server 10.0 with Enterprise Replication:

1. Drop all replicate sets.
2. Drop any replicate that contains a smart large object or user-defined data type.
3. Stop applications doing replicatable transactions.
4. Make sure that control and TRG send queues are empty:
 - Run **onstat -g grp** to ensure that grouper does not have any pending transactions.
 - Run **onstat -g rqm** to check for queued messages.
5. Shut down Enterprise Replication with the following command:


```
cdr stop
```
6. Back up the **syscdr** databases with **dbschema** or UNLOAD.
7. Run the reversion script, named **revcdr.sh**, in the **\$INFORMIXDIR/etc/conv** directory on UNIX, or **revcdr.bat**, in the **%INFORMIXDIR%\etc\conv** directory on Windows:

```
% sh revcdr.sh 10.0 to_version
```

Valid *to_version* values are:

```
9.21
9.20
7.31
```

This script does a reversion test followed by the actual Enterprise Replication reversion.

8. If the reversion test or actual reversion fails, then check the file **\$INFORMIXDIR/etc/revtestcdr.out** or **revcdr.out**, respectively. Attempt reversion after resolving problems reported.
9. Drop the Enterprise Replication sbspaces.

10. Perform database server reversion tasks, as described in “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
11. Run **onmode -l** and **onmode -c**. If you do not do this after reversion and before starting Enterprise Replication, the database server might fail when you start Enterprise Replication.
12. Start Enterprise Replication:
% cdr start

Chapter 5. Converting to Dynamic Server 10.0

Converting to Version 10.0	5-1
Summary of Conversion Steps	5-2
Check and Configure Available Space	5-3
Save Copies of the Current Configuration Files	5-5
Close All Transactions and Shut Down the Source Database Server	5-6
Check for Any Open Transactions	5-6
Verify the Integrity of the Data	5-7
Verify the Database Server Mode	5-8
Disable High-Availability Data Replication	5-8
Make a Final Backup of the Source Database Server	5-9
Verify That the Source Database Server Is Offline	5-9
On UNIX or Linux, Modify Kernel Parameters	5-9
Install Dynamic Server 10.0	5-9
Set Environment Variables	5-10
Customize Configuration Files	5-11
Add Any Communications Support Modules (UNIX/Linux)	5-12
Install and Configure Any DataBlade Modules.	5-12
Initialize Dynamic Server 10.0	5-12
Monitor the Conversion Complete Status	5-13
Upgrade the High-Performance Loader onpload Database	5-14
For ON-Bar, Rename the sm_versions.std File	5-15
Update Statistics	5-15
Verify the Integrity of the Data	5-15
Make an Initial Backup of Dynamic Server 10.0	5-16
Tune Dynamic Server 10.0 for Performance	5-16
Enable HDR	5-16

In This Chapter

This chapter describes the procedures for converting to Dynamic Server 10.0 from any of the following database servers:

- Dynamic Server 9.40
- Dynamic Server 9.30
- Dynamic Server 9.2x
- Dynamic Server 7.31, 7.30, or 7.24

Converting to Version 10.0

When you migrate to Dynamic Server 10.0, you can install and test a database server instance with the same configuration files, environment variables, and **sqlhosts** information that you used for your source database server.

After successful conversion to Dynamic Server 10.0, you might want to modify configuration files and environment variables to take advantage of Dynamic Server 10.0 features. For more information, see the *IBM Informix: Dynamic Server Getting Started Guide* and your *IBM Informix: Dynamic Server Administrator's Guide*.

Important: Read the release notes and the machine notes for any late-breaking information.

Summary of Conversion Steps

To convert to Dynamic Server 10.0, complete the following conversion steps. The subsections that appear below these summary steps provide more information about each step.

To convert to Dynamic Server 10.0:

1. If necessary, perform Enterprise Replication preconversion tasks. For more information, see Chapter 4, "Migrating to Dynamic Server 10.0 with Enterprise Replication," on page 4-1.
2. Check and configure available space.
3. Save copies of the current configuration files.
4. Close all transactions and shut down the source database server.
5. Check for any open transactions.
6. Verify the integrity of the data.
7. Verify the database server mode.
8. If necessary, disable High-Availability Data Replication.
9. Make a final backup of the source database server.
10. Run **ontape -a** after the backup is complete.
11. Verify that the source database server is offline.

UNIX/Linux Only

12. On UNIX or Linux, modify kernel parameters.

End of UNIX/Linux Only

13. Install Dynamic Server 10.0, according to instructions in your *IBM Informix: Installation Guide*. Be sure not to install the new database server over the old database server.
14. Monitor the message log, **online.log**, during the conversion for any error messages.
15. Set environment variables.

Note: Before you start the 10.0 database server, you must set the environment variable `DBONPLOAD` to the name of the pload database if the name is not `onpload`, the default name.

16. Customize configuration files.
17. Add any Communications Support Modules.
18. Install and configure any DataBlade modules.
19. Switch to user **informix** and initialize Dynamic Server 10.0 to trigger the conversion.
20. Monitor the conversion complete status.
21. Upgrade the High-Performance Loader **onpload** database for the current version of Dynamic Server if necessary, for example if **onpload** failed during server conversion.
22. For ON-Bar, rename or edit the **sm_versions.std** file.
23. Update statistics.
24. Verify the integrity of the data.
25. Make an initial backup of Dynamic Server 10.0.
26. Tune Dynamic Server 10.0 for performance.
27. If necessary, perform Enterprise Replication conversion tasks. For more information, see “Converting to Dynamic Server 10.0 with Enterprise Replication” on page 4-1.
28. If necessary, enable HDR.

Repeat this procedure for each instance of Dynamic Server 10.0 that you plan to run on the computer.

Warning: If a serious error occurs during the conversion, it might be necessary to return to the previous version, restore from a backup, and then correct the problem prior to restarting the conversion tasks.

Check and Configure Available Space

During conversion, Dynamic Server drops and then recreates the **sysmaster** database. Depending on which version of Dynamic Server you convert from, the Version 10.0 **sysmaster** database can be significantly larger.

UNIX/Linux Only

Dynamic Server 10.0 requires 3000 free pages of logical-log space (approximately 6000 kilobytes for a 2-kilobyte page size) to build the **sysmaster** database on UNIX or Linux.

Windows Only

Dynamic Server 10.0 requires 1500 to 3000 free pages of logical-log space (approximately 6000 kilobytes for a 4-kilobyte page size) to build the **sysmaster** database on Windows.

End of Windows Only

Partition header pages should not be full; key descriptors will occupy slightly more space after conversion to Dynamic Server 10.0. Use the **oncheck -me** command to compress extents and reduce the amount of storage used in partition headers.

The root chunk should contain at least ten percent (10%) free space when converting to Dynamic Server 10.0.

In some cases, even if the database server conversion is successful, internal conversion of some databases might fail because of insufficient space for system catalog tables. For more information, see the release notes for this version of Dynamic Server.

You need to add any additional free space to the system prior to the conversion. If the dbspaces are nearly full, you need to add space before you start the conversion procedure. When you initialize Dynamic Server 10.0 on the same root dbspace of the earlier database server, Dynamic Server 10.0 automatically converts the **sysmaster** database and then each database individually. For a successful conversion of each database, ensure that 2000 kilobytes of free space per database is available in each dbspace.

To ensure enough free space is available:

1. Calculate the amount of free space that each dbspace requires.
In the following equation, n is the number of databases in the dbspace and X is the amount of free space they require:
$$X \text{ kilobytes free space} = 2000 \text{ kilobytes} * n$$
2. Check the amount of free space in each dbspace to determine whether you need to add more space.

Use the following SQL statements to determine the free space that each dbspace requires and the free space available. These statements return the free-space calculation in page-size units. The **free_space_req** column value is the free-space requirement, and the **free_space_avail** column value is the free space available.

The following SQL statement shows how to determine the free space that each dbspace requires:

```

DATABASE sysmaster;
SELECT partdbsnum(partnum) dbspace_num,
       trunc(count(*) * 2000) free_space_req
FROM sysdatabases
GROUP BY 1
ORDER BY 1;

```

The following SQL statement queries the **syschunks** table and displays the free space available for each dbspace:

```

SELECT dbsnum dbspace_num, sum(nfree) free_space_avail
FROM syschunks
GROUP BY 1
ORDER BY 1;

```

Important: If less free space is available than the dbspace requires, either move a table from the dbspace to another dbspace or add a chunk to the dbspace.

The dbspace estimates could be higher if you have an unusually large number of SPL routines or indexes in the database.

Save Copies of the Current Configuration Files

Save copies of the configuration files that exist for each instance of your source database server. Configuration files that you might have are listed in Table 5-1. Keep the copies available in case you need to use them.

While you can use an old ONCONFIG configuration file with Version 10.0, it is recommended that you use the new Version 10.0 ONCONFIG file, or at least examine the file for new parameters.

Table 5-1. Configuration Files to Save from the Source Database Server

UNIX or Linux	Windows
\$INFORMIXDIR/etc/\$ONCONFIG	%INFORMIXDIR%\etc\%ONCONFIG%
\$INFORMIXDIR/etc/onconfig.std	%INFORMIXDIR%\etc\onconfig.std
\$INFORMIXDIR/etc/oncfg*	%INFORMIXDIR%\etc\oncfg*
\$INFORMIXDIR/etc/sm_versions	%INFORMIXDIR%\etc\sm_versions
\$INFORMIXDIR/aaodir/adtcfg	%INFORMIXDIR%\aaodir\adtcfg.*
\$INFORMIXDIR/dbssodir/adtmasks	%INFORMIXDIR%\dbssodir\adtmasks.*
\$INFORMIXDIR/etc/sqlhosts	
\$INFORMIXDIR/etc/tctermcap	
\$INFORMIXDIR/etc/termcap	

If you use ON-Bar to back up your source database server and the logical logs, you also need to save a copy of any important storage manager files as well as the following file:

UNIX or Linux:

`$INFORMIXDIR/etc/ixbar.servernum`

Windows:

`%INFORMIXDIR%\etc\ixbar.servernum`

Close All Transactions and Shut Down the Source Database Server

Communicate to client users how long you expect the database server to be offline for the migration. Terminate all database server processes and shut down your source database server. This lets users exit and shuts down the database server gracefully. If necessary, you can perform an immediate shutdown of the database server, which does not let users save their work.

Before you convert from the original source database server, make sure that no open transactions exist. Otherwise, fast recovery will fail when rolling back open transactions during the conversion.

To let users exit and shut down the database server gracefully:

1. Execute the **onmode -sy** command to put the database server in quiescent mode.
2. Wait for all users to exit.
3. Execute the **onmode -l** command to move to the next logical log.
4. Execute the **onmode -c** to force a checkpoint.
5. Make a level-0 backup of the database server.
6. Run **ontape -a** after the level-0 backup is complete.
7. Execute the **onmode -yuk** command to shut down the system.

To perform an immediate shutdown of the database server:

```
onmode -l  
onmode -c  
onmode -ky
```

Check for Any Open Transactions

A shutdown procedure does not guarantee a rollback of all open transactions. To guarantee that the source database server has no open transactions prior to the conversion, you need to put the source database server in quiescent mode. Execute the following command to enter quiescent mode and initiate a fast recovery:

oninit -s

UNIX/Linux Only

On UNIX or Linux, the **oninit -s** command rolls forward all committed transactions and rolls back all incomplete transactions since the last checkpoint and then leaves a new checkpoint record in the log with no open transactions pending. You need to execute **oninit -s** before you initialize Dynamic Server 10.0. If any transactions remain when you try to initialize the new database server, you will receive the following error when you try to initialize the new database server, and it goes offline:

```
Open transaction detected when changing log versions.
```

For more information about fast recovery, see your *IBM Informix: Dynamic Server Administrator's Guide*.

End of UNIX/Linux Only

After you put the database server in quiescent mode and initiate fast recovery, issue the **onmode -yuk** command again to shut down the database server. Then review **online.log** for any possible problems and fix them.

Only after proper shutdown can you bring the new database server (Dynamic Server 10.0) through the conversion path. Any open transaction during the conversion would cause an execution failure in fast recovery.

Verify the Integrity of the Data

Use the **oncheck** utility to verify the integrity of the data before you make a level-0 backup. If you find any problems with the data, fix them before you make the backup. You can verify the integrity of the reserve pages, extents, system catalog tables, data, and indexes.

To obtain the database names, use the following statements with DB-Access:

```
DATABASE sysmaster;  
SELECT name FROM sysdatabases;
```

Table 5-2 lists the commands that verify the data integrity.

Table 5-2. Commands for Verifying the Data Integrity

Action	oncheck Command
Check reserve pages	oncheck -cr
Check extents	oncheck -ce
Check system catalog tables	oncheck -cc database_name
Check data	oncheck -cD database_name
Check indexes	oncheck -cI database_name

For information on **oncheck**, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Verify the Database Server Mode

Before you make a backup, execute the following command to verify that your source database server is in quiescent mode:

```
onstat -
```

The first line of the onstat output contains the status of your source database server.

```
IBM Informix Dynamic Server Version X.XX.XXX -- Quiescent -- Up  
XX:XX:XX-- XXXX  
Kbytes
```

The word Quiescent indicates that Dynamic Server is in quiescent mode.

Disable High-Availability Data Replication

If you use High-Availability Data Replication (HDR), you must disable it before conversion to Dynamic Server 10.0.

To disable HDR, set the primary database server to standard with the following command on the primary database server:

```
onmode -d standard
```

For more information on HDR, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Tip: You do not need to perform the whole conversion process on the secondary database server. For more information, see "Enable HDR" on page 5-16.

Make a Final Backup of the Source Database Server

Use ON-Bar or **ontape** to make a level-0 backup of the source database server, including all storage spaces and all used logs. After you make a level-0 backup, also perform a complete backup of the logical log, including the current logical-log file. Be sure to retain and properly label the tape volume that contains the backup. For more information about making backups, see the *IBM Informix: Backup and Restore Guide*.

Important: Make a final backup for each source database server instance that you plan to convert.

For ON-Bar, remove the **ixbar** file, if any, from the **\$INFORMIXDIR%/etc** or **%INFORMIXDIR%\etc** directory after the final backup. This ensures that backups for the original source database server are not confused with backups about to be done for the new database server. Follow the instructions regarding expiration in your storage-manager documentation.

Verify That the Source Database Server Is Offline

The source database server must be offline because the new database server uses the same files. You cannot install the new database server if any of the files that it uses are active.

Check the message log to verify that you obtained this message
shared memory not initialized...

You can also use the **onstat** utility to determine that shared memory was not initialized

On UNIX or Linux, Modify Kernel Parameters

You might need to change some of the kernel parameters for your UNIX or Linux operating system before you install Dynamic Server 10.0. To reconfigure the operating system, follow the directions in both the:

- Machine Notes file included on your database server distribution media
- Kernel-configuration instructions for your operating system

Install Dynamic Server 10.0

On UNIX or Linux, you must be logged in as user **root** to install Dynamic Server 10.0.

On Windows, you must be a member of the **Informix-Admin** group. Set the **INFORMIXDIR** environment variable to the directory where you plan to install the database server.

Warning: Do not install the new database server over the old database server.

If you install the new database server in the same directory where the source database server resided, the installation script overwrites the older files. If you want to preserve your source database server files, you must install the new database server in a different directory. If you install the new database server in a different directory, you need to change the value of the **INFORMIXDIR** environment variable, or the older version of the database server will start up when you reboot.

Before you overwrite the source database server, you must take the following precautions:

- If you do not have the original media for the source database server, back up the **INFORMIXDIR** directory before you install Dynamic Server 10.0.
- Copy the configuration and **sqlhosts** files from the **etc** directory of **INFORMIXDIR** to another location in the file system.

To install and configure Dynamic Server 10.0, follow the directions in your *IBM Informix: Installation Guide* and in your *IBM Informix: Dynamic Server Administrator's Guide*.

The installation script installs Dynamic Server 10.0 into the **INFORMIXDIR** directory specified for user **root** on UNIX or Linux or for the **Informix-Admin** group on Windows.

Important: Monitor the database server message log, **online.log**, during the conversion for any error messages. If you see an error message, resolve the error condition before you continue the conversion procedure.

When you finish installation as user **root** and are ready to initialize the server, you must switch to user **informix**.

Set Environment Variables

After you install Dynamic Server 10.0, verify that the **INFORMIXDIR** environment variable and the following environment variables are set to the correct values:

INFORMIXSERVER	PATH
ONCONFIG	INFORMIXSQLHOSTS (if used)

Important: On UNIX or Linux, the client application looks for the **sqlhosts** file in the **etc** directory in the **INFORMIXDIR** directory. However,

you can use the **INFORMIXSQLHOSTS** environment variable to change the location or name of the **sqlhosts** file.

If the name of the **pload** database is not **onpload**, the default name, you must set the environment variable **DBONPLOAD** to the name of the **pload** database.

Customize Configuration Files

You can customize your ONCONFIG configuration file and environment variables to take advantage of the new features that Dynamic Server 10.0 introduced. When you initialize Dynamic Server 10.0, use the same configuration that the source database server used. After you observe the performance of Dynamic Server 10.0, you might want to adjust the configuration.

While you can use an old ONCONFIG configuration file with Version 10.0, it is recommended that you use the new Version 10.0 ONCONFIG file, or at least examine the file for new parameters.

Set the **ALARMPROGRAM** configuration parameter to either nothing or **no_log.sh** to prevent the generation of errors if the logical log fills during the conversion. For more details, see “Initialize Dynamic Server 10.0” on page 5-12. After the conversion, change the value of **ALARMPROGRAM** to **log_full.sh**.

If you use an optical storage manager, set the **OPTICAL_LIB_PATH** configuration parameter in the ONCONFIG file to the location of the optical storage manager library. For more information, see the *IBM Informix: Optical Subsystem Guide*.

If you use custom-code files with the High-Performance Loader, set the **HPL_DYNAMIC_LIB_PATH** configuration parameter in the **plconfig** file to the location of the shared library. For Dynamic Server 10.0, the default location of this library is **\$INFORMIXDIR/lib/ipldd09a.so**. For more information, see the *IBM Informix: High-Performance Loader User's Guide*.

For information on how to configure Dynamic Server 10.0, see your *IBM Informix: Dynamic Server Administrator's Guide*. For information about environment variables, see the *IBM Informix: Guide to SQL Reference*. For information about how to tune the configuration parameters, see the *IBM Informix: Dynamic Server Performance Guide*.

Important: To facilitate conversion (and reversion), use the same values for your new database server for **ROOTOFFSET**, **ROOTSIZE**, and **ROOTPATH** that you used for the source database server. Also,

keep the same size for physical logs and logical logs, including the same number of logical logs, and the same **sqlhosts** file.

Add Any Communications Support Modules (UNIX/Linux)

You can use a Communications Support Module (CSM) with Dynamic Server 10.0. After you install the CSM components, create entries in the **concsn.cfg** file and in the options field of the **sqlhosts** file to configure the CSM. For information on how to set up the CSM, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Existing client applications do not need to be recompiled or relinked if your database server does not use CSMs. If your database server uses a CSM, client applications must relink with new Informix libraries. The client applications must also have a CSM installed and configured.

Install and Configure Any DataBlade Modules

After you install Dynamic Server 10.0, and before you initialize the database server, install and register any DataBlade modules supplied by IBM or third-party vendors that you want to add to the database server. *Registration* is the process that makes the DataBlade module code available to use in a particular database. For more information on how to use DataBlade modules, see the DataBlade documentation.

Initialize Dynamic Server 10.0

If you just installed Dynamic Server as user **root**, you must switch to user **informix** before initializing the server.

Initialize Dynamic Server 10.0 without disk initialization.

Warning: Dynamic Server 10.0 writes to the logical logs with the transactions that result from creating the **sysmaster** database. If you run out of log space before the creation of the **sysmaster** database is complete, Dynamic Server 10.0 halts and indicates that you must back up the logical logs. After you back up the logical logs, the database server can finish building the **sysmaster** database. You cannot use ON-Bar to back up the logical logs because the database has not been converted yet. If you have **ALARMPROGRAM** set to **log_full.sh** in the **ONCONFIG** configuration file, errors are generated as each log file fills during the conversion. Set the value of **ALARMPROGRAM** to either nothing or **no_log.sh** so that these errors are not generated. If your logical log does fill up during the conversion, you need to back it up with **ontape**, the only backup tool you can use at this point. Issue the **ontape -a** command.

Bring Dynamic Server 10.0 online for the first time by executing **oninit** on UNIX or by using the **Service** control application on Windows. For instructions, see the *IBM Informix: Dynamic Server Administrator's Guide*.

As Dynamic Server 10.0 comes online for the first time, it modifies certain disk structures. This operation should extend the initialization process by only a minute or two. In the unlikely event that your disks cannot accommodate the growth in disk structures, you will find a message in the message-log file that instructs you to run **oncheck** on a table. The **oncheck** utility will tell you that you need to rebuild an index. You should rebuild the index as instructed.

Monitor the Conversion Complete Status

Check your message log (**online.log**) for status messages that pertain to bringing Dynamic Server 10.0 online. The conversion process ends when the following message is written to **online.log**:

```
Conversion completed successfully
```

This message indicates that the conversion process completed successfully, but it does not guarantee that each individual database was converted successfully. The message log could contain additional information regarding the success or failure of each individual database conversion. If a particular database conversion fails, then you should try to connect to the database to find out the exact cause of the failure.

At the end of the conversion of each individual database, the conversion process runs a script to update some system catalog table entries. The message log includes messages related to this script. The success or failure of the script does not prevent the usage of a database. If the script fails for a database, however, for better performance run the following script as user **informix** while connected to the database for converting to Dynamic Server 10.0 from Dynamic Server 7.3x or 7.24:

```
$INFORMIXDIR/etc/dummyupds7x.sql
```

If you encounter a failure during the conversion, you should revert to the source database server and then restore it from a backup.

The conversion process involves two phases: internal conversion and external conversion. Internal conversion is the process of converting each individual database in the system. External conversion is the process of converting the utilities, such as ON-Bar.

For information about any messages in the message log, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Important: If the message file notes problems, solve the problems before you continue to the next step.

Upgrade the High-Performance Loader onpload Database

If onpload conversion failed during server conversion, you can manually convert the **onpload** database.

Starting with Version 9.40.xC3, Dynamic 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

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

To upgrade the onpload database:

1. If you are upgrading from a version of Dynamic Server that is prior to Version 9.40, run the **conploadlegacy.sh** script, as shown in this example:

```
conploadlegacy.sh 7.31 9.40
```
2. If are upgrading from a version of Dynamic Server that is prior to Version 9.40xC3 and, if necessary, have already run the **conploadlegacy.sh** script, you must also perform one of the following tasks:
 - Run the **conpload.sh** script, as shown in this example:

```
conpload.sh 9.40 10.0
```
 - Set the new environment variable **IFX_ONPLOAD_AUTO_UPGRADE** to 1 for the upgrade to happen automatically 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.

If you invoke an HPL utility before upgrading the **onpload** database, then you receive an error stating that the **onpload** database must be converted.

Note: Starting with Dynamic Server Version 9.40.xC3, the **ipload** utility does not operate properly if you are using object names that contain more than 18 characters. The utility will continue to operate properly if legacy applications do not use long object names.

For ON-Bar, Rename the `sm_versions.std` File

After installation of the database server, rename the `sm_versions.std` file to `sm_versions` for the ON-Bar backup and restore system to run. Use one of the following methods:

- If you are using the same version of ISM, copy the same `sm_versions` file from your source database server to the new database server installation.
- If you are using other storage managers, copy your previous `sm_versions` file from the source `$INFORMIXDIR/etc` directory to the new `$INFORMIXDIR/etc` directory.
- If you are upgrading from Version 7.3x before 7.31.xC5 or from Version 7.24, unload the contents of the `sysutils:bar_version` table.

Update Statistics

After a successful conversion, you need to run `UPDATE STATISTICS` on some of the system catalog tables in your databases, as the following lists show.

For a conversion to Dynamic Server 10.0 from a Version 7.3x or Version 7.24 database server, run `UPDATE STATISTICS` on the following system catalog tables in Dynamic Server 10.0:

<code>sysblobs</code>	<code>sysfragments</code>	<code>syssynonyms</code>
<code>syscolauth</code>	<code>sysindices</code>	<code>sysstable</code>
<code>syscolumns</code>	<code>sysobjstate</code>	<code>systabauth</code>
<code>sysconstraints</code>	<code>sysopclstr</code>	<code>systables</code>
<code>sysdefaults</code>	<code>sysprocauth</code>	<code>systriggers</code>
<code>sysdistrib</code>	<code>sysprocedures</code>	<code>sysusers</code>
<code>sysfragauth</code>	<code>sysroleauth</code>	

Verify the Integrity of the Data

After Dynamic Server finishes converting the system catalog tables, open each database with DB–Access and use **oncheck** to verify that data was not corrupted during the migration process. You can verify the integrity of the reserve pages, extents, system catalog tables, data, indexes, and smart large objects, as Table 5-3 shows.

Table 5-3. Commands for Verifying the Data Integrity

Action	oncheck Command
Check reserve pages	<code>oncheck -cr</code>
Check extents	<code>oncheck -ce</code>
Check system catalog tables	<code>oncheck -cc database_name</code>
Check data	<code>oncheck -cD database_name</code>
Check indexes	<code>oncheck -cI database_name</code>
Check smart large objects	<code>oncheck -cs sbspace_name</code>
Check smart large objects plus extents	<code>oncheck -cS sbspace_name</code>

Make an Initial Backup of Dynamic Server 10.0

Use a Dynamic Server 10.0 backup and restore tool (ON-Bar or **ontape**) to make a level-0 backup of the new database server. Do not overwrite the tapes that contain the final backup of the source database server. For more information, see the *IBM Informix: Backup and Restore Guide*.

Important: Do not restore the backed up logical-log files from your source database server for your new database server.

Tune Dynamic Server 10.0 for Performance

When you finish the level-0 backup, the migration process is complete and users can safely use Dynamic Server 10.0 to access data.

After successful migration to Dynamic Server 10.0, you can tune the database server to obtain maximum performance. If you created sample queries for comparison, you can use them to characterize the performance differences between the source database server and Dynamic Server 10.0. The results of these comparisons might suggest adjustments to configuration parameters or to the layout of databases, tables, and chunks. For details on performance topics, see the *IBM Informix: Dynamic Server Performance Guide*.

Enable HDR

You do not need to perform the conversion process on the secondary server. On the secondary server, install the new database server and any user-defined objects or DataBlade modules, and then copy the necessary supporting files from the primary server: for example, the ONCONFIG file. For a list of requirements, see the section on configuring a system for HDR in the *IBM Informix: Dynamic Server Administrator's Guide*.

When you start HDR, follow the procedure for starting HDR for the first time as described in the *IBM Informix: Dynamic Server Administrator's Guide*.

Chapter 6. Reverting from Dynamic Server 10.0

Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24	6-1
Determine Whether Reversion Is Possible.	6-2
Recompile Java UDRs That Have Been Compiled Using JDK 1.4.x	6-7
Revert an onpload Database to the Previous Version	6-8
Check and Configure Available Space	6-8
Save Copies of the Current Configuration Files (UNIX/Linux).	6-8
Verify the Integrity of the Data	6-8
Back Up Dynamic Server 10.0.	6-8
Remove BladeManager Extensions	6-8
Remove Version 10.0 Features.	6-9
Disable HDR	6-9
Run the Reversion Utility	6-9
Modify Configuration Parameters	6-10
Reset Environment Variables.	6-10
Remove Any Communications Support Module Settings (UNIX/Linux)	6-10
Reinstall and Start the Target Database Server	6-10
Update Statistics.	6-10
Verify the Integrity of the Data	6-11
Back Up the Target Database Server	6-11
Return the Target Database Server to Online Mode	6-11
Enable HDR	6-11

In This Chapter

This chapter describes the steps for reverting from Dynamic Server 10.0 to Dynamic Server 9.40, 9.30, 9.2x, 7.3x, or 7.24.

Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24

Important: Before you revert from Dynamic Server 10.0:

- Remove all objects from the databases that the old database server does not support, except those that the boot scripts created in the system catalog. Do not remove the objects that the **boot90.sql** and **boot901.sql** scripts created because the reversion utility uses them.
- Read the release notes and the machine notes for any late-breaking information.

If you are reverting to Dynamic Server 7.3, you must drop any index whose key size is greater than 254. Dynamic Server 10.0 supports a maximum index

key size of 255 for VARCHAR and NVARCHAR types. However, Dynamic Server 7.3 only supports a maximum index key size of 254.

To revert from Dynamic Server 10.0, complete the following steps. The subsections that appear below these summary steps provide more information about each step.

To revert from Dynamic Server 10.0:

1. Review the database schema to determine whether reversion is possible.
2. Check and configure available space.
3. Save copies of the current configuration files.
4. Verify the integrity of the data.
5. Back up Dynamic Server Version 10.0.
6. Remove new features and any pending in-place alters which were performed in using Dynamic Server Version 10.0.
7. Remove BladeManager extensions.
8. If necessary, disable HDR.
9. If necessary, perform Enterprise Replication reversion tasks (see “Reverting from Dynamic Server 10.0 with Enterprise Replication” on page 4-3).
10. Run the reversion utility (**onmode -b**).
11. Modify configuration parameters.
12. Reset environment variables.
13. If necessary, remove any Communications Support Module settings.
14. Reinstall and start the target database server.
15. Update statistics.
16. Verify the integrity of the data.
17. Back up the target database server.
18. Return the target database server to online mode.

Determine Whether Reversion Is Possible

The easiest reversion scenario is to a Dynamic Server 10.0 database that does not contain any new features. Run the reversion utility and modify the values of the configuration parameters.

You can revert from Dynamic Server 10.0 to a 9.40, 9.30, 9.2x, 7.3x, or 7.24 database server if you have not added any extensions to the 10.0 database server.

Warning: When you revert to a previous version of the database server, do not reinitialize the database server by using the **-i** command-line

parameter. Using the **-i** parameter for reversion would reinitialize the root dbspace, which would destroy your databases.

When you revert to the older version, you see a message similar to the following:

```
This will make all necessary modifications to disk structures
so that the Informix Dynamic Server space will be
compatible with Informix Dynamic Server Version 7.31
Beginning process of reverting system to 7.31 ...
...
Reversion complete.
Install Informix Dynamic Server Version 7.31 before reinitializing.
Do you wish to continue (y/n)? qaonmode -b 7.31: passed
```

In the next-to-last line of the message, *reinitializing* refers to restarting the database server (sometimes referred to as reinitializing shared memory), *not* to reinitializing the existing root dbspace.

Review the database schema to determine whether reversion to the earlier database server is possible. Ask the following questions:

- Does the schema file contain SQL statements that the earlier database server does not support?
- Does the database contain features that the earlier database server does not support, such as long identifiers?
- Have any new SPL routines been created in Dynamic Server 10.0, or have you imported existing routines using **dbimport**?
- Have tables or indexes using expression fragmentation had expressions changed or new fragments added?
- Have any new triggers or procedures been created?
- Have any new check constraints been created?

To review the database schema, execute the **dbschema** utility command. The following example displays complete information about the database **db1**:

```
dbschema -d db1 -ss
```

Database reversion occurs in two phases: a check phase and an actual reversion phase. If a database cannot be reverted, the check phase highlights it and prevents reversion.

Table 6-1 lists the restrictions for to a Version 9.40, 9.30, 9.2x, or 7.x (7.31, 7.30, or 7.24) database server.

Table 6-1. Reversion Restrictions

7.x	9.2x	9.30	9.40	Restriction
x	x	x	x	Column-level encryption: If your tables contain encrypted data, you should not revert to a version of the server that does not contain encryption support (any version prior to Version 10.0) because you will not be able to interpret column data without writing a custom DataBlade module that is equivalent to the facilities provided by Dynamic Server. Dynamic Server does not record whether encrypted data is stored in a database.
x	x	x	x	DataBlade User-Defined Routines (UDRs) that Have the EXTERNAL Clause: The database server administrator (DBSA), user informix by default, must revoke the "extend" role from all users to whom the role has been granted.
x	x	x	x	Built-in UDRs: Many system catalog tables use built-in UDRs. If you changed the definition of a built-in UDR, you must drop the UDR before reverting.
x	x	x	x	Multiple INOUT Parameter Support: You must drop any new UDRs that were created using INOUT parameters before reverting.
x	x	x	x	Default Roles: During reversion, the defrole column is dropped from the sysusers table. You must revoke default roles from users before reverting.
x	x	x	x	Tables and Indexes that Use Fragment Partition Syntax: If you created tables using the new fragment partition syntax, you must drop the tables or you must use the ALTER FRAGMENT INIT statement to change the syntax before reverting to a pre-10.0 version of Dynamic Server. Dynamic Server drops the partname column from the sysfragments table during reversion.
x	x	x	x	External Optimizer Directives: You cannot revert if external optimizer directives have been created.
x	x	x	x	New Configuration Parameters: These cannot be reverted.
x	x	x	x	Non-default Page Size: If you specified the page size for a standard or temporary dbspace, instead of using the default dbspace page size, you must drop all non-default-size dbspaces before you revert to a pre-10.0 version of Dynamic Server.
x	x	x	x	IPv6 Addresses: If you used an IPv6 address in the SQLHOSTS file during reversion, you must replace the IPv6 address with either the machine name or IPv4 address assigned to the machine.
x	x	x	x	UDRs and Applications That Use TRUNCATE Syntax or the am_truncate() Method: You cannot revert these to a pre-10.0 version of Dynamic Server.
x	x	x	x	Procedures and Triggers Created with Version 9.40 or 10.0: You must drop all triggers and procedures created with the new version before reverting.

Table 6-1. Reversion Restrictions (continued)

7.x	9.2x	9.30	9.40	Restriction
x	x	x	x	JAVA UDRs That Have Been Compiled Using JDK 1.4.x. These must be recompiled with older JDK versions. For details, see “Recompile Java UDRs That Have Been Compiled Using JDK 1.4.x” on page 6-7.
x	x	x	x	<p>No in-place ALTER TABLE statement performed in Version 10.0 should be outstanding against any table.</p> <p>If a user table has an incomplete new in-place ALTER operation, then you need to ensure that the in-place ALTER operation is complete by running a dummy UPDATE statement against the table. If the reversion process does not complete successfully because of in-place ALTER operations, it lists all the tables that need dummy updates. You need to perform a dummy update on each of the tables in the list before you can revert to the older database server.</p> <p>If an in-place ALTER operation is incomplete against a system table, run the following script while connected to the database for reversion to a 7.3x or 7.24 database server from Dynamic Server 10.0:</p> <p>\$INFORMIXDIR/etc/dummyupds7x.sql</p> <p>Note: Any in-place alter performed in a Dynamic Server Version prior to Version 10.0 will successfully revert and dummy updates are not necessary for them. However, any new in-place alter that you performed in Version 10.0 must be completed before reversion.</p>
x	x	x		Dbspaces cannot have chunks larger than 2 GB or in the new chunk format, chunks that extend further than 2 GB into their device or file, or contain more than 2047 chunks. For more information, see “Chunk Format” on page 3-20.
x	x	x		No files larger than 2 GB should be stored in or in use by the database server.
x	x	x		The TAPESIZE or LTAPESIZE configuration parameters cannot be set to 0.
x	x	x		The ALARMPROGRAM configuration parameter cannot be set to the alarmprogram.sh file.
x	x	x		The LRU_MAX_DIRTY or LRU_MIN_DIRTY configuration parameters must not be set to a value of type FLOAT. To revert, set them to integers. (These configuration parameters have been removed in Version 10.0.)
x	x	x		UDRs should not use multiple IN or OUT parameters. Drop all such UDRs before reversion.
x	x	x		UDRs and stored procedures should not use named return parameters.
x	x	x		Sequence objects should not be in use. Drop all sequences before reversion.
x	x	x		No triggers created with the INSTEAD OF clause should be in use. Drop all such triggers before reversion.

Table 6-1. Reversion Restrictions (continued)

7.x	9.2x	9.30	9.40	Restriction
x	x	x		Multiple collations among indexes, stored procedures, triggers, and constraints should not be in use.
x	x	x		Functional indexes cannot contain more than 16 parameters.
x	x	x		High-Data Availability Replication and Enterprise Replication cannot co-exist on the same database server.
x	x	x		The <code>LVARCHAR(n)</code> data types should not be in use if n is not equal to 2042.
x	x	x		<p>You cannot revert to an earlier database server from a database server that has had extensions added unless you remove the extensions.</p> <p>You need to remove any new data types or routines that you created either explicitly or by registering a different version of a DataBlade module.</p> <p>To be able to revert, you need to downgrade any DataBlade module to the version that was registered prior to reversion and explicitly drop any data types and routines that were created outside of any DataBlade registration. For information on how to use DataBlade modules, see the DataBlade documentation.</p>
x	x	x		No new user-defined or SPL routines should have been created in the converted databases (either implicitly or explicitly). If you plan to use dbexport to export a database containing existing user-defined or SPL routines, you must drop these routines prior to reversion.
x	x	x		No new triggers should have been defined in the converted databases.
x	x	x		<p>No fragment expressions or check constraints created on the 9.40 or 10.0 database server should exist in the databases. To revert, convert fragmented tables to nonfragmented tables by detaching fragment expressions.</p> <p>You cannot use <code>ALTER TABLE</code> or <code>ALTER INDEX</code> statements to change fragment strategies that existed before the conversion to Dynamic Server 9.40.</p>
x				Indexes created with Dynamic Server 10.0 and an opclass that supports nearest-neighbor search cannot be reverted to the earlier database server.
x				Reversion fails if, for an index, the value of item_nvarchar is 255 or higher.
x				If Dynamic Server uses a newly added log file, you cannot reset the status of the file to “newly added” after reversion to the earlier database server.
x				A DataBlade module that uses the <code>PER_STMT_EXEC</code> or <code>PER_STMT_PREP</code> memory duration cannot be used with the earlier database server.
x				<p>You cannot revert a database that was created with the database server from which you are reverting.</p> <p>Drop the database before you attempt reversion.</p>

Table 6-1. Reversion Restrictions (continued)

7.x	9.2x	9.30	9.40	Restriction
x				Select triggers should not be in use.
x				User-defined statistics should not be in use.
x				No long identifiers or long usernames should be in use. Before reversion, make sure that the R-tree indexes do not use long identifiers as indexed column names, opclass names, or opclass function names. Also, make sure that the following disk structures do not use long identifiers: <ul style="list-style-type: none"> • Database tablespaces (owner and database name length) • Tablespace tablespaces (owner and tablespace name length) • Dbspaces (owner and dbspace name length) and chunks (path length)
x				No storage space should have a name more than 18 characters long.
x				No new routine languages should be defined in the converted databases.
x				No new language authorizations must have been done in the converted databases.
x				No new operator classes, casts, or extended types should be defined in the 10.0 database server.
x				No semidetached indexes should be in the databases.
x				The databases cannot have tables whose primary access method is a user-defined access method.
x				Databases cannot have typed tables.
x				Tables cannot have any user-defined type columns.
x				Tables cannot have named row types with default values.
x				All indexes must be B-tree indexes with a total key length less than or equal to 255.
x				Tables cannot have any functional or VII indexes.
x				Databases cannot use any extensibility features, including user-defined access methods, user-defined types, aggregates, routine languages, language authorizations, trace messages, trace message classes, operator classes, errors, type and casts.

Recompile Java UDRs That Have Been Compiled Using JDK 1.4.x

When reverting to an older Dynamic Server versions, JAVA UDRs that have been compiled using JDK 1.4.x must be recompiled with older JDK versions. What you do depends on whether your application uses external JAR/class files or JAR files installed on the server:

- If your application uses external JAR/class files (for example, JAR/class files that are listed in JVPCLASSPATH), just recompile the JAR/class.
- If your application uses JAR files installed in the server (for example, via the `install_jar()` support function), then you must remove the old JAR file (using `remove_jar()` support function) and re-install the re-compiled JAR file in the database.

Revert an onpload Database to the Previous Version

To revert an **onpload** database from Version 10.0 to the previous database version, first revert to Version 9.40xC3. Then, if necessary revert to another version, as follows:

1. Revert to Version 9.40 by running the **revpload.sh** script, as shown in this example:

```
revpload.sh 9.40 10.0
```

2. If you are reverting to a version of Dynamic Server that is prior to Version 9.40 and, if necessary, have already run the **revpload.sh** script, run the **revploadlegacy.sh** script, as shown in this example:

```
revploadlegacy.sh 7.31 9.40
```

Check and Configure Available Space

For Dynamic Server 10.0 and 9.40, chunk reserve pages can be allocated in non-root chunks. If the 10.0 root chunk is full and chunk reserve pages were allocated in non-root chunks, make sure you have enough space in the root chunk of the target database server. During reversion, all chunk reserve pages are written to the root chunk of the 9.30, 9.2x, 7.3x, or 7.24 database server.

Save Copies of the Current Configuration Files (UNIX/Linux)

Save copies of the ONCONFIG and **concsm.cfg** files for when you convert to Dynamic Server 10.0 again. Dynamic Server 10.0 uses the **concsm.cfg** file to configure CSMs.

Verify the Integrity of the Data

Execute the following commands to verify the integrity of the data:

```
oncheck -cI database_name
oncheck -cD database_name
oncheck -cr
oncheck -cc database_name
```

Back Up Dynamic Server 10.0

Before you begin the reversion, make a complete level-0 backup of Dynamic Server 10.0. For more information, see the *IBM Informix: Backup and Restore Guide*.

Remove BladeManager Extensions

When you run BladeManager against a database, you automatically create extensions because BladeManager registers its utility DataBlade module, which adds extensions to the database. If you need to revert from a 10.0

database server, and you have run BladeManager against a database, you must first run BladeManager and specify the following command to remove the BladeManager extensions:

```
unprep database name
```

Remove Version 10.0 Features

Before you revert, remove all features that your older database server does not support. For a list of features that you need to remove before reversion, see “Determine Whether Reversion Is Possible” on page 6-2.

Disable HDR

If you use High-Availability Data Replication (HDR), you must disable it before conversion from Dynamic Server 10.0.

To disable HDR, set the primary database server to standard with the following command on the primary database server:

```
onmode -d standard
```

For more information on HDR, see the *IBM Informix: Dynamic Server Administrator's Guide*.

Tip: You do not need to perform the whole reversion process on the secondary database server. For more information, see “Enable HDR” on page 6-11.

Run the Reversion Utility

Dynamic Server 10.0 must be running when you execute the reversion utility. The reversion utility detects and lists any remaining features that are specific to Dynamic Server 10.0. You must remove these features before reversion can complete.

Execute the reversion utility with one of the following commands:

- `onmode -b 9.4`
- `onmode -b 9.3`
- `onmode -b 9.2`
- `onmode -b 7.3`
- `onmode -b 7.2`

After the reversion is complete, Dynamic Server 10.0 is offline. The reversion utility drops the Dynamic Server 10.0 system catalog tables and restores compatibility so that users can access the data with the earlier database server. The reversion utility does not revert changes made to the layout of the data that do not affect compatibility.

For more information about the **onmode -b** command, see Chapter 12, “The onmode Utility,” on page 12-1.

Modify Configuration Parameters

Replace the Dynamic Server 10.0 ONCONFIG configuration file with the ONCONFIG file that you used before you converted. Alternatively, you can remove configuration parameters that the earlier database server does not support. You might also need to adjust the values of existing configuration parameters.

For a list of new configuration parameters by server version, see “Configuration Parameters” on page 3-6.

Reset Environment Variables

Reset the environment variables to values that are appropriate for the earlier database server.

Remove Any Communications Support Module Settings (UNIX/Linux)

If your Dynamic Server 10.0 instance used CSMs, remove any **csm** option settings that are not supported in the older database server from the **sqlhosts** file entries for the database server. Otherwise, the older database server will return an invalid **sqlhosts** options error. Delete the **concsm.cfg** file if the older database server does not support CSMs.

Reinstall and Start the Target Database Server

Reinstall and configure the 9.40, 9.30, 9.2x, 7.3x, or 7.24 database server according to the instructions in your *IBM Informix: Installation Guide* and your *IBM Informix: Dynamic Server Administrator's Guide*.

Execute the **oninit -s** command to put the 9.30, 9.2x, 7.3x, or 7.24 database server in quiescent mode.

Update Statistics

After a successful reversion, you need to run UPDATE STATISTICS on some of the system catalog tables in your databases when the database server starts.

For reversion to a 7.3x or 7.24 database server from Dynamic Server 10.0 run UPDATE STATISTICS on the following system catalog tables in the 7.3x or 7.24 database server:

SYSBLOBS	SYSFRAGMENTS	SYSSYNONYMS
SYSCOLAUTH	SYSINDEXES	SYSSYNTABLE
SYSCOLUMNS	SYSOBJSTATE	SYSTABAUTH
SYSCONSTRAINTS	SYSOPCLSTR	SYSTABLES
SYSDEFAULTS	SYSROCAUTH	SYSTRIGGERS
SYSDISTRIB	SYS PROCEDURES	SYSUSERS
SYSFRAGAETH	SYSROLEAUTH	

Verify the Integrity of the Data

Before you allow users to access the databases, check the integrity of the data. Follow the steps under “Verify the Integrity of the Data” on page 5-7.

Back Up the Target Database Server

After you complete the reversion, use ON-Bar or **ontape** to make a level-0 backup of the 9.30, 9.2x, 7.3x, or 7.24 database server. For more information about making backups, see your *IBM Informix: Backup and Restore Guide*.

Important: Do not overwrite the tapes that you used to back up your source database server.

Return the Target Database Server to Online Mode

To bring the 9.30, 9.2x, 7.3x, or 7.24 database server online, execute the **onmode -m** command. Then users can access the data.

Enable HDR

You do not need to perform the reversion process on the secondary server. On the secondary server, install the target database server and any user-defined objects or DataBlade modules, and then copy the necessary supporting files from the primary server: for example, the ONCONFIG file. For a list of requirements, see the section on configuring a system for HDR in the *IBM Informix: Dynamic Server Administrator's Guide*. The secondary server is reverted automatically when you start HDR. To start HDR, follow the procedure for starting HDR for the first time in the *IBM Informix: Dynamic Server Administrator's Guide*.

Part 3. Migration of Data Between Database Servers

Chapter 7. Migrating Between Database Servers and Operating Systems

Choosing a Migration Method	7-1
Adjusting Database Tables for File-System Variations	7-2
Moving Data to a Database Server on a Different Operating System	7-2
Using the Migration Utilities	7-3
Adapting Your Programs for UNIX or Windows	7-3
Completing Migration	7-4
Moving Data Between Dynamic Server and Workgroup Edition on Different Operating Systems	7-4

In This Chapter

This chapter describes the steps for moving data between database servers. This chapter covers the following topics:

- Choosing a migration method
- Adjusting database tables for file-system variations
- Moving data to a database server on a different operating system
- Moving data between Dynamic Server and Workgroup Edition on different operating systems

Choosing a Migration Method

UNIX or Linux and Windows store data in different page sizes. When your migration involves different operating systems, you must export data and its schema information from one database server and import the exported data into the other database server.

The method that you choose for exporting and importing data depends on how much data you plan to move. All these methods deliver similar performance and enable you to modify the schema of the database. You can use the following migration methods:

- **dbexport** and **dbimport**
To move an entire database, use the **dbexport** and **dbimport** utilities.
- UNLOAD and LOAD
To move selected columns or tables, use the UNLOAD statement. Use LOAD when you do not want to change the data format.
- UNLOAD, **dbload**, and **dbschema**
To move selected columns or tables, use the UNLOAD statement. Use **dbload** to change the data format.

- **onunload** and **onload**

To unload data in page-sized chunks, use the **onunload** utility. Use the **onload** utility to move data to an identical database server on a computer of the same type.

- The High-Performance Loader (HPL)

To move selected columns or tables or an entire database, use the HPL.

Important: Do not use **onunload** or **onload** with Dynamic Server 9.30 or 9.2x. You can use the HPL for Dynamic Server 9.2x.

For information on migration paths from some older database servers to intermediary servers before migrating to Version 10.0, see “Database Server Migration Paths” on page 1-7.

Adjusting Database Tables for File-System Variations

File-system limitations vary between NFS and non-NFS file systems. You might need to break up large tables when you migrate to a new operating system.

For example, if you have a three-gigabyte table, but your operating system allows only two-gigabyte files, break up your table into separate files before you migrate. For more information, see your *IBM Informix: Administrator’s Guide*.

Important: Informix database servers support only certified versions of NFS. For information about the NFS products you can use to NFS mount a storage space for an Informix database server, be sure to check product compatibility information.

Moving Data to a Database Server on a Different Operating System

This section describes the steps for moving data between Informix database servers on UNIX or Linux and Windows.

To move data to a database server on a different operating system:

1. Save a copy of the current configuration files.
2. Use ON-Bar, ON-Archive, or **ontape** to make a final level-0 backup.
For more information, refer to your *IBM Informix: Backup and Restore Guide* or your *IBM Informix: Archive and Backup Guide*.
3. Choose one of the following sets of migration utilities to unload the databases:
 - **dbexport** and **dbimport**
 - UNLOAD, **dbschema**, and LOAD

- UNLOAD, **dbschema**, and **dbload**
4. Bring the source database server offline.
 5. Install and configure the target database server. If you are migrating to Windows, also install the administration tools.
 6. Bring the target database server online.
 7. Use **dbimport**, LOAD, or **dbload**, or external tables to load the databases into the target database server, depending on which utility you used to export the databases.
 8. Make an initial level-0 backup of the target database server.
 9. Run UPDATE STATISTICS to update the information that the target database server uses to plan efficient queries.

Using the Migration Utilities

If you intend to move an entire database on Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x, or 7.24 between different environments, the **dbexport** and **dbimport** combination is the easiest migration method.

You can also choose one of the following migration utilities:

- If you want to move selected tables or columns, instead of an entire database, use the UNLOAD and LOAD statements with the **dbschema** utility.
- If you need to manipulate the data in the specified UNLOAD file before you load it into a new table, use a combination of the UNLOAD statement and the **dbschema** and **dbload** utilities.

For information on **dbexport**, **dbimport**, UNLOAD, LOAD, **dbload**, and **dbschema**, see Part 4, “Data Migration Utilities.”

Adapting Your Programs for UNIX or Windows

Certain database server configuration parameters and environment variables are environment dependent, as follows:

- Dynamic Server 10.0, 9.40, 9.30 and 9.2x support Enterprise Replication.
- Dynamic Server 7.3x supports Enterprise Replication and uses Version 3.0 of IBM Informix Enterprise Command Center (IECC).
- Workgroup Edition 7.3x on Windows supports GLS, ON-Bar, Enterprise Replication, and the Gateway products and uses Version 3.0 of IECC.
- Workgroup Edition 7.24 on UNIX supports GLS and uses Version 1.0 of IECC.

For details, see the information on configuration parameters in your *IBM Informix: Dynamic Server Administrator's Guide* and the *IBM Informix:*

Dynamic Server Administrator's Reference and the information on environment variables in your *IBM Informix: Dynamic Server Administrator's Guide* and the *IBM Informix: Guide to SQL Reference*.

Completing Migration

The first time the target database server is brought online, the **sysmaster** and **sysutils** databases are built. Check the message log to ensure that the **sysmaster** and **sysutils** databases have been created successfully before you allow users to access the database server. After you ensure that client users can access data on the database server, the migration process is complete.

Next you might want to seek ways to obtain maximum performance. For details on topics related to performance, see your *IBM Informix: Performance Guide*.

Moving Data Between Dynamic Server and Workgroup Edition on Different Operating Systems

The UNLOAD statement lets you retrieve selected rows from a database and write those rows to a text file. If you want to move selected tables or columns instead of an entire database between Dynamic Server and Workgroup Edition, use the UNLOAD and LOAD statements in the DB-Access utility with the **dbschema** utility.

If you need to manipulate the data in the specified UNLOAD file before you load it into a new table, use a combination of the UNLOAD statement and the **dbschema** and **dbload** utilities.

For information on UNLOAD, LOAD, **dbload**, and **dbschema**, see Part 4, "Data Migration Utilities." For information on how to use DB-Access, see the *IBM Informix: DB-Access User's Guide*.

For information about using UNLOAD, **dbschema**, and LOAD to move data between Dynamic Server 7.3x or 7.24 and Workgroup Edition or information about migrating to Workgroup Edition 7.3x from Dynamic Server 7.3x on a different operating system, see the Version 7.3x Migration Guide, Version 7.3x installation information, and Version 7.3x Release Notes.

Part 4. Data Migration Utilities

Chapter 8. The dbexport and dbimport Utilities

Syntax of the dbexport Command	8-2
Syntax of the dbimport Command	8-7
Simple Large Objects (IDS 9.x or Later)	8-14
Database Locale Changes	8-14

In This Chapter

This chapter describes the **dbexport** and **dbimport** utilities and how to use them. You can use **dbexport** and **dbimport** with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x, or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE
- OnLine 5.1x

The **dbexport** utility unloads a database into text files for later import into another database and creates a schema file. The **dbimport** utility creates and populates a database from text files. You can use the schema file with **dbimport** to re-create the database schema in another Informix environment. You can edit the schema file to modify the database that **dbimport** creates. The **dbexport** and **dbimport** utilities support Dynamic Server 10.0, 9.40, 9.30, and 9.2x extended data types.

Dates are stored in four-digit years. By default, **dbexport** exports dates in four-digit year dates unless the environment variable **DBDATE** is set to "mdy2" or to some other value that specifies abbreviated years. We do not recommend this setting for exporting a database because data imported back into the database depends on either the **DBCENTURY** environment variable, if set, or the current century if **DBCENTURY** is not set.

Important: Disable SELECT triggers before exporting a database with **dbexport**. The **dbexport** utility executes SELECT statements during export. The SELECT statement triggers can modify the database content.

Warning: When you import a database, use the same environment variable settings that were used when the database was created or you might get unexpected results. If any fragmentation expressions, check constraints, triggers, or user-defined routines were created

with different settings than you use with **dbimport**, you cannot reproduce the database accurately with a single import.

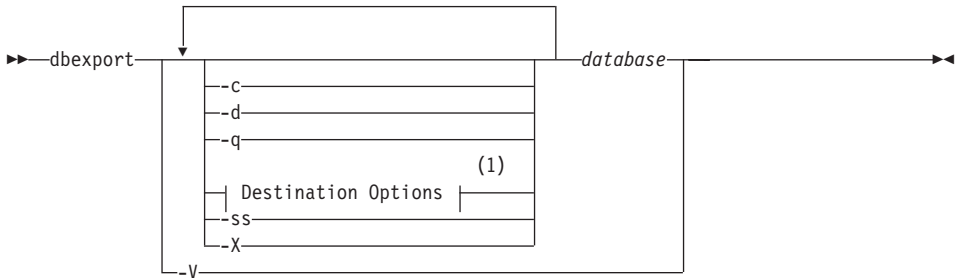
Note: If the date context during import is not the same as when these objects were created, you might get explicit errors, or you might not be able to find your data, or a check constraint might not work as expected, and so forth. Many of these problems do not generate errors. The date context for an object includes the date the object was created, the values of the **DBCENTURY** and **DBDATE** environment variables, and some other environment variables. To avoid such problems with the date context, use four-digit dates in all cases.

Global Language Support

Similar problems might occur with environment variables that specify GLS locales, such as **DB_LOCALE**, **SERVER_LOCALE**, and **CLIENT_LOCALE**. For more information, see the *IBM Informix: GLS User's Guide*.

End of Global Language Support

Syntax of the dbexport Command



Notes:

- 1 See page 8-4

Element	Purpose	Key Considerations
-c	Makes dbexport complete exporting unless a fatal error occurs	References: For details on this option, see “Errors” on page 8-4.
-d	Makes dbexport export simple-large-object descriptors only, not simple-large-object data	References: For information about simple-large-object descriptors, see the <i>IBM Informix: Optical Subsystem Guide</i> . Restrictions: Not supported by SE.
-q	Suppresses the display of error messages, warnings, and generated SQL data-definition statements	None.
-ss	Generates database server-specific information for all tables in the specified database	References: For details on this option, see “Server-Specific Information” on page 8-4.
-X	Recognizes HEX binary data in character fields	None.
-V	Displays product version information	None.
<i>database</i>	Specifies the name of the database that you want to export	Additional Information: If your locale is set to use multibyte characters, you can use multibyte characters for the database name. References: If you want to use more than the simple name of the database, refer to the Database Name section of the <i>IBM Informix: Guide to SQL Syntax</i> .

You must have DBA privileges or log in as user **informix** to export a database.

Global Language Support

When the environment variables are set correctly, as described in the *IBM Informix: GLS User’s Guide*, **dbexport** can handle foreign characters in data and export the data from GLS databases. For more information, refer to “Database Renaming” on page 8-13.

End of Global Language Support

You can use delimited identifiers with the **dbexport** utility. The utility detects database objects that are keywords, mixed case, or have special characters and places double quotes around them.

In addition to the data files and the schema file, **dbexport** creates a file of messages named **dbexport.out** in the current directory. This file contains error

messages, warnings, and a display of the SQL data definition statements that it generates. The same material is also written to the standard output unless you specify the **-q** option.

During the export, the database is locked in exclusive mode. If **dbexport** cannot obtain an exclusive lock, it displays a diagnostic message and exits.

Tip: The **dbexport** utility can create files larger than 2 GB. To support such large files, make sure your operating system file-size limits are set sufficiently high. For example, on UNIX, set **ulimit** to unlimited.

Termination of dbexport

You can press the Interrupt key at any time to cancel **dbexport**. The **dbexport** utility asks for confirmation before it terminates.

Errors

The **-c** option tells **dbexport** to complete exporting unless a fatal error occurs. Even if you use the **-c** option, **dbimport** interrupts processing if one of the following fatal errors occurs:

- Unable to open the tape device specified
- Bad writes to the tape or disk
- Invalid command parameters
- Cannot open database or no system permission

Server-Specific Information

The **-ss** option generates server-specific information. The **-ss** option specifies initial- and next-extent sizes, fragmentation information if the table is fragmented, the locking mode, the dbspace for a table, the blob space for any simple large objects, and the dbspace for any smart large objects.

Destination Options

Destination Options:

<code>--o--directory</code>
<code>--t--device--b--blocksize--s--tapesize</code>
<code>--t--pathname</code>

Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies, in kilobytes, the block size of the tape device	None.
-f <i>pathname</i>	Specifies the pathname where you want the schema file stored, if you are storing the data files on tape	Additional Information: The pathname can be a complete pathname or simply a filename. If only a filename is given, the file is stored in the current directory.
-o <i>directory</i>	Specifies the directory on disk in which dbexport creates the <i>database.exp</i> directory. This holds the data files and the schema file that dbexport creates for the <i>database</i> .	Restrictions: The directory specified as <i>directory name</i> must already exist.
-s <i>tapesize</i>	Specifies, in kilobytes, the amount of data that you can store on the tape	Additional Information: To write to the end of the tape, specify <i>tapesize</i> as 0. Restrictions: If you do not specify 0, then the maximum <i>tapesize</i> is 2,097,151 kilobytes.
-t <i>device</i>	Specifies the pathname of the tape device where you want the text files and, possibly, the schema file stored	Restrictions: The -t option does not allow you to specify a remote tape device.

When you write to disk, **dbexport** creates a subdirectory, *database.exp*, in the directory that the **-o** option specifies. The **dbexport** utility creates a file with the *.unl* extension for each table in the database. The schema file is written to the file *database.sql*. The *.unl* and *.sql* files are in the *database.exp* directory.

If you do not specify a destination for the data and schema files, the subdirectory *database.exp* is placed in the current working directory.

When you write the data files to tape, you can use the **-f** option to store the schema file to disk. You are not required to name the schema file *database.sql*. You can give it any name.

UNIX/Linux Only

For non-SE database servers on UNIX or Linux, the command is as follows:
`dbexport //finland/reports`

The following command exports the database **stores_demo** to tape with a block size of 16 kilobytes and a tape capacity of 24,000 kilobytes. The schema file is written to */tmp/stores_demo.imp*.

```
dbexport -t /dev/rmt0 -b 16 -s 24000 -f /tmp/stores_demo.imp
stores_demo
```

The following command exports the same **stores_demo** database to the directory named **/work/exports/stores_demo.exp**. The resulting schema file is **/work/exports/stores_demo.exp/stores_demo.sql**.

```
dbexport -o /work/exports stores_demo
```

End of UNIX/Linux Only

Windows Only

For Windows, the following command exports the database **stores_demo** to tape with a block size of 16 kilobytes and a tape capacity of 24,000 kilobytes. The schema file is written to **C:\temp\stores_demo.imp**.

```
dbexport -t \\.\TAPE2 -b 16 -s 24000 -f  
C:\temp\stores_demo.imp stores_demo
```

The following command exports the same **stores_demo** database to the directory named **D:\work\exports\stores_demo.exp**. The resulting schema file is **D:\work\exports\stores_demo.exp\stores_demo.sql**.

```
dbexport -o D:\work\exports stores_demo
```

End of Windows Only

Contents of the Schema File

The schema file contains the SQL statements that you need to re-create the exported database. You can edit the schema file to modify the schema of the database.

The schema file supports all Dynamic Server 10.0, 9.40, 9.30, and 9.2x data types.

If you use the **-ss** option, the schema file contains server-specific information, such as initial- and next-extent sizes, fragmentation information, lock mode, the dbspace where each table resides, the blobspace where each simple-large-object column resides, and the dbspace for smart large objects. The following information is not retained:

- Logging mode of the database
For information about logging modes, see the *IBM Informix: Guide to SQL Reference*.
- The starting values of SERIAL columns

The statements in the schema file that create tables, views, indexes, partition-fragmented tables and indexes, roles, and grant privileges do so with the name of the user who originally created the database. In this way, the

original owner retains DBA privileges for the database and is the owner of all the tables, indexes, and views. In addition, the person who executes the **dbimport** command also has DBA privileges for the database.

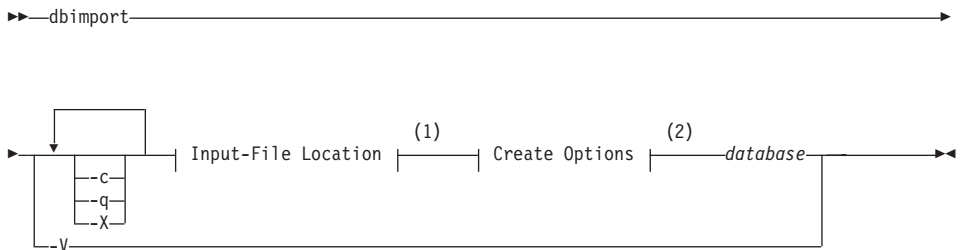
The schema file that **dbexport** creates contains comments, enclosed in braces, with information about the number of rows, columns, and indexes in tables, and information about the unload files. The **dbimport** utility uses the information in these comments to load the database.

Warning: Do not delete any comments in the schema file. It is strongly recommended that you do not change any existing comments or add any new comments, or the **dbimport** might abort or produce unpredictable results.

Note: The number of rows should match in the unload file and the corresponding unload comment in the schema file. If you change the number of rows in the unload file but not the number of rows in the schema file, a mismatch occurs.

Tip: If you delete rows from an unload file, update the comment in the schema file with the correct number of rows in the unload file. Then **dbimport** will be successful.

Syntax of the dbimport Command



Notes:

- 1 See page 8-9
- 2 See page 8-11

Element	Purpose	Key Considerations
-c	Instructs dbimport to complete importing even though it encounters certain nonfatal errors	References: For more information, see “Errors and Warnings” on page 8-9.
-q	Suppresses the display of error messages, warnings, and generated SQL data-definition statements	None.
-V	Displays product version information	None.
-X	Recognizes HEX binary data in character fields	None.
<i>database</i>	Specifies the name of the database to create	Additional Information: To use more than the simple name of the database, see the Database Names segment in the <i>IBM Informix: Guide to SQL Syntax</i> .

The **dbimport** utility can use files from the following location options:

- All input files are located on disk.
- All input files are located on tape.
- The schema file is located on disk, and the data files are on tape.

Important: Do not put comments into your input file. Comments might cause unpredictable results when the **dbimport** utility reads them.

The **dbimport** utility supports the following tasks for a new Informix database server (excluding SE):

- Create an ANSI-compliant database (includes unbuffered logging)
- Establish transaction logging for a database (unbuffered or buffered logging)
- Specify the dbspace where the database will reside

The user who runs **dbimport** is granted the DBA privilege on the newly created database. The **dbimport** process locks each table as it is being loaded and unlocks the table when the loading is complete.

Global Language Support

When the GLS environment variables are set correctly, as the *IBM Informix: GLS User's Guide* describes, **dbimport** can import data into database versions that support GLS.

End of Global Language Support

Termination of dbimport

To cancel **dbimport**, press the Interrupt key at any time. The **dbimport** program asks for confirmation before it terminates.

Errors and Warnings

If you include the **-c** option, **dbimport** ignores the following errors:

- A data row that contains too many columns
- Inability to put a lock on a table
- Inability to release a lock

Even if you use the **-c** option, **dbimport** interrupts processing if one of the following fatal errors occurs:

- Unable to open the tape device specified
- Bad writes to the tape or disk
- Invalid command parameters
- Cannot open database or no system permission
- Cannot convert the data

The **dbimport** utility creates a file of messages called **dbimport.out** in the current directory. This file contains any error messages and warnings that are related to **dbimport** processing. The same information is also written to the standard output unless you specify the **-q** option.

Input-File Location Options

The input-file location tells **dbimport** where to look for the *database.exp* directory, which contains the files that **dbimport** will import. If you do not specify an input-file location, **dbimport** looks for data files in the directory *database.exp* under the current directory and for the schema file in *database.exp/database.sql*.

Input-File Location:

```
-----|-----  
-i-directory-----  
-t-device--b-blocksize--s-tapesize--  
-t-pathname-----
```

Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies, in kilobytes, the block size of the tape device	Restrictions: If you are importing from tape, you must use the same block size that you used to export the database.
-f <i>pathname</i>	Specifies where dbimport can find the schema file to use as input to create the database when the data files are read from tape	Additional Information: If you use the -f option to export a database, you typically use the same pathname that you specified in the dbexport command. If you specify only a filename, dbimport looks for the file in the .exp subdirectory of your current directory.
-i <i>directory</i>	Specifies the complete pathname on disk of the database.exp directory, which holds the input data files and schema file that dbimport uses to create and load the new database. The directory name should be the same as the database name.	Additional Information: This directory should be the same directory that you specified with the dbexport -o option. If you change the directory name, you also rename your database.
-s <i>tapesize</i>	Specifies, in kilobytes, the amount of data that you can store on the tape	Additional Information: To read to the end of the tape, specify a tape size of 0. Restrictions: If you are importing from tape, you must use the same tape size that you used to export the database. If you do not specify 0 as the tapesize , then the maximum tapesize is 2,097,151 kilobytes.
-t <i>device</i>	Specifies the pathname of the tape device that holds the input files	Restrictions: The -t option does <i>not</i> allow you to specify a remote tape device.

UNIX/Linux Only

The following command imports the **stores_demo** database from a tape with a block size of 16 kilobytes and a capacity of 24,000 kilobytes. The schema file is read from **/tmp/stores_demo.imp**.

```
dbimport -c -t /dev/rmt0 -b 16 -s 24000 -f
/tmp/stores_demo.imp stores_demo
```

The following command imports the **stores_demo** database from the **stores_demo.exp** directory under the **/work/exports** directory. The schema file is assumed to be **/work/exports/stores_demo.exp/stores_demo.sql**.

```
dbimport -c -i /work/exports stores_demo
```

End of UNIX/Linux Only

Windows Only

The following command imports the **stores_demo** database from a tape with a block size of 16 kilobytes and a capacity of 24,000 kilobytes. The schema file is read from **C:\temp\stores_demo.imp**.

```
dbimport -c -t \\.\TAPEDRIVE -b 16 -s 24000 -f
C:\temp\stores_demo.imp stores_demo
```

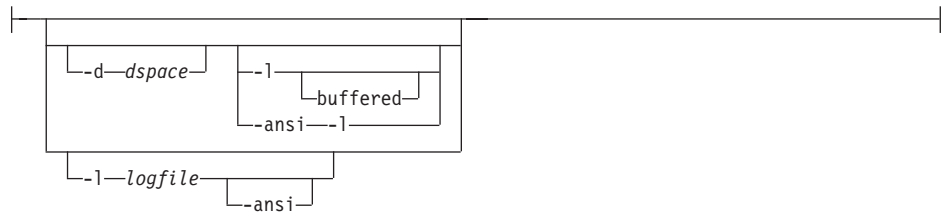
The following command imports the **stores_demo** database from the **stores_demo.exp** directory under the **D:\work\exports** directory. The schema file is assumed to be **D:\work\exports\stores_demo.exp\stores_demo.sql**.

```
dbimport -c -i D:\work\exports stores_demo
```

End of Windows Only

Create Options

Create Options:



Element	Purpose	Key Considerations
-ansi	Creates an ANSI-compliant database in which the ANSI rules for transaction logging are enabled	Additional Information: If you specify the -ansi option, you must also specify the -l logfile option. For more information about ANSI-compliant databases, see the <i>IBM Informix: Guide to SQL Reference</i> .
-d dbspace	Specifies the dbspace where the database is created. The default dbspace location is the rootdbs .	Additional Information: For SE, the database is always in the current directory.
-l	Establishes unbuffered transaction logging for the imported database	References: For more information, see “Database-Logging Mode” on page 8-13.
-l buffered	Establishes buffered transaction logging for the imported database	References: For more information, see “Database-Logging Mode” on page 8-13.
-l logfile	Establishes transaction logging for the imported database and specifies the name of the transaction-log file	Restrictions: For SE, the <i>logfile</i> filename must be an absolute pathname or in the current directory. References: For more information, see “Database-Logging Mode” on page 8-13.

If you created a table or index fragment containing partitions in Dynamic Server 10.0 or a later version, you must use syntax containing the partition name when importing a database that contains multiple partitions within a single dbspace. See the *IBM Informix: Guide to SQL Syntax* for syntax details.

UNIX/Linux Only

The following command imports the **stores_demo** database from the **/usr/informix/port/stores_demo.exp** directory. The new database is ANSI compliant, and the transaction-log file is specified as **stores_demo.log** in **/usr/work**.

```
dbimport -c stores_demo -i /usr/informix/port -l
/usr/work/stores_demo.log -ansi
```

End of UNIX/Linux Only

Windows Only

The following command imports the **stores_demo** database from the **C:\USER\informix\port\stores_demo.exp** directory. The new database is ANSI compliant, and the transaction-log file is specified as **stores_demo.log** in **C:\USER\work**.

```
dbimport -c stores_demo -i C:\USER\informix\port -l
C:\USER\work\stores_demo.log -ansi
```


Database-Logging Mode

The logging mode is not retained in the schema file. You can specify any of the following options when you use **dbimport** to import a database:

- ANSI-compliant database with unbuffered logging
- Unbuffered logging
- Buffered logging

For more information, see “Create Options” on page 8-11.

The **-l** options are equivalent to the logging clauses of the CREATE DATABASE statement, as follows:

- The **-l** option is equivalent to the WITH LOG clause.
- The **-l buffered** option is equivalent to the WITH BUFFERED LOG.

For more information about the CREATE DATABASE statement, see the *IBM Informix: Guide to SQL Syntax*.

Database Renaming

The **dbimport** utility gives the new database the same name as the database that you exported. If you export a database to tape, you cannot change its name when you import it with **dbimport**.

If you export a database to disk, you can change the database name.

In the following example, assume that **dbexport** unloaded the database **stores_demo** into the directory **/work/exports/stores_demo.exp**. Thus, the data files (the **.unl** files) are stored in **/work/exports/stores_demo.exp**, and the schema file is **/work/exports/stores_demo.exp/stores_demo.sql**.

To change the database name to new name on UNIX or Linux:

1. Change the name of the **.exp** directory. That is, change **/work/exports/stores_demo.exp** to **/work/exports/newname.exp**.
2. Change the name of the schema file. That is, change **/work/exports/stores_demo.exp/stores_demo.sql** to **/work/exports/stores_demo.exp/newname.sql**. Do not change the names of the **.unl** files.
3. Import the database with the following command:

```
dbimport -i /work/exports newname
```

To change the database name to new name on Windows:

In the following example, assume that **dbexport** unloaded the database **stores_demo** into the directory **D:\work\exports\stores_demo.exp**. Thus, the data files (the **.unl** files) are stored in **D:\work\exports\stores_demo.exp**, and the schema file is **D:\work\exports\stores_demo.exp\stores_demo.sql**.

1. Change the name of the **.exp** directory. That is, change **D:\work\exports\stores_demo.exp** to **D:\work\exports\newname.exp**.
2. Change the name of the schema file. That is, change **D:\work\exports\stores_demo.exp\stores_demo.sql** to **D:\work\exports\stores_demo.exp\newname.sql**. Do not change the names of the **.unl** files.
3. Import the database with the following command:

```
dbimport -i D:\work\exports
```

Simple Large Objects (IDS 9.x or Later)

When **dbimport**, **dbexport**, and DB–Access process simple-large-object data, they create temporary files for that data. Before you export or import data from tables that contain simple large objects, you must have one of the following items:

- A **\tmp** directory on your currently active drive
- The **DBTEMP** environment variable set to point to a directory that is available for temporary storage of the simple large objects

Windows Only

Windows sets the **TMP** and **TEMP** environment variables in the command prompt sessions, by default. However, if the **TMP**, **TEMP**, and **DBTEMP** environment variables are not set, **dbimport** places the temporary files for the simple large objects in the **\tmp** directory.

End of Windows Only

Warning: If a table has a CLOB or BLOB in a column, you cannot use **dbexport** to export the table to a tape. If a table has a user-defined type in a column, using **dbexport** to export the table to a tape might yield unpredictable results, depending on the export function of the user-defined type. Exported CLOB sizes are stored in hex format in the unload file.

Database Locale Changes

You can use **dbimport** to change the locale of a database.

To change the locale of a database:

1. Set the **DB_LOCALE** environment variable to the name of the current database locale.
2. Run **dbexport** on the database.
3. Use the DROP DATABASE statement to drop the database that has the current locale name.
4. Set the **DB_LOCALE** environment variable to the desired database locale for the database.
5. Run **dbimport** to create a new database with the desired locale and import the data into this database.

Chapter 9. The dbload Utility

Syntax of the dbload Command 9-1
 Command File for dbload 9-5
 Command File to Load Complex Data Types (IDS 9.x or Later) 9-14

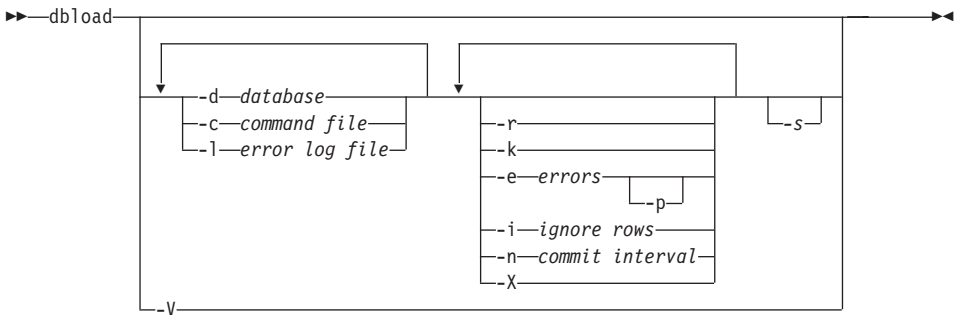
In This Chapter

This chapter describes the **dbload** utility and how to use it. You can use **dbload** with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE
- OnLine 5.1x

The **dbload** utility loads data into databases or tables that IBM Informix products created. It transfers data from one or more text files into one or more existing tables. This utility supports new data types in Dynamic Server 10.0, 9.40, 9.30 and 9.2x.

Syntax of the dbload Command



Element	Purpose	Key Considerations
-c <i>command file</i>	Specifies the filename or pathname of a dbload command file	References: For information about building the command file, see "Command File for dbload" on page 9-5.

Element	Purpose	Key Considerations
-d database	Specifies the name of the database to receive the data	Additional Information: If you want to use more than the simple name of the database, see the Database Name section of the <i>IBM Informix: Guide to SQL Syntax</i> .
-e errors	Specifies the number of bad rows that dbload reads before terminating . The default value for <i>errors</i> is 10.	References: For more information, see “Bad-Row Limit” on page 9-3.
-i ignore rows	Specifies the number of rows to ignore in the input file	References: For more information, see “Rows to Ignore” on page 9-3.
-k	Instructs dbload to lock the tables listed in the command file in exclusive mode during the load operation	References: For more information, see “Table Locking” on page 9-3. Restrictions: You cannot use the -k option with the -r option because the -r option specifies that no tables are locked during the load operation.
-l error log file	Specifies the filename or pathname of an error log file	Restrictions: If you specify an existing file, its contents are overwritten. If you specify a file that does not exist, dbload creates the file. Additional Information: The error log file stores diagnostic information and any input file rows that dbload cannot insert into the database.
-n commit interval	Specifies the commit interval in number of rows The default interval is 100 rows.	Additional Information: If your database supports transactions, dbload commits a transaction after the specified number of new rows is read and inserted. A message appears after each commit. References: For information about transactions, see the <i>IBM Informix: Guide to SQL Tutorial</i> .
-p	Prompts for instructions if the number of bad rows exceeds the limit	References: For more information, see “Bad-Row Limit” on page 9-3.
-r	Prevents dbload from locking the tables during a load, thus enabling other users to update data in the table during the load	Additional Information: For more information, see “Table Locking” on page 9-3. Restrictions: You cannot use the -r option with the -k option because the -r option specifies that the tables are not locked during the load operation while the -k option specifies that the tables are locked in exclusive mode.
-s	Checks the syntax of the statements in the command file without inserting data	Additional Information: The standard output displays the command file with any errors marked where they are found.

Element	Purpose	Key Considerations
-V	Displays product version information	None.
-X	Recognizes HEX binary data in character fields	None.

Tip: If you specify part (but not all) of the required information, **dbload** prompts you for additional specifications. The database name, command file, and error log file are all required. If you are missing all three options, you receive an error message.

Table Locking

If you do not specify the **-k** option, the tables specified in the command file are locked in shared mode. When tables are locked in shared mode, the database server still has to acquire exclusive row or page locks when it inserts rows into the table.

When you specify the **-k** option, the database server places an exclusive lock on the entire table. The **-k** option increases performance for large loads because the database server does not have to acquire exclusive locks on rows or pages as it inserts rows during the load operation.

If you do not specify the **-r** option, the tables specified in the command file are locked during loading so that other users cannot update data in the table. Table locking reduces the number of locks needed during the load but reduces concurrency. If you are planning to load a large number of rows, use table locking and load during nonpeak hours.

To override this default lock mode, specify the **-k** option. The **-k** option instructs **dbload** to lock the tables in exclusive mode rather than shared mode during the load operation.

Rows to Ignore

The **-i** option instructs **dbload** to read and ignore the specified number of new-line characters in the input file before it begins to process. This option is useful if your most recent **dbload** session ended prematurely. For example, if **dbload** ends after it inserts 240 lines of input, you can begin to load again at line 241 if you set *number rows ignore* to 240. It is also useful if header information in the input file precedes the data records.

Bad-Row Limit

The **-e** option lets you specify how many bad rows to allow before **dbload** terminates.

If you set *errors* to a positive integer, **dbload** terminates when it reads (*errors* + 1) bad rows. If you set *errors* to zero, **dbload** terminates when it reads the first bad row.

If **dbload** exceeds the bad-row limit and the **-p** option is specified, **dbload** prompts you for instructions before it terminates. The prompt asks whether you want to roll back or to commit all rows that were inserted since the last transaction.

If **dbload** exceeds the bad-row limit and the **-p** option is not specified, **dbload** commits all rows that were inserted since the last transaction.

Guidelines for Using dbload

This section includes the following guidelines for using the **dbload** utility:

- Termination of dbload
- Network names
- Simple large objects
- Indexes
- Delimited identifiers
- SE Example

Termination of dbload

If you press the Interrupt key, **dbload** terminates and discards any new rows that were inserted but not yet committed to the database (if the database has transactions).

Network Names

If you are on a network, include the database server name and directory path with the database name to specify a database on another database server or coserver.

Simple Large Objects

You can load simple large objects with the **dbload** utility as long as the simple large objects are in text files.

Indexes

The presence of indexes greatly affects the speed with which the **dbload** utility loads data. For best performance, drop any indexes on the tables that receive the data before you run **dbload**. You can create new indexes after **dbload** has finished.

Delimited Identifiers

You can use delimited identifiers with the **dbload** utility. The utility detects database objects that are keywords, mixed case, or have special characters, and places double quotes around them.

If your most recent **dbload** session ended prematurely, specify the starting line number in the command-line syntax to resume loading with the next record in the file.

Example

The following command loads data into the **stores_demo** database in the **turku** directory on the a database server called **finland**:

```
dbload -d //finland/turku/stores_demo -c commands -l errlog
```

Command File for dbload

Before you use **dbload**, you must create a command file that names the input data files and the tables that receive the data. The command file maps fields from one or more input files into columns of one or more tables within your database.

The command file contains only FILE and INSERT statements. Each FILE statement names an input data file. The FILE statement also defines the data fields from the input file that are inserted into the table. Each INSERT statement names a table to receive the data. The INSERT statement also defines how **dbload** places the data that is described in the FILE statement into the table columns.

Within the command file, the FILE statement can appear in these forms:

- Delimiter form
- Character-position form

The FILE statement has a size limit of 4,096 bytes.

Use the delimiter form of the FILE statement when every field in the input data row uses the same delimiter and every row ends with a new-line character. This format is typical of data rows with variable-length fields. You can also use the delimiter form of the FILE statement with fixed-length fields as long as the data rows meet the delimiter and new-line requirements. The delimiter form of the FILE and INSERT statements is easier to use than the character-position form.

Use the character-position form of the FILE statement when you cannot rely on delimiters and you need to identify the input data fields by character position within the input row. For example, use this form to indicate that the first input data field begins at character position 1 and continues until character position 20. You can also use this form if you must translate a character string into a null value. For example, if your input data file uses a

sequence of blanks to indicate a null value, you must use this form if you want to instruct **dbload** to substitute null at every occurrence of the blank-character string.

You can use both forms of the FILE statement in a single command file. For clarity, however, the two forms are described separately in sections that follow.

FILE and INSERT Statements: Delimiter Form

The following example of a **dbload** command file illustrates a simple delimiter form of the FILE and INSERT statements. The example is based on the **stores_demo** database. An UNLOAD statement created the three input data files, **stock.unl**, **customer.unl**, and **manufact.unl**. To see the **.unl** input data files, refer to the directory **\$INFORMIXDIR/demo/prod_name** (UNIX or Linux) or **%INFORMIXDIR%\demo\prod_name** (Windows).

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO stock;
FILE customer.unl DELIMITER '|' 10;
INSERT INTO customer;
FILE manufact.unl DELIMITER '|' 3;
INSERT INTO manufact;
```

Syntax for the Delimiter Form

The following diagram shows the syntax of the delimiter FILE statement.

►—FILE—*filename*—DELIMITER—'*c*'—*nfields*—►

Element	Purpose	Key Considerations
<i>c</i>	Specifies the character as the field delimiter for the specific input file	Restrictions: If the delimiter specified by c appears as a literal character anywhere in the input file, the character must be preceded with a backslash (\) in the input file. For example, if the value of c is specified as a square bracket ([), you must place a backslash before any literal square bracket that appears in the input file. Similarly, you must precede any backslash that appears in the input file with an additional backslash.
<i>filename</i>	Specifies the input file	None.
<i>nfields</i>	Indicates the number of fields in each data row	None.

The **dbload** utility assigns the sequential names **f01**, **f02**, **f03**, and so on to fields in the input file. You cannot see these names, but if you refer to these fields to specify a value list in an associated INSERT statement, you must use the **f01**, **f02**, **f03** format. For details, refer to “How to Write a dbload Command File in Delimiter Form” on page 9-8.

Two consecutive delimiters define a null field. As a precaution, you can place a delimiter immediately before the new-line character that marks the end of each data row. If the last field of a data row has data, you must use a delimiter. If you omit this delimiter, an error results whenever the last field of a data row is not empty.

Inserted data types correspond to the explicit or default column list. If the data field width is different from its corresponding character column width, the data is made to fit. That is, inserted values are padded with blanks if the data is not wide enough for the column or truncated if the data is too wide for the column.

If the number of columns named is fewer than the number of columns in the table, **dbload** inserts the default value that was specified when the table was created for the unnamed columns. If no default value is specified, **dbload** attempts to insert a null value. If the attempt violates a not null restriction or a unique constraint, the insert fails, and an error message is returned.

If the INSERT statement omits the column names, the default INSERT specification is every column in the named table. If the INSERT statement omits the VALUES clause, the default INSERT specification is every field of the previous FILE statement.

An error results if the number of column names listed (or implied by default) does not match the number of values listed (or implied by default).

The syntax of **dbload** INSERT statements resembles INSERT statements in SQL, except that in **dbload**, INSERT statements cannot incorporate SELECT statements.

Warning: Do not use the CURRENT, TODAY, and USER keywords of the INSERT INTO statement in a **dbload** command file; they are not supported in the **dbload** command file. These keywords are supported in SQL only.

For example, the following **dbload** command is not supported:

```
FILE "testtb12.un1" DELIMITER '|' 1;
      INSERT INTO testtb1
      (testuser, testtime, testfield)
      VALUES
      ('kae', CURRENT, f01);
```

Load the existing data first and then write an SQL query to insert or update the data with the current time, date, or user login. You could write the following SQL statement:

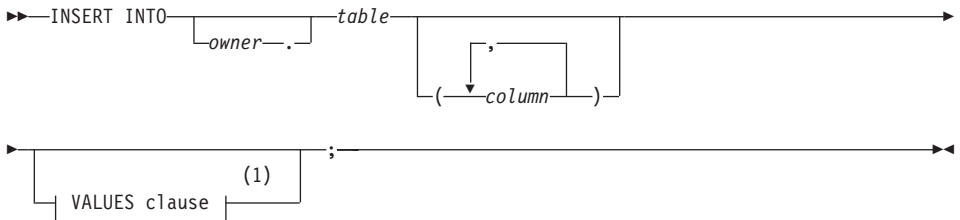
```

INSERT INTO testtbl
      (testuser, testtime, testfield)
VALUES
      ('kae', CURRENT, f01);

```

The **CURRENT** keyword returns the system date and time. The **TODAY** keyword returns the system date. The **USER** keyword returns the user login name.

The following diagram shows the syntax of the **dbload** INSERT statement for delimiter form.



Notes:

- 1 See *IBM Informix Guide to SQL: Syntax*

Element	Purpose	Key Considerations
<i>column</i>	Specifies the column that receives the new data	None.
<i>owner.</i>	Specifies the user name of the table owner	None.
<i>table</i>	Specifies the table that receives the new data	None.

Users who execute **dbload** with this command file must have the Insert privilege on the named table.

How to Write a dbload Command File in Delimiter Form

The first **FILE** and **INSERT** statement set in the delimiter example on page 9-6 is repeated in the following example:

```

FILE stock.unl DELIMITER '|' 6;
INSERT INTO stock;

```

The **FILE** statement describes the **stock.unl** data rows as composed of six fields, each separated by a vertical bar (|) as the delimiter. Two consecutive delimiters define a null field. As a precaution, you can place a delimiter immediately before the new-line character that marks the end of each data row. If the last field of a data row has data, you must use a delimiter. If you omit this delimiter, an error results.

Compare the FILE statement with the data rows in the following example, which appear in the input file **stock.unl**. (Because the last field is not followed by a delimiter, an error results if any data row ends with an empty field.)

```
1|SMT|baseball gloves|450.00|case|10 gloves/case
2|HR0|baseball|126.00|case|24/case
3|SHK|baseball bat|240.00|case|12/case
```

The example INSERT statement contains only the required elements. Because the column list is omitted, the INSERT statement implies that values are to be inserted into every field in the **stock** table. Because the VALUES clause is omitted, the INSERT statement implies that the input values for every field are defined in the most recent FILE statement. This INSERT statement is valid because the **stock** table contains six fields, which is the same number of values that the FILE statement defines. The following example shows the first data row that is inserted into **stock** from this INSERT statement.

Field	Column	Value
f01	stock_num	1
f02	manu_code	SMT
f03	description	baseball gloves
f04	unit_price	450.00
f05	unit	case
f06	unit_descr	10 gloves/case

The FILE and INSERT statement in the following example illustrates a more complex INSERT statement syntax:

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO new_stock (col1, col2, col3, col5, col6)
VALUES (f01, f03, f02, f05, 'autographed');
```

In this example, the VALUES clause uses the field names that **dbload** assigns automatically. You must reference the automatically assigned field names with the letter **f** followed by a number: **f01**, **f02**, **f10**, **f100**, **f999**, **f1000**, and so on. All other formats are incorrect.

Tip: The first nine fields must include a zero: **f01**, **f02**, ..., **f09**.

The user changed the column names, the order of the data, and the meaning of **col6** in the new **stock** table. Because the fourth column in **new_stock** (**col4**) is not named in the column list, the new data row contains a null in the **col4** position (assuming that the column permits nulls). If no default is specified for **col4**, the inserted value is null.

The following table shows the first data row that is inserted into **new_stock** from this INSERT statement.

Column	Value
col1	1
col2	baseball gloves
col3	SMT
col4	null
col5	case
col6	autographed

FILE and INSERT Statements: Character-Position Form

The examples in this section are based on an input data file, **cust_loc_data**, which contains the last four columns (**city**, **state**, **zipcode**, and **phone**) of the **customer** table. Fields in the input file are padded with blanks to create data rows in which the location of data fields and the number of characters are the same across all rows. The definitions for these fields are CHAR(15), CHAR(2), CHAR(5), and CHAR(12), respectively. Figure 9-1 displays the character positions and five example data rows from the **cust_loc_data** file.

	12	3
	1234567890123456789012345678901234	
Sunnyvale	CA94086408-789-8075	
Denver	C080219303-936-7731	
Blue Island	NY60406312-944-5691	
Brighton	MA02135617-232-4159	
Tempe	AZ85253xxx-xxx-xxxx	

Figure 9-1. A Sample Data File

The following example of a **dbload** command file illustrates the character-position form of the FILE and INSERT statements. The example includes two new tables, **cust_address** and **cust_sort**, to receive the data. For the purpose of this example, **cust_address** contains four columns, the second of which is omitted from the column list. The **cust_sort** table contains two columns.

```
FILE cust_loc_data
(city 1-15,
 state 16-17,
 area_cd 23-25 NULL = 'xxx',
 phone 23-34 NULL = 'xxx-xxx-xxxx',
 zip 18-22,
 state_area 16-17 : 23-25);
```

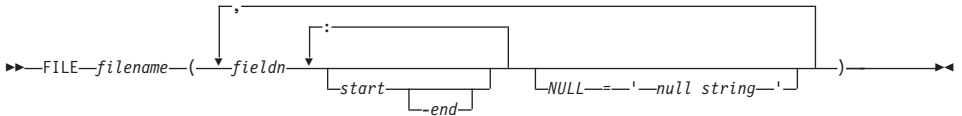
```

INSERT INTO cust_address (col1, col3, col4)
VALUES (city, state, zip);
INSERT INTO cust_sort
VALUES (area_cd, zip);

```

Syntax for the Character-Position Form

The following diagram shows the syntax of the character-position FILE statement.



Element	Purpose	Key Considerations
<i>-end</i>	Indicates the character position within a data row that ends a range of character positions	Restrictions: A hyphen must precede the <i>end</i> value.
<i>fieldn</i>	Assigns a name to the data field that you are defining with the range of character positions	None.
<i>filename</i>	Specifies the name of the input file	None.
<i>null string</i>	Specifies the data value for which dbload should substitute a null	Restrictions: Must be a quoted string.
<i>start</i>	Indicates the character position within a data row that starts a range of character positions. If you specify <i>start</i> without <i>end</i> , it represents a single character.	None.

You can repeat the same character position in a data-field definition or in different fields.

The *null string* scope of reference is the data field for which you define it. You can define an explicit null string for each field that allows null entries.

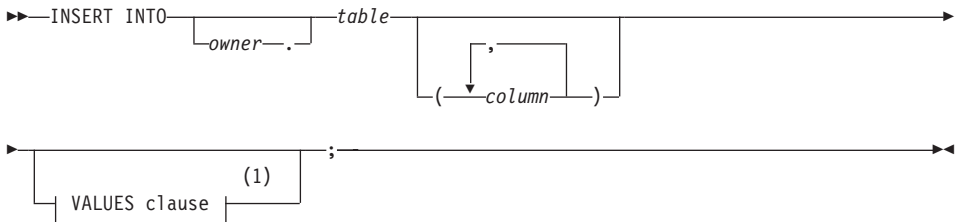
Inserted data types correspond to the explicit or default column list. If the data-field width is different from its corresponding character column, inserted values are padded with blanks if the column is wider or are truncated if the field is wider.

If the number of columns named is fewer than the number of columns in the table, **dbload** inserts the default value that is specified for the unnamed columns. If no default value is specified, **dbload** attempts to insert a null value. If the attempt violates a not-null restriction or a unique constraint, the insert fails, and an error message is returned.

If the INSERT statement omits the column names, the default INSERT specification is every column in the named table. If the INSERT statement omits the VALUES clause, the default INSERT specification is every field of the previous FILE statement.

An error results if the number of column names listed (or implied by default) does not match the number of values listed (or implied by default).

The syntax of **dbload** INSERT statements resembles INSERT statements in SQL, except that in **dbload**, INSERT statements cannot incorporate SELECT statements. The following diagram shows the syntax of the **dbload** INSERT statement for character-position form.



Notes:

- 1 See *IBM Informix: Guide to SQL Syntax*

Element	Purpose	Key Considerations
<i>column</i>	Specifies the column that receives the new data	None.
<i>owner.</i>	Specifies the user name of the table owner	None.
<i>table</i>	Specifies the table that receives the new data	None.

The syntax for character-position form is identical to the syntax for delimiter form.

The user who executes **dbload** with this command file must have the Insert privilege on the named table.

How to Write a dbload Command File in Character-Position Form

The first FILE and INSERT statement set in the character-position example on page 9-10 is repeated in the following example:

```

FILE cust_loc_data
  (city 1-15,
   state 16-17,
   area_cd 23-25 NULL = 'xxx',
   phone 23-34 NULL = 'xxx-xxx-xxxx',

```



```

zip 18-22,
state_area 16-17 : 23-25);
INSERT INTO cust_address (col1, col3, col4)
VALUES (city, state, zip);

```

The FILE statement defines six data fields from the **cust_loc_data** table data rows. The statement names the fields and uses character positions to define the length of each field. Compare the FILE statement in the preceding example with the data rows in Figure 9-2.

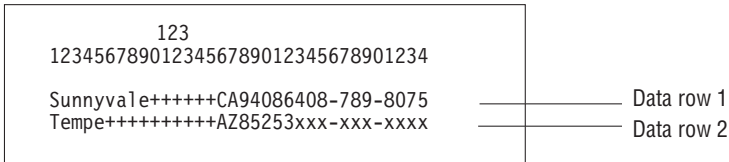


Figure 9-2. A Sample Data File

The FILE statement defines the following data fields, which are derived from the data rows in Figure 9-2.

Column	Values from Data Row 1	Values from Data Row 2
city	Sunnyvale+++++	Tempe+++++xxxx
state	CA	AZ
area_cd	408	null
phone	408-789-8075	null
zip	94086	85253
state_area	CA408	AZxxx

The null strings that are defined for the **phone** and **area_cd** fields generate the null values in those columns, but they do not affect the values that are stored in the **state_area** column.

The INSERT statement uses the field names and values that are derived from the FILE statement as the value-list input. Consider the following INSERT statement:

```

INSERT INTO cust_address (col1, col3, col4)
VALUES (city, state, zip);

```

The INSERT statement uses the data in Figure 9-2 and the FILE statement on page 9-12 to put the following information into the **cust_address** table.

Column	Values from Data Row 1	Values from Data Row 2
col1	Sunnyvale++++++	Tempe+++++++++
col2	null	null
col3	CA	AZ
col4	94086	85253

Because the second column (**col2**) in **cust_address** is not named, the new data row contains a null (assuming that the column permits nulls).

Consider the following INSERT statement:

```
INSERT INTO cust_sort
VALUES (area_cd, zip);
```

This INSERT statement inserts the following data rows into the **cust_sort** table.

Column	Values from Data Row 1	Values from Data Row 2
col1	408	null
col2	94086	85253

Because no column list is provided, **dbload** reads the names of all the columns in **cust_sort** from the system catalog. (You cannot insert data into a temporary table because temporary tables are not entered into the system catalog.) Field names from the previous FILE statement specify the values to load into each column. You do not need one FILE statement for each INSERT statement.

Command File to Load Complex Data Types (IDS 9.x or Later)

This section describes how to write **dbload** command files that load columns that contain complex data types into tables. The examples cover how to use **dbload** with named row types, unnamed row types, sets, and lists.

Using dbload with Named Row Types

The procedure for how to use **dbload** with named row types is somewhat different than for other complex data types because named row types are actually user-defined data types. In fact, you can follow these steps for any user-defined data type.

This example uses a table **person** that contains one column with a named row type. The **person_t** named row type contains six fields: **name**, **address**, **city**, **state**, **zip**, and **bdate**.

The following syntax shows how to create the named row type and the table used in this example:

```
CREATE ROW TYPE person_t
(
    name VARCHAR(30) NOT NULL,
    address VARCHAR(20),
    city VARCHAR(20),
    state CHAR(2),
    zip VARCHAR(9),
    bdate DATE
);
CREATE TABLE person OF TYPE person_t;
```

To load data for a named row type:

1. Use the UNLOAD statement to unload the table to an input file. In this example, the input file sees the named row type as six separate fields:

```
Brown, James|13 First St.|San Francisco|CA|94070|01/04/1940|
Karen Smith|5820 Easy Ave #100|Fremont|CA|94502|01/13/1983|
```

1. Use the **dbschema** utility to capture the schema of the table and the row type. You must use the **dbschema -u** option to pick up the named row type.

```
dbschema -d stores_demo -u person_t > schema.sql
dbschema -d stores_demo -t person > schema.sql
```

2. Use DB–Access to re-create the **person** table in the new database. For detailed steps, see “DB–Access Input from dbschema Output” on page 10-14.
3. Create the **dbload** command file. This **dbload** command file inserts two rows into the **person** table in the new database.

```
FILE person.unl DELIMITER '|' 6;
INSERT INTO person;
```

This **dbload** example shows how to insert new data rows into the **person** table. The number of rows in the INSERT statement and the **dbload** command file must match:

```
FILE person.unl DELIMITER '|' 6;
INSERT INTO person
VALUES ('Jones, Richard', '95 East Ave.',
        'Philadelphia', 'PA',
        '19115',
        '03/15/97');
```

4. Execute the **dbload** command:
dbload -d newdb -c uds_command -l errlog

Tip: To find the number of fields in an unloaded table that contains a named row type, count the number of fields between each vertical bar (|) delimiter.

Using dbload with Unnamed Row Types

You can use **dbload** with unnamed row types. In the following example, the **devtest** table contains two columns with unnamed row types, **s_name** and **s_address**. The **s_name** column contains three fields: **f_name**, **m_init**, and **l_name**. The **s_address** column contains four fields: **street**, **city**, **state**, and **zip**.

```
CREATE TABLE devtest
(
  s_name ROW(f_name varchar(20), m_init char(1), l_name varchar(20)
not null),
  s_address ROW(street varchar(20), city varchar(20), state char(20),
zip varchar(9)
);
```

The data from the **devtest** table is unloaded into the **devtest.unl** file. Each data row contains two delimited fields, one for each unnamed row type. The **ROW** constructor precedes each unnamed row type, as follows:

```
ROW('Jim','K','Johnson')|ROW('10 Grove St.','Eldorado','CA','94108')|
ROW('Candy','S','Cane')|ROW('7 Willy Wonka
Ave.','Hershey','PA','17033')|
```

This **dbload** example shows how to insert data that contains unnamed row types into the **devtest** table. Put double quotes around each unnamed row type or the insert will not work.

```
FILE devtest.unl DELIMITER '|' 2;
INSERT INTO devtest (s_name, s_address)
VALUES ("row('Craig','X','Smith')",
"row('1200 Cheese Ave.', 'Rainy City', 'OR', '97200')");
```

Using dbload with Collection Data Types

You can use **dbload** with collection data types such as **SET**, **LIST**, and **MULTISET**.

SET Data Type Example

In a **SET**, each element is unique, and no nulls are allowed. The number of elements in a **SET** can vary. The following statement creates a table in which the **children** column is defined as a **SET**:

```
CREATE TABLE employee
(
  name char(30),
  address char(40),
  children SET (varchar(30) NOT NULL)
);
```

The data from the **employee** table is unloaded into the **employee.unl** file. Each data row contains four delimited fields. The first set contains three elements (**Karen**, **Lauren**, and **Andrea**), whereas the second set contains four elements. The **SET** constructor precedes each **SET** data row.

```
Muriel|5555 SW Merry
Sailing Dr.|02/06/1926|SET{'Karen','Lauren','Andrea'}|
  Larry|1234 Indian Lane|07/31/1927|SET{'Martha',
    'Melissa','Craig','Larry'}|
```

This **dbload** example shows how to insert data that contains SET data types into the **employee** table in the new database. Put double quotes around each SET data type or the insert does not work.

```
FILE employee.unl DELIMITER '|' 4;
INSERT INTO employee
VALUES ('Marvin', '10734 Pardee', '06/17/27',
  "SET{'Joe', 'Ann'}");
```

LIST Data Type Example

A list is an ordered collection of elements that allows duplicate values. The following statement creates a table in which the **month_sales** column is defined as a LIST:

```
CREATE TABLE sales_person
(
  name CHAR(30),
  month_sales LIST(MONEY NOT NULL)
);
```

The data from the **sales_person** table is unloaded into the **sales.unl** file. Each data row contains two delimited fields, as follows:

```
Jane Doe|LIST{'4.00','20.45','000.99'}|
Big Earner|LIST{'0000.00','00000.00','999.99'}|
```

This **dbload** example shows how to insert data that contains LIST data types into the **sales_person** table in the new database. Put double quotes around each LIST data type or the insert does not work.

```
FILE sales_person.unl DELIMITER '|' 2;
INSERT INTO sales_person
VALUES ('Jenny Chow', "{587900, 600000}");
```

You can load multisets in a similar manner.

Using dbload with Other Data Types

You can use **dbload** with the following data types:

- A BLOB or CLOB
- A SET inside a ROW type

The **dbload** utility does not work with the following data types:

- A CLOB or BLOB inside a ROW type
- A ROW type inside a SET

Warning: All the load utilities (**dbexport**, **dbimport**, **dbload**, **onload**, **onunload**, and **onxfer**) rely on an export and import function. If you do not define this function when you write a user-defined data type, you cannot use these utilities.

Note: Loading a new data type inside another data type can cause problems if the representation of the data contains handles. If a string represents the data, you should be able to load it.

Chapter 10. The dbschema Utility

Syntax of the dbschema Command	10-2
Database Schema Creation	10-3
Creating Schemas for Databases Across a UNIX or Linux Network	10-4
Changing the Owner of an Object	10-4
Server-Specific Information	10-5
User-Defined and Complex Data Types (IDS 9.x or Later)	10-5
Sequence Creation	10-6
Synonym Creation	10-7
Privileges	10-7
Granting Privileges	10-8
Displaying Privilege Information for a Role	10-8
Table, View, or Procedure Creation.	10-9
Table Information	10-9
Role Creation	10-10
Distribution Information for Tables	10-11
Example Output	10-12
Distribution Description	10-12
Distribution Information.	10-13
Overflow Information	10-13
DB-Access Input from dbschema Output	10-14
Inserting a Table into a Database Example.	10-14
Re-Creating the Schema of a Database	10-14

In This Chapter

This chapter describes the **dbschema** utility and how to use it. You can use **dbschema** with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE
- OnLine 5.1x

The **dbschema** utility prints the SQL statements that are necessary to replicate a specified table, view, or database. It also shows the distributions that the UPDATE STATISTICS statement creates.

You can use the **dbschema** utility for the following purposes:

- To display the SQL statements (the *schema*) that are required to replicate a database or a specific table, view, synonym, sequence, or procedure
- To display the schema for the Information Schema views

- To display the distribution information that is stored for one or more tables in the database
- To display information on user-defined data types and row types

Caution: Use of the **dbschema** utility can increment sequence objects in the database, creating gaps in the generated numbers that might not be expected in applications that require serialized integers.

Syntax of the dbschema Command

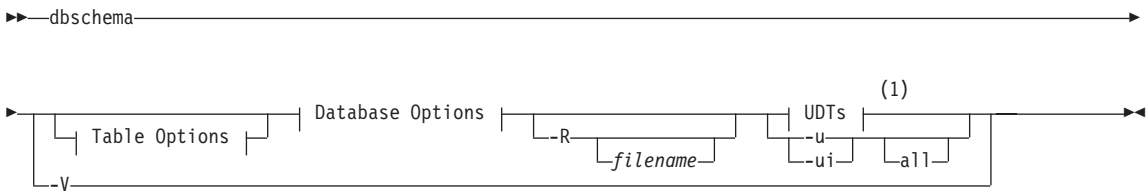
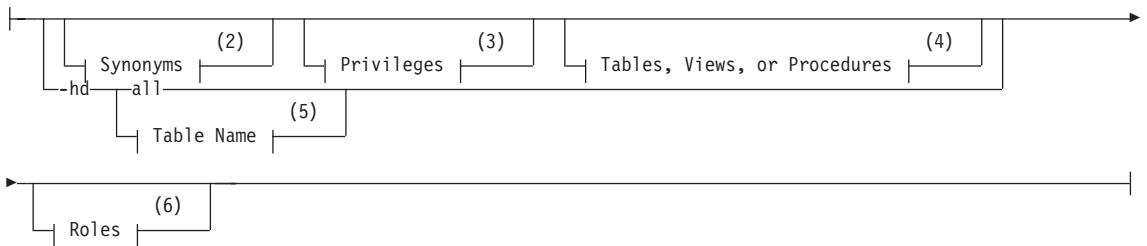
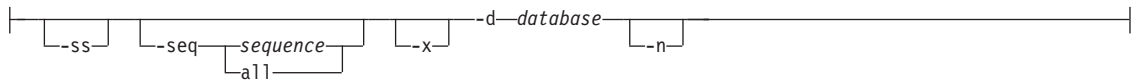


Table Options:



Database Options:



Notes:

- 1 See page 10-5
- 2 See page 10-7
- 3 See page 10-7
- 4 See page 10-9
- 5 See *IBM Informix Guide to SQL: Syntax*
- 6 See page 10-10

Element	Purpose	Additional Information
all	Directs dbschema to include all the tables or sequence objects in the database, or all the user-defined data types in the display of distributions	None.
-d database	Specifies the database or co-server to which the schema applies. The <i>database</i> can be on a remote database server.	References: If you want to qualify the name of the <i>database</i> , see the "Database Name" section of the <i>IBM Informix: Guide to SQL Syntax</i> .
filename	Specifies the filename to contain the dbschema output	If you omit <i>filename</i> , dbschema sends output to the screen. If you specify a <i>filename</i> , dbschema creates a file named <i>filename</i> to contain the dbschema output.
-hd	Displays the distribution as data values	References: For more information, see "Distribution Information for Tables" on page 10-11.
-it	Displays the isolation type.	None.
-l	Displays the lock level.	None.
-seq sequence	Generates the DDL statement to define the specified <i>sequence</i> object	References: For more information, see "Sequence Creation" on page 10-6.
-ss	Generates server-specific information	Restrictions: This option is ignored if no table schema is generated. References: For more information, see "Server-Specific Information" on page 10-5.
-u	Prints the definitions of functions, casts, and user-defined data types	References: For more information, see "User-Defined and Complex Data Types (IDS 9.x or Later)" on page 10-5.
-ui	Prints the definitions of user-defined data types, including type inheritance	References: For more information, see "User-Defined and Complex Data Types (IDS 9.x or Later)" on page 10-5.
-V	Displays product version information	None.
-x	Expands dbslice names into dbspace name lists in -ss output	If the dbspace contains multiple partitions, dbspace partition names appear in the output.

You must be the DBA or have the Connect or Resource privilege for the database before you can run **dbschema** on it.

Database Schema Creation

You can create the schema for an entire database or for a portion of the database. The options for **dbschema** allow you to perform the following actions:

- Display CREATE SYNONYM statements by owner, for a specific table or for the entire database.
- Display the CREATE TABLE, CREATE VIEW, CREATE FUNCTION, or CREATE PROCEDURE statement for a specific table or for the entire database.
- Display all GRANT privilege statements that affect a specified user or that affect all users for a database or a specific table. The user can be either a user name or role name.

Dynamic Server Versions 9.20 to 9.40

- Display user-defined and row data types with or without type inheritance.
- Display the CREATE SEQUENCE statement defining the specified *sequence* object, or defining all sequence objects in the database.

End of Dynamic Server Versions 9.20 to 9.40

When you use **dbschema** and specify only the database name, it is equivalent to using **dbschema** with all its options (except for the **-hd** and **-ss** options). In addition, if Information Schema views were created for the database, this schema is shown. For example, the following two commands are equivalent:

```
dbschema -d stores_demo
dbschema -s all -p all -t all -f all -d stores_demo
```

SERIAL fields included in CREATE TABLE statements that **dbschema** displays do not specify a starting value. New SERIAL fields created with the schema file have a starting value of 1, regardless of their starting value in the original database. If this value is not acceptable, you must modify the schema file.

Creating Schemas for Databases Across a UNIX or Linux Network

You can specify a database on any accessible non-SE Informix database server with the **-d** database syntax. The following command displays the schema for the **stores_demo** database on the **finland** database server on the UNIX or Linux system console:

```
dbschema -d //finland/stores_demo
```

Changing the Owner of an Object

The **dbschema** utility uses the *owner.object* convention when it generates any CREATE TABLE, CREATE INDEX, CREATE SYNONYM, CREATE VIEW, CREATE SEQUENCE, CREATE PROCEDURE, CREATE FUNCTION, or GRANT statement, and when it reproduces any unique, referential, or check constraint. As a result, if you use the **dbschema** output to create a new object (table, index, view, procedure, constraint, sequence, or synonym), the owner of

the original object owns the new object. If you want to change the owner of the new object, you must edit the **dbschema** output before you run it as an SQL script.

You can use the output of **dbschema** to create a new function if you also specify the *pathname* to a file in which compile-time warnings are stored. This pathname is displayed in the **dbschema** output.

For more information about the CREATE TABLE, CREATE INDEX, CREATE SYNONYM, CREATE VIEW, CREATE SEQUENCE, CREATE PROCEDURE, CREATE FUNCTION, and GRANT statements, see the *IBM Informix: Guide to SQL Syntax*.

Server-Specific Information

The **-ss** option generates server-specific information. In all Informix database servers except SE, the **-ss** option always generates the lock mode, extent sizes, and the dbspace name if the dbspace name is different from the database dbspace. In addition, if tables are fragmented, the **-ss** option displays information about the fragmentation strategy.

When you specify the **dbschema -ss** option, the output also displays any GRANT FRAGMENT statements that are issued for a particular user or in the entire schema.

The **-x** option expands dbslice names into dbspace name lists in **-ss** output.

Important: Use the **dbschema -ss** option to obtain information specific to a database server, including fragmentation and storage options.

If the dbspace contains multiple partitions, dbspace partition names appear in the output.

For information about fragment-level authority, see the GRANT FRAGMENT and REVOKE FRAGMENT statements in the *IBM Informix: Guide to SQL Syntax*.

User-Defined and Complex Data Types (IDS 9.x or Later)

When you specify the **dbschema -u** option, the output displays the definitions of any user-defined and complex data types that the database contains. The suboption **i** lets you display the type inheritance.

The following command displays all the user-defined and complex data types for the **stork** database:

```
dbschema -d stork -u all
```

Output from **dbschema** that is executed with the specified option `-u all` might appear as the following example shows:

```
create row type 'informix'.person_t
(
  name varchar(30, 10) not null,
  address varchar(20, 10),
  city varchar(20, 10),
  state char(2),
  zip integer,
  bdate date
);
create row type 'informix'.employee_t
(
  salary integer,
  manager varchar(30, 10)
) under person_t;
```

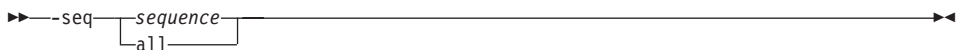
The following command displays the user-defined and complex data types, as well as their type inheritance for the **person_t** table in the **stork** database:

```
dbschema -d stork -ui person_t
```

Output from **dbschema** executed with the option `-ui person_t` might appear as the following example shows:

```
create row type 'informix'.person_t
(
  name varchar(30, 10) not null,
  address varchar(20, 10),
  city varchar(20, 10),
  state char(2),
  zip integer,
  bdate date
);
create row type 'informix'.employee_t
(
  salary integer,
  manager varchar(30, 10)
) under person_t;
create row type 'informix'.sales_rep_t
(
  rep_num integer,
  region_num integer,
  commission decimal(16),
  home_office boolean
) under employee_t;
```

Sequence Creation



Element	Purpose	Key Considerations
<code>-seq <i>sequence</i></code>	Displays the CREATE SEQUENCE statement defining <i>sequence</i>	None.
<code>-seq all</code>	Displays all CREATE SEQUENCE statements for the database	None.

Executing **dbschema** with option `-seq` sequitur might produce this output:
CREATE SEQUENCE sequitur INCREMENT 10 START 100 NOCACHE CYCLE

For more information about the CREATE SEQUENCE statement, see the *IBM Informix: Guide to SQL Syntax*.

Synonym Creation

Synonyms:

```
|--s ownername
      |
      |all
```

Element	Purpose	Key Considerations
<code>-s <i>ownername</i></code>	Displays the CREATE SYNONYM statements owned by <i>ownername</i>	None.
<code>-s all</code>	Displays all CREATE SYNONYM statements for the database, table, or view specified	None.

Output from **dbschema** that is executed with the specified option `-s alice` might appear as the following example shows:
CREATE SYNONYM 'alice'.cust FOR 'alice'.customer

For more information about the CREATE SYNONYM statement, see the *IBM Informix: Guide to SQL Syntax*.

Privileges

Privileges:

```
|--p user
      |
      |all
```

Element	Purpose	Key Considerations
-p user	Displays the GRANT statements that grant privileges to <i>user</i> , where <i>user</i> is a user name or role name. Specify only one user or role	Restriction: You cannot specify a specific list of users with the -p option. You can specify either one user or role, or all users and roles.
-p all	Displays the GRANT statements for all users for the database, table, or view specified, or to all roles for the table specified	None.

The output also displays any GRANT FRAGMENT statements that are issued for a specified user or role or (with the **all** option) for the entire schema.

Granting Privileges

In the **dbschema** output, the AS keyword indicates the grantor of a GRANT statement. The following example output indicates that **norma** issued the GRANT statement:

```
GRANT ALL ON 'tom'.customer TO 'claire' AS 'norma'
```

When the GRANT and AS keywords appear in the **dbschema** output, you might need to grant privileges before you run the **dbschema** output as an SQL script. Referring to the previous example output line, the following conditions must be true before you can run the statement as part of a script:

- User **norma** must have the Connect privilege to the database.
- User **norma** must have all privileges WITH GRANT OPTION for the table **tom.customer**.

For more information about the GRANT, GRANT FRAGMENT, and REVOKE FRAGMENT statements, see the *IBM Informix: Guide to SQL Syntax*.

Displaying Privilege Information for a Role

A *role* is a classification with privileges on database objects granted to the role. The DBA can assign the privileges of a related work task, such as an engineer, to a role and then grant that role to users, instead of granting the same set of privileges to every user. After a role is created, the DBA can use the GRANT statement to grant the role to users or to other roles.

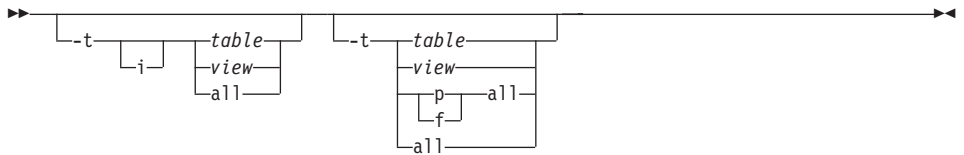
The following **dbschema** command and output show the privileges that were granted for the **calen** role:

```
sharky% dbschema -p calen -d stores_demo
```

```
DBSCHEMA Schema Utility      INFORMIX-SQL Version 7.22
Copyright (C) IBM Corporation, 1984-1996
Software Serial Number RDS#N000000
```

```
grant alter on table1 to 'calen'
```

Table, View, or Procedure Creation



Element	Purpose	Key Considerations
-f all	Limits the SQL statement output to those statements that are needed to replicate all functions and procedures	None.
-f function	Limits the SQL statement output to only those statements that are needed to replicate the specified function	None.
-f procedure	Limits the SQL statement output to only those statements that are needed to replicate the specified procedure	None.
-ff all	Limits the SQL statement output to those statements that are needed to replicate all functions	None.
-fp all	Limits the SQL statement output to those statements that are needed to replicate all procedures	None.
-t table	Limits the SQL statement output to only those statements that are needed to replicate the specified table	None.
-t view	Limits the SQL statement output to only those statements that are needed to replicate the specified view	None.
-t all	Includes in the SQL statement output all statements that are needed to replicate all tables and views	None.
-ti table	Includes in the SQL statement output all statements that are needed to replicate all table levels	None.
-ti all	Includes in the SQL statement output all statements that are needed to replicate all tables and views Functionally equivalent to -t all .	None.

For more information about the CREATE PROCEDURE and CREATE FUNCTION statements, see the *IBM Informix: Guide to SQL Syntax*.

Table Information

When you use the **-ss** option, you can retrieve information about fragmented tables, the lock mode, and extent sizes.

The following **dbschema** output shows the expressions specified for fragmented table.

```

DBSCHEMA Schema Utility      INFORMIX-SQL Version X.XX.UC1
Copyright (C) IBM Corporation, 1984-1997
{ TABLE "sallyc".t1 row size = 8 number of columns = 1 index size = 0 }
create table "sallyc".t1
(
c1 integer
) fragment by expression
(c1 < 100 ) in db1 ,
((c1 >= 100 ) AND (c1 < 200 ) ) in db2 ,
remainder in db4
extent size 16 next size 16 lock mode page;
revoke all on "sallyc".t1 from "public";

```

The following **dbschema** output shows information on partitions in partition-fragmented tables.

```

DBSCHEMA Schema Utility      INFORMIX-SQL Version 10.00.U  grant dba
to "sqlqa";
{ TABLE "sqlqa".t1 row size = 24 number of columns = 2 index size = 13 }
create table "sqlqa".t1
(
c1 integer,
c2 char(20)
)
fragment by expression
partition part_1 (c1 = 10 ) in dbs1 ,
partition part_2 (c1 = 20 ) in dbs1 ,
partition part_3 (c1 = 30 ) in dbs1 ,
partition part_4 (c1 = 40 ) in dbs1 ,
partition part_5 (c1 = 50 ) in dbs1
extent size 16 next size 16 lock mode page;

```

Role Creation

Roles:



Element	Purpose	Key Considerations
-r role	Displays the CREATE ROLE and GRANT statements that are needed to replicate and grant the specified role.	Restriction: You cannot specify a list of users or roles with the -r option. You can specify either one role or all roles. SE does not support the -r option.

Element	Purpose	Key Considerations
-r all	Displays all CREATE ROLE and GRANT statements that are needed to replicate and grant all roles.	None

The following **dbschema** command and output show that the role **calen** was created and was granted to **cathl**, **judith**, and **sallyc**:

```
sharky% dbschema -r calen -d stores_demo
```

```
DBSCHEMA Schema Utility          INFORMIX-SQL Version 7.22
Copyright (C) IBM Corporation, 1984-1996
Software Serial Number RDS#N0000000
create role calen;
```

```
grant calen to cathl with grant option;
grant calen to judith ;
grant calen to sallyc ;
```

Distribution Information for Tables

To display the distribution information that is stored for a table in a database, use the **-hd** option with the name of the table. If you specify the **ALL** keyword for the table name, the distributions for all the tables in the database are displayed.

Distribution information is stored only if you have run the **UPDATE STATISTICS...MEDIUM** or **HIGH** statement for one or more columns of a table. For information about the **UPDATE STATISTICS** statement, refer to the *IBM Informix: Guide to SQL Syntax*.

The output of **dbschema** for distributions is provided in the following parts:

- Distribution description
- Distribution information
- Overflow information

Each section of **dbschema** output is explained in the following sections. As an example, the discussion uses the following distribution for the fictional table called **invoices**. This table contains 165 rows, including duplicates.

You can generate the output for this discussion with a call to **dbschema** that is similar to the following example:

```
dbschema -hd invoices -d pubs_stores_demo
```

Example Output

Distribution for cath1.invoices.invoice_num

Constructed on 03/10/2001

High Mode, 10.000000 Resolution

--- DISTRIBUTION ---

```
(
1: ( 16, 7, 5)
2: ( 16, 6, 11)
3: ( 16, 8, 17)
4: ( 16, 8, 25)
5: ( 16, 7, 38)
6: ( 16, 8, 52)
7: ( 16, 8, 73)
8: ( 16, 12, 95)
9: ( 16, 12, 139)
10: ( 16, 11, 182)
10: ( 10, 5, 200)
```

--- OVERFLOW ---

```
1: ( 5, 56)
2: ( 6, 63)
}
```

Distribution Description

The first part of the **dbschema** output describes which data distributions have been created for the specified table. The name of the table is stated in the following example:

Distribution for cath1.invoices.invoice_num

The output is for the **invoices** table, which is owned by user cath1. This data distribution describes the column **invoice_num**. If a table has distributions that are built on more than one column, **dbschema** lists the distributions for each column separately.

The date on which the distributions are constructed is listed. In this example, the date is 03/10/2001, which is the date when the UPDATE STATISTICS statement that generated the distributions was executed. You can use this date to tell how outdated your distributions are. Although the system records the date, it does not record the time.

The last line of the description portion of the output describes the mode (medium or high) in which the distributions were created, and the resolution. If you create the distributions with medium mode, the confidence of the

sample is also listed. For example, if the UPDATE STATISTICS statement is executed with high mode with a resolution of 10, the last line appears as the following example shows:

High Mode, 10.000000 Resolution

Distribution Information

The distribution information describes the bins that are created for the distribution, the range of values in the table and in each bin, and the number of distinct values in each bin. Consider the following example:

	(5)	
1:	(16,	7,	11)
2:	(16,	6,	17)
3:	(16,	8,	25)
4:	(16,	8,	38)
5:	(16,	7,	52)
6:	(16,	8,	73)
7:	(16,	12,	95)
8:	(16,	12,	139)
9:	(16,	11,	182)
10:	(10,	5,	200)

The first value in the rightmost column is the smallest value in this column. In this example, it is 5.

The column on the left shows the bin number, in this case 1 through 10. The first number in parentheses shows how many values are in the bin. For this table, 10 percent of the total number of rows (165) is rounded down to 16. The first number is the same for all the bins except for the last. The last row might have a smaller value, indicating that it does not have as many row values. In this example, all the bins contain 16 rows except the last one, which contains 10.

The middle column within the parentheses indicates how many distinct values are contained in this bin. Thus, if there are 11 distinct values for a 16-value bin, it implies that one or more of those values are duplicated at least once.

The right column within the parentheses is the highest value in the bin. The highest value in the last bin is also the highest value in the table. For this example, the highest value in the last bin is 200.

Overflow Information

The last portion of the **dbschema** output shows values that have many duplicates. The number of duplicates of indicated values must be greater than a critical amount that is determined as approximately 25 percent of the resolution times the number of rows. If left in the general distribution data, the duplicates would skew the distribution, so they are moved from the distribution to a separate list, as the following example shows:

--- OVERFLOW ---

```
1: ( 5,      56)
2: ( 6,      63)
```

For this example, the critical amount is $0.25 * 0.10 * 165$, or 4.125. Therefore, any value that is duplicated five or more times is listed in the overflow section. Two values in this distribution are duplicated five or more times in the table: the value 56 is duplicated five times, and the value 63 is duplicated six times.

DB-Access Input from dbschema Output

You can use the **dbschema** utility to get the schema of a database and redirect the **dbschema** output to a file. Later, you can use this file as input to DB-Access to re-create the database.

Inserting a Table into a Database Example

The following example copies the CREATE TABLE statements for the customer table into the **dbschema** output file, **tab.sql**:

```
dbschema -d db -t customer > tab.sql
```

Remove the header information about **dbschema** from the output file, **tab.sql**, and then use DB-Access to re-create the table in another database, as follows:

```
dbaccess db1 tab.sql
```

Re-Creating the Schema of a Database

You can use **dbschema** and DB-Access to save the schema from a database and then re-create the schema in another database. A **dbschema** output file can contain the statements for creating an entire database.

To save a database schema and re-create the database:

1. Use **dbschema** to save the schema to an output file, such as **db.sql**:

```
dbschema -d db > db.sql
```

You can also use the **-ss** option to generate server-specific information:

```
dbschema -d db -ss > db.sql
```

2. Remove the header information about **dbschema**, if any, from the output file.
3. Add a CREATE DATABASE statement at the beginning of the output file or use DB-Access to create a new database.
4. Use DB-Access to re-create the schema in a new database:

```
dbaccess - db.sql
```

When you use **db.sql** to create a database on a different database server, confirm that dbspaces exist.

The databases **db** and **testdb** differ in name but have the same schema.

Chapter 11. The LOAD and UNLOAD Statements

Syntax of the UNLOAD Statement	11-1
Syntax of the LOAD Statement	11-2
Using Load and Unload Statements for Locales That Support Multibyte Code Sets	11-2

In This Chapter

This chapter shows the syntax of the SQL UNLOAD and LOAD statements. You can use UNLOAD and LOAD with the following database servers:

- Dynamic Server 10.0, 9.40, 9.3x, 9.2x, 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE
- OnLine 5.1x

Syntax of the UNLOAD Statement

You can use the UNLOAD statement in DB–Access to unload selected rows from a table into a text file.

```
▶▶ UNLOAD TO 'filename' [ DELIMITER 'delimiter' ] SELECT Statement (1) ▶▶
```

Notes:

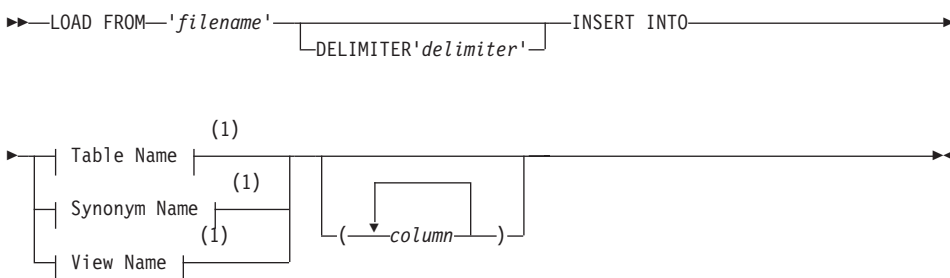
- 1 See the *IBM Informix: Guide to SQL Syntax*.

Element	Purpose	Key Considerations
<i>delimiter</i>	Character to use as delimiter	Restrictions: See “Syntax for the Delimiter Form” on page 9-6
<i>filename</i>	Specifies the input file	None.

This syntax diagram is only for quick reference. For details about the syntax and use of the UNLOAD statement, see the *IBM Informix: Guide to SQL Syntax*.

Syntax of the LOAD Statement

You can use the LOAD statement in DB–Access to append rows to an existing table of a database.



Notes:

- 1 See *IBM Informix: Guide to SQL Syntax*

Element	Purpose	Key Considerations
<i>column</i>	The name of a column to receive data from <i>filename</i>	Restrictions: Must be a column in the specified table or view.
<i>delimiter</i>	Character to use as delimiter	Restrictions: See “Syntax for the Delimiter Form” on page 9-6
<i>filename</i>	Specifies the input file	None.

This syntax diagram is only for quick reference. For details about the syntax and use of the LOAD statement, see the *IBM Informix: Guide to SQL Syntax*.

Using Load and Unload Statements for Locales That Support Multibyte Code Sets

For locales that support multibyte code sets, be sure that the declared size (in bytes) of any column that receives character data is large enough to store the entire data string. For some locales, this can require up to 4 times the number of logical characters in the longest data string.

Chapter 12. The onmode Utility

Use of the onmode -b Command for Reversion	12-1
Preparation for Reversion.	12-2
Syntax of the onmode -b Command	12-2

In This Chapter

This chapter describes the **-b** option of the **onmode** utility and how to use it. You can use **onmode -b** with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x
- Workgroup Edition 7.3x or 7.24
- SE
- OnLine 5.1x

After you convert to a newer database server, you can use the **-b** option of the **onmode** utility for reversion to the older database server from which you converted. The **onmode** utility modifies the data in an Informix database so that the earlier version of the database server can access it. For information about other **onmode** options, see your *IBM Informix: Administrator's Guide*.

Use of the onmode -b Command for Reversion

When you convert a database server, several modifications make the format of the databases incompatible with the older version. The **onmode -b** command restores the databases to a format that is compatible with the earlier version. You must revert the databases before users can access the data with the earlier database server version. The utility does not revert changes made to the layout of the data that do not affect compatibility.

————— UNIX/Linux Only —————

You must be user **root** or user **informix** to execute **onmode**.

————— End of UNIX/Linux Only —————

————— Windows Only —————

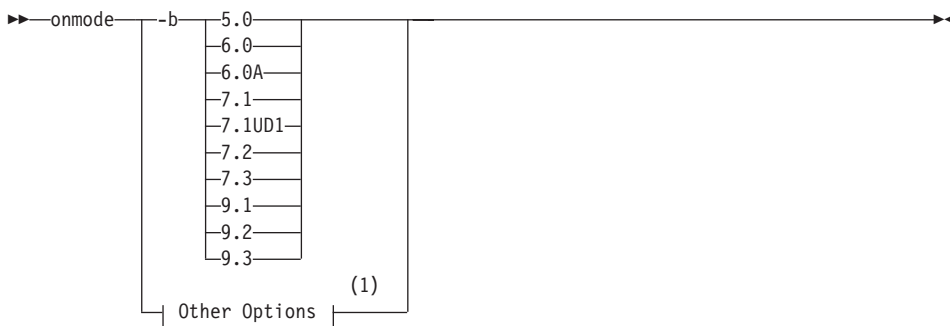
You must be a member of the **Informix-Admin** group to execute **onmode**.

————— End of Windows Only —————

Preparation for Reversion

Before you use the **-b** option, notify users that you are going to bring the database server offline. The reversion utility forcibly removes all users and shuts down the database server. The **-b** option includes an implicit **-yuk**. Make sure that the **INFORMIXSERVER** environment variable is set to the correct database server.

Syntax of the onmode -b Command



Notes:

- 1 For all other onmode options, see your *IBM Informix: Administrator's Reference*.

Element	Purpose	Key Considerations
-b 6.0	Changes the database to the Version 6.0 format	Additional Information: See the 9.2/8.3 version of the <i>IBM Informix: Migration Guide</i> .
-b 6.0A	Changes the database to the Version 6.0 ALS format	Additional Information: See the 9.2/8.3 version of the <i>IBM Informix: Migration Guide</i> .
-b 7.1	Changes the database to the Version 7.10.UC1 format, which is compatible with all 7.10.UCx formats	Additional Information: See the 9.2/8.3 version of the <i>IBM Informix: Migration Guide</i> .
-b 7.1UD1	Changes the database to the Version 7.1UD1 format, which is compatible with 7.11, 7.12, 7.13, and 7.14 formats	Additional Information: See the 9.2/8.3 version of the <i>IBM Informix: Migration Guide</i> .
-b 7.2	Changes the database to the Version 7.2x format	Additional Information: See “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
-b 7.3	Changes the database to the Version 7.3x format	Additional Information: See “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
-b 9.1	Changes the database to the Version 9.14 format	Additional Information: See “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
-b 9.2	Changes the database to the Version 9.2x format	Additional Information: See “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.
-b 9.3	Changes the database to the Version 9.30 format	Additional Information: See “Reverting from Dynamic Server 10.0 to Versions 9.40, 9.30, 9.2x, 7.3x, or 7.24” on page 6-1.

Notes:

1. You cannot use redirection to a file when you execute **onmode -b 7.2** because this command does not function with redirection.
2. To list the available options for your database server, type **onmode -b -**.

Chapter 13. The **onunload** and **onload** Utilities

How onunload and onload Work	13-1
Syntax of the onunload Command.	13-2
Destination Parameters	13-3
Constraints That Affect onunload	13-3
Database or Table Unloading	13-4
Unloading a Database	13-4
Unloading a Table	13-4
Logging Mode	13-5
Locking During Unload Operation.	13-5
Syntax of the onload Command	13-5
Source Parameters	13-6
Create Options	13-7
Constraints That Affect onload	13-8
Constraints That Affect onload and onunload	13-9
Restrictions That Affect onload and onunload	13-10
Logging During Loading	13-10
Movement of Simple Large Objects to a BlobSpace	13-11
Ownership and Privileges	13-11
Exclusive Locking During Load Operation.	13-11
Steps for Using onunload and onload	13-11

In This Chapter

This chapter describes the **onunload** and **onload** utilities and how to use them. These utilities unload and load databases and tables. You can use **onunload** and **onload** with the following database servers:

- Dynamic Server 10.0, 9.40, 9.30, 9.2x, 7.3x or 7.24
- Dynamic Server, Linux Edition 7.3x and Workgroup Edition 7.3x or 7.24

Important: You can use **onunload** and **onload** with Dynamic Server 10.0, 9.40, 9.30, or 9.2x only if the databases contain only legacy data types and no extended data types.

How **onunload** and **onload** Work

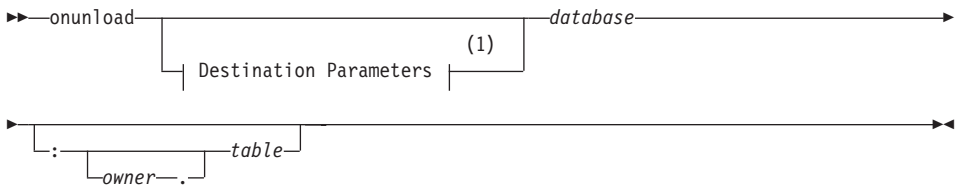
The **onunload** utility unloads data from a database. The **onunload** utility writes a database or table into a file on tape or disk. The **onunload** utility unloads the data in binary form in disk-page units, making this utility more efficient than **dbexport**. You can use the **onunload** utility to move data between computers that have the same version of the database server.

Note: You cannot use **onunload** and **onload** to move data between different versions of Dynamic Server.

The **onload** utility loads data that was created with the **onunload** command into the database server. The **onload** utility creates a database or table in a specified dbspace. The **onload** utility then loads it with data from an input tape or disk file that the **onunload** utility creates.

During the load, you can move simple large objects that are stored in a blobspace to another blobspace.

Syntax of the onunload Command



Notes:

1 See page 13-3

Element	Purpose	Key Considerations
<i>database</i>	Specifies the name of a database	Additional Information: The database name cannot be qualified by a database server name (<i>database@dbservername</i>). References: Syntax must conform to the Identifier segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .
<i>owner.</i>	Specifies the owner of the table	Additional Information: The owner name must not include invalid characters. References: For pathname syntax, see your operating-system documentation.
<i>table</i>	Specifies the name of the table	Restriction: The table must exist. References: Syntax must conform to the Table Name segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .

If you do not specify any destination parameter options, **onunload** uses the device that TAPEDEV specifies. The block size and tape size are the values specified as TAPEBLK and TAPESIZE, respectively. (For information about TAPEDEV, TAPEBLK, and TAPESIZE, see your *IBM Informix: Administrator's Guide*.)

Destination Parameters

Destination Parameters:



Notes:

- 1 Only one occurrence of each option allowed. More than one option may occur in a single invocation.

Element	Purpose	Key Considerations
-b <i>blocksizes</i>	Specifies in kilobytes the block size of the tape device	Restrictions: The <i>blocksizes</i> must be an integer. Additional Information: This option overrides the default value in TAPEBLK or LTAPEBLK.
-l	Directs onunload to read the values for tape device, block size, and tape size from LTAPEDEV, LTAPEBLK, and LTAPESIZE, respectively	None.
-s <i>tapesize</i>	Specifies in kilobytes the amount of data that can be stored on the tape	Restrictions: The <i>tapesize</i> must be an integer. If you do not specify 0, then the maximum <i>tapesize</i> is 2,097,151 kilobytes Additional Information: This option overrides the default value in TAPESIZE or LTAPESIZE. To write to the end of the tape, specify a tape size of 0.
-t <i>source</i>	Specifies the pathname of the file on disk or of the tape device where the input tape is mounted	Additional Information: This option overrides the tape device specified by TAPEDEV or LTAPEDEV. It must be a valid pathname.

Constraints That Affect onunload

The **onunload** utility can unload data more quickly than either **dbexport** or the UNLOAD statement because **onunload** copies the data in binary format and in page-sized units. The following constraints apply to **onunload**:

- You must load the data on the **onunload** tape into a database or table that your database server (excluding SE) manages.
- You can use **onunload** and **onload** with Dynamic Server 10.0, 9.40, 9.30, or 9.2x if the databases contain only legacy data types and no extended data types.

- You must load the tape that **onunload** writes onto a computer with the same page size and the same representation of numeric data as the original computer.
- You must read the file that **onunload** creates with the **onload** utility of the same version of your database server. You cannot use **onunload** and **onload** to move data from one version to another.
- When you unload a complete database, you cannot modify the ownership of database objects (such as tables, indexes, and views) until after you finish reloading the database.
- When you unload and load a table, **onunload** does not preserve access privileges, synonyms, views, constraints, triggers, or default values that were associated with the original tables. Before you run **onunload**, use the **dbschema** utility to obtain a listing of the access privileges, synonyms, views, constraints, triggers, and default values. After you finish loading the table, use **dbschema** to re-create the specific information for the table.

Database or Table Unloading

To unload a database, you must have DBA privileges for the database or be user **informix**. To unload a table, you must either own the table, have DBA privileges for the database in which the table resides, or be user **informix**. (User **root** does not have special privileges with respect to **onunload** and **onload**.)

Unloading a Database

If you unload a database, all the tables in the database, including the system catalog tables, are unloaded. All the triggers, SPL routines, defaults, constraints, and synonyms for all the tables in the database are also unloaded.

Unloading a Table

If you unload a table, **onunload** unloads the table data and information from the following system catalog tables:

systables
syscolumns
sysindexes
sysblobs

When you unload a table, **onunload** does not unload information about constraints, triggers, or default values that are associated with a table. In addition, access privileges that are defined for the table and synonyms or views that are associated with the table are not unloaded.

Logging Mode

The **onunload** utility does not preserve the logging mode of a database. After you load the database with **onload**, you can make a database ANSI compliant or add logging. For information about logging modes, refer to the *IBM Informix: Guide to SQL Syntax*.

During the load, you can move simple large objects that are stored in a blobspace to another blobspace.

If you do not specify any source-parameter options, **onload** uses the device that is specified as TAPEDEV. The block size and tape size are the values that are specified as TAPEBLK and TAPESIZE, respectively. (For more information about TAPEDEV, TAPEBLK, and TAPESIZE, refer to your *IBM Informix: Dynamic Server Administrator's Guide*.)

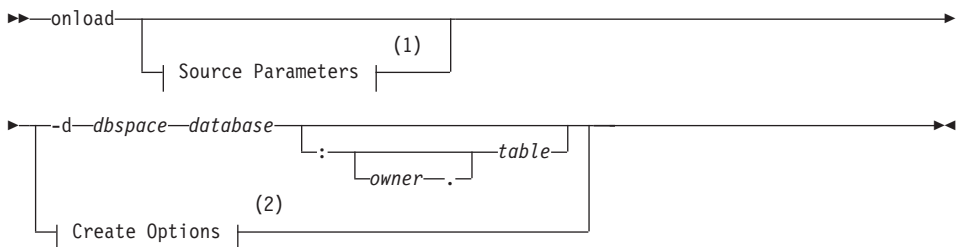
If you do not specify creation options, **onload** stores the database or table in the root dbspace.

Locking During Unload Operation

During the unload operation, the database or table is locked in shared mode. An error is returned if **onunload** cannot obtain a shared lock.

The **onload** utility creates a database or table in a specified dbspace (excluding SE). The **onload** utility then loads it with data from an input tape or disk file that the **onunload** utility creates.

Syntax of the onload Command



Notes:

- 1 See page 13-6
- 2 See page 13-7

Element	Purpose	Key Considerations
-d dbspace	Loads a database or table into the specified dbspace	Restriction: The tape being loaded must contain the specified database or table.
<i>database</i>	Specifies the name of the database	Restriction: The database name cannot include a database server name, such as <i>database@dbservername</i> . References: Syntax must conform to the Identifier segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .
<i>owner.</i>	Specifies the owner of the table	Restriction: The owner name must not include invalid characters. References: For pathname syntax, refer to your operating-system documentation.
<i>table</i>	Specifies the name of the table	Restriction: The table must not exist. References: Syntax must conform to the Table Name segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .

Source Parameters

Source Parameters:



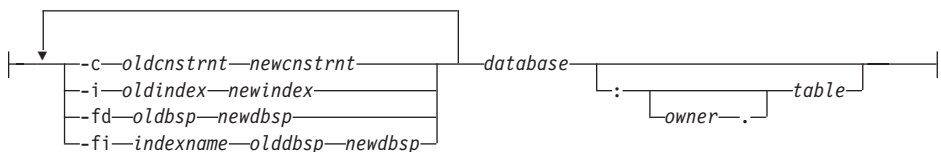
Notes:

- 1 Only one occurrence of each option allowed. More than one option may occur in a single invocation.

Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies in kilobytes the block size of the tape device	<p>Restrictions: Unsigned integer. Must specify the block size of the tape device.</p> <p>Additional Information: This option overrides the default value in TAPEBLK or LTAPEBLK.</p>
-l	Directs onload to read the values for tape device, block size, and tape size from the configuration parameters LTAPEDEV, LTAPEBLK, and LTAPESIZE, respectively	<p>Additional Information: If you specify -l, and then -b, -s, or -t, the value that you specify overrides the value in the configuration file.</p>
-s <i>tapesize</i>	Specifies in kilobytes the amount of data that the database server can store on the tape	<p>Restrictions: Unsigned integer. If you do not specify 0, then the maximum <i>tapesize</i> is 2,097,151 kilobytes.</p> <p>Additional Information: This option overrides the default value in TAPESIZE or LTAPESIZE. To write to the end of the tape, specify a tape size of 0.</p>
-t <i>source</i>	Specifies the pathname of the file on disk or of the tape device where the input tape is mounted	<p>Restriction: Must be a legal pathname.</p> <p>Additional Information: This option overrides the tape device that TAPEDEV or LTAPEDEV specifies.</p> <p>References: For pathname syntax, see your operating-system documentation.</p>

Create Options

Create Options:



Element	Purpose	Key Considerations
<i>-c oldcnstrnt newcnstrnt</i>	Directs onload to rename the specified constraint.	None.
<i>-i oldindex newindex</i>	Directs onload to rename the table index when it stores the index on disk.	Additional Information: Use the -i option to rename indexes during the load to avoid conflict with existing index names. References: Syntax must conform to the Identifier segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .
<i>-fd olddbsp newdbsp</i>	Allows you to move a data fragment from one dbspace to another.	Restriction: The new dbspace must exist and not already contain another data fragment for the table. Additional Information: This option is used with parallel data query (PDQ) and table fragmentation.
<i>-fi indexname olddbbs newdbsp</i>	Allows you to move index fragments from one dbspace to another.	Restriction: The new dbspace must exist and not already contain another index fragment for the table. Additional Information: This option is used with PDQ and table fragmentation.
<i>database</i>	Specifies the name of the database	Restriction: The database name cannot include a database server name, such as <i>database@dbservername</i> . References: Syntax must conform to the Identifier segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .
<i>owner.</i>	Specifies the owner of the table	Restriction: The owner name must not include illegal characters. References: For pathname syntax, refer to your operating-system documentation.
<i>table</i>	Specifies the name of the table	Restriction: The table must not exist. References: Syntax must conform to the Table Name segment; see the <i>IBM Informix: Guide to SQL Syntax</i> .

If you do not specify any create options, the **onload** utility stores the database or table in the root dbspace.

You can use the **-c**, **-i**, **-fd**, and **-fi** options in any order and as often as necessary as long as you use unique pairs.

Constraints That Affect onload

The **onload** utility performs faster than the **dbimport**, **dbload**, or **LOAD** methods. In exchange for this higher performance, **onload** has the following constraints:

- The **onload** utility can only create a new database or table; you must drop or rename an existing database or table of the same name before you run **onload**. During execution, the **onload** utility's prompt will ask you if you want to rename blobspaces.
- The **onload** utility places a shared lock on each of the tables in the database during the load. While you cannot update a table row with the lock in place, the database is available for queries.
- When you load a complete database, the user who executes **onload** becomes the owner of the database.
- The **onload** utility creates a database without logging; you must initiate logging after **onload** loads the database.
- When you use **onload** to load a table into a logged database, you must turn off logging for the database during the operation.
- The **onload** utility does not preserve dbspace assignments of table creation or table fragmentation.

Constraints That Affect onload and onunload

Global Language Support

Native Language Support

You can use **onunload** and **onload** to move data between databases if the NLS and GLS locales are identical. For example, if user A has a French locale NLS table on server A and tries to load data into a German locale GLS table on server B, **onload** and **onunload** report errors. However, if both the NLS and GLS tables were created with the same French locale, **onload** and **onunload** would work.

End of Native Language Support

End of Global Language Support

The tape that **onload** reads contains binary data that is stored in disk-page-sized units. For this reason, the computer where the original database resides (where you use **onunload**) and the computer where the target database will reside (where you use **onload**) must share the following characteristics:

- The same page size
- The same representation of numeric data
- The same byte alignment for structures and unions

If the page sizes are different, **onload** fails. If the alignment or numeric data types on the two computers are different (for example, with the most significant byte last instead of first, or different float-type representations), the contents of the data page could be misinterpreted.

Restrictions That Affect **onload** and **onunload**

The **onload** and **onunload** utilities have the following restrictions:

- The original database and the target database must be from the same version of the database server.

Global Language Support

- You cannot use **onload** and **onunload** to move data between non-GLS and GLS locales.

End of Global Language Support

- Do not use **onload** and **onunload** to move data between two Dynamic Server 10.0, 9.40, 9.30, or 9.2x databases if they contain extended data types. Use the HPL instead to move 9.40, 9.30, or 9.2x data. However, you can use **onload** and **onunload** with this data if the databases contain only legacy data types.

Note: You cannot use the **onload** and **onunload** utilities to move data from one version to another. You also cannot use these utilities to move data between different types of database servers.

Logging During Loading

The **onload** utility performs all its loading within a transaction. This feature allows the changes to be rolled back if an error occurs.

When you use **onload** to create tables from an **onunload** input tape, **onload** can only load information into a database without logging. Thus, before you load a table into an existing, logged database, end logging for the database. You also might want to consider loading during off-peak hours. Otherwise, you might fill the logical-log files or consume excessive shared-memory resources. After you load the table, create a level-0 dbspace backup before you resume database logging.

When you use **onload** to create databases from an **onunload** input tape, the databases that result are not ANSI compliant and do not use transaction logging. You can make a database ANSI compliant or add logging after you load the database. (For more information about logging, refer to the *IBM Informix: Guide to SQL Reference*.)

Movement of Simple Large Objects to a BlobSpace

If you load a table that contains simple large objects stored in a blobSpace, **onload** asks you if you want to move them to another blobSpace. If you respond yes, **onload** displays the blobSpace name where the simple large objects were stored when the tape was created. It then asks you to enter the name of the blobSpace where you want the simple large objects stored. If you enter a valid blobSpace name, **onload** moves all simple-large-object columns in the table to the new blobSpace. Otherwise, **onload** prompts you again for a valid blobSpace name.

Ownership and Privileges

When you load a new database, the user who runs **onload** becomes the owner. Ownership within the database (tables, views, and indexes) remains the same as when the database was unloaded to tape with **onunload**.

To load a table, you must have the Resource privilege on the database. When **onload** loads a new table, the user who runs **onload** becomes the owner unless you specify an owner in the table name. (You need the DBA privilege for the database to specify an owner in the table name.)

The **onunload** utility does not preserve synonyms or access privileges. To obtain a listing of defined synonyms or access privileges, use the **dbschema** utility, which Chapter 10, "The dbschema Utility," on page 10-1 describes, before you run **onunload**.

Exclusive Locking During Load Operation

During the load operation, **onload** places an exclusive lock on the new database or table. Loading proceeds as a single transaction, and **onload** drops the new database or table if an error or system failure occurs.

Steps for Using **onunload** and **onload**

This section describes the procedure for using **onunload** and **onload** to move a database. You can use these commands to move either a complete database or a table from one computer to another. The syntax and description of the **onunload** utility starts on page 13-2. The syntax and description of the **onload** utility starts on page 13-5.

To move a database from one computer to another:

1. Make sure that the page size, numeric representations, and byte alignment on structures and unions are the same on both computers. (The page size is two kilobytes on certain UNIX systems and four kilobytes on Windows NT.) The page size is an Informix characteristic. For information about page size, see your *IBM Informix: Administrator's Guide*. The numeric representation and the byte alignment are

characteristics of your operating system. For information about numeric representation and byte alignment, refer to the manuals for your operating systems.

2. Decide where to store the unloaded data:
 - **On disk.** Create an empty file for **onunload** to hold the data. Make sure that you have write permission for the file.
 - **On tape.** Use the tape device and characteristics specified in the ONCONFIG configuration file by either TAPEDEV or LTAPEDEV or specify another tape device. Make sure that the tape device that you specify is available for **onunload** .
3. Run the **oncheck** utility to make sure that your database is consistent. For information about **oncheck**, see your *IBM Informix: Administrator's Guide*.
4. Run the **onunload** utility to unload the data from the database.
5. If necessary, transfer the storage medium (tape or disk) to the new computer.

If the two computers are on the same network, you can read or write the data remotely.
6. Run the **onload** utility to load the data into the new database.
7. Set the desired logging status for the new database.

For information about logging status, see your *IBM Informix: Administrator's Guide*.
8. If necessary, change the DBA privileges of the database.
9. If you want to restore the triggers, access privileges, SPL routines, defaults, constraints, and synonyms for the tables in the database, run the **dbschema** utility.
10. Create a level-0 backup of the new database.

To move a table from one computer to another:

1. Make sure that the page size, numeric representations, and byte alignment on structures and unions are the same on both computers. (The page size is two kilobytes on certain UNIX systems and four kilobytes on Windows NT.)
2. Decide where to store the unloaded data. (See step 2 of the previous section.)
3. Run the **oncheck** utility to make sure that your database is consistent.
4. If you want to save the triggers, access privileges, SPL routines, defaults, constraints, and synonyms for the table, run the **dbschema** utility.
5. Run the **onunload** utility.
6. If necessary, transfer the storage medium to the new computer.

7. If the table includes simple large objects that are stored in blobspaces, decide where to store the simple large objects. If necessary, create new blobspaces.
8. Turn off logging.
When you are loading a table, logging on the target database must be turned off. (When you are creating and loading an entire database, the logging status does not matter.)
9. Run the **onload** utility.
10. Create a level-0 backup of the modified database.
11. Turn logging back on, if you want logging.
12. If you want to restore the triggers, access privileges, SPL routines, defaults, constraints, and synonyms for the table, run the **dbschema** utility or create them manually.

To move a table from one dbspace to another dbspace on the same computer:

1. Run the **onunload** utility to unload the table.
2. Turn off logging.
When you are loading a table, logging on the target database must be turned off.
3. Run the **onload** utility.
Specify a new table name and new dbspace name in the **onload** statement.
4. If the data loads successfully, delete the old table in the old dbspace and rename the new table to the old table name.
5. Create a level-0 backup of the modified database.
6. Turn logging back on, if you want logging.

Part 5. Appendixes

Appendix. Accessibility

The syntax diagrams in the HTML version of this manual are available in dotted decimal syntax format, which is an accessible format that is available only if you are using a screen reader.

Dotted Decimal Syntax Diagrams

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, this identifies a reference that 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

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years).
All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

AIX; DB2; DB2 Universal Database; Distributed Relational Database Architecture; NUMA-Q; OS/2, OS/390, and OS/400; IBM Informix®; C-ISAM®; Foundation.2000™; IBM Informix® 4GL; IBM Informix® DataBlade® Module; Client SDK™; Cloudscape™; Cloudsync™; IBM Informix® Connect; IBM Informix® Driver for JDBC; Dynamic Connect™; IBM Informix® Dynamic Scalable Architecture™ (DSA); IBM Informix® Dynamic Server™; IBM Informix® Enterprise Gateway Manager (Enterprise Gateway Manager); IBM Informix® Extended Parallel Server™; i.Financial Services™; J/Foundation™; MaxConnect™; Object Translator™; Red Brick™; IBM Informix® SE; IBM Informix® SQL; InformiXML™; RedBack®; SystemBuilder™; U2™; UniData®; UniVerse®; wintegrate® are trademarks or registered trademarks of International Business Machines Corporation.

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

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Index

A

Abbreviated years 8-1
Accessibility xxii
 dotted decimal format of syntax diagrams A-1
 syntax diagrams, reading in a screen reader A-1
AFDEBUG environment variable 3-5
AFF_NPROCS configuration parameter 3-11
AFF_SPROC configuration parameter 3-11
ALARMPROGRAM configuration parameter 3-10,
 3-19, 5-11, 5-12
ALL keyword
 dbschema utility 10-3
ALLOW_NEWLINE configuration parameter 3-9
ALRM_ALL_EVENTS configuration parameter 3-6
Applications
 client
 Communications Support Module 5-12
archecker utility 3-25
ASCII file, produced by unloading 2-4
ASF_SOCTCP_BACKLOG configuration
 parameter 3-7
Authentication
 policy 5-12, 6-10

B

Backup
 Dynamic Server 10.0 5-16, 6-8
 Dynamic Server source when converting 5-9
 logical logs 5-12
 ON-Bar 5-9, 5-16, 6-11
 ontape utility 5-9, 5-16, 6-11
 source database 5-9
Binary files, loading 2-5, 2-7, 13-9
BladeManager
 installing and registering DataBlade modules 5-12
 removing extensions during reversion 6-8
Blobs.
 See Simple large objects.
Blobspace
 backing up 3-4
 moving, with onunload and onload 13-13
Boldface type xiii
boot90.sql and boot901.sql 6-1
BUFFERPOOL configuration parameter 3-6, 3-19
BUFFERS configuration parameter 3-10, 3-19

C

Case-sensitive name space 3-21
CDR_DBSPACE configuration parameter 3-8

CDR_ENV configuration parameter 3-8
CDR_LOGBUFFERS configuration parameter 3-11
CDR_LOGDELTA configuration parameter 3-11
CDR_LOGDELTA environment variable 3-5
CDR_MAX_DYNAMIC_LOGS configuration
 parameter 3-8
CDR_NIFRETRY configuration parameter 3-11
CDR_NUMCONNECT configuration parameter 3-11
CDR_PERFLOG environment variable 3-5
CDR_QDATA_SBFLAGS configuration parameter 3-10
CDR_QDATA_SBSPACE configuration parameter 3-8,
 3-10
CDR_QHDR_DBSPACE configuration parameter 3-8
CDR_RMSCALEFACT environment variable 3-5
CDR_ROUTER environment variable 3-5
CDR_SUPPRESS_ATSRISWARN configuration
 parameter 3-6
Character-position form of FILE and INSERT
 statements 9-10
Checking available space
 Dynamic Server 10.0 5-3
Checking database integrity 5-16
Choosing a migration method 1-1
Chunk
 format 3-20
 reserve pages
 and reversion from Dynamic Server 6-8
Client application
 Communications Support Module 5-12
Closing
 Dynamic Server 5-6, 5-9
Code-set conversion
 HPL 2-12
Code, sample, conventions for xviii
Column-level encryption 6-4
Command file
 dbload 9-5
Command-line conventions
 how to read xvi
 sample diagram xvi
Communications Support Module
 client applications 5-12
 configuring 5-12
 removing 6-10
 saving configuration 6-8
Compliance
 with industry standards xxv
conscm.cfg file 5-12
 creating 5-12

concsm.cfg file (*continued*)

- deleting 6-10
- saving 6-8

Configuration file

- customizing 5-11
- replacing after reversion from Dynamic Server 10.0 6-10
- saving a copy of current version 6-8
- saving during conversion 5-5

Configuration parameter

- ALARMPROGRAM 5-11, 5-12
- ALLOW_NEWLINE 3-9
- ALRM_ALL_EVENTS 3-6
- altered in Dynamic Server 9.30
 - JDKVERSION 3-11
 - JVPJAVAHOME 3-11
 - JVPJAVALIB 3-11
 - JVPJAVAVM 3-11
- altered in Dynamic Server 9.40
 - ALARMPROGRAM 3-10
 - CDR_QDATA_SBSpace 3-10
 - DBSERVERALIASES 3-10
 - LRU_MAX_DIRTY 3-10
 - LRU_MIN_DIRTY 3-11
 - LTAPEBLK 3-10
 - LTAPESIZE 3-10
 - OPTICAL_LIB_PATH 3-11
 - TAPEBLK 3-11
 - TAPESIZE 3-11
- ASF_SOCTCP_BACKLOG 3-7
- BUFFERPOOL 3-6, 3-19
- BUFFERS 3-10, 3-19
- CDR_DBSPACE 3-8
- CDR_ENV 3-8
- CDR_MAX_DYNAMIC_LOGS 3-8
- CDR_QDATA_SBSpace 3-8
- CDR_QHDR_DBSPACE 3-8
- CDR_SUPPRESS_ATSRISWARN 3-6
- DD_HASHMAX 3-9
- DD_HASHSIZE 3-9
- DEF_TABLE_LOCKMODE 3-8
- DR_IDXAUTO 3-7
- DRAUTO 3-19
- DS_HASHSIZE 3-9
- DS_NONPDQ_MEM 3-7
- DS_POOLSIZe 3-9
- DYNAMIC_LOGS 3-9
- ENCRYPT_CDR 3-8
- ENCRYPT_CIPHER 3-8
- ENCRYPT_MAC 3-8
- ENCRYPT_MACFILE 3-8
- ENCRYPT_SWITCH 3-8
- EXT_DIRECTIVES 3-7
- FAST_RESTART_CKPT_FUZZYLOG 3-7
- FAST_RESTART_PHYSLOG 3-7

Configuration parameter (*continued*)

- HPL_DYNAMIC_LIB_PATH 3-8, 5-11
- IFX_EXTEND_ROLE 3-7
- IFX_LISTEN_TIMEOUT 3-7
- LRU_MAX_DIRTY 3-10, 3-19
- LRU_MIN_DIRTY 3-19
- LRUS 3-10, 3-19
- LTAPEBLK 3-26
- MAX_INCOMPLETE_CONNECTIONS 3-7
- new in Dynamic Server 10.0 3-6
- new in Dynamic Server 9.20 3-9
- new in Dynamic Server 9.21 3-9
- new in Dynamic Server 9.30 3-8
- new in Dynamic Server 9.40 3-7
- ONLIDX_MAXMEM 3-7
- OPT_GOAL 3-9
- OPTICAL_LIB_PATH 5-11
- PC_HASHSIZE 3-9
- PC_POOLSIZe 3-9
- PLOG_OVERFLOW_PATH 3-8
- removed in Dynamic Server 9.30
 - AFF_NPROCS 3-11
 - AFF_SPROC 3-11
 - CDR_LOGBUFFERS 3-11
 - CDR_LOGDELTA 3-11
 - CDR_NIFRETRY 3-11
 - CDR_NUMCONNECT 3-11
 - LBU_PRESERVE 3-11
 - LOGSMAX 3-11
 - LTXEHWM 3-11
 - LTXHWM 3-11
 - NOAGE 3-11
 - NUMAIOVPS 3-11
 - NUMCPOVPS 3-11
- removed in Dynamic Server 9.40
 - CDR_QDATA_SBFflags 3-10
- ROOTOFFSET 5-11
- ROOTPATH 5-11
- ROOTSIZE 5-11
- SBSpaceTEMP 3-9
- STMT_CACHE 3-9
- STMT_CACHE_HITS 3-9
- STMT_CACHE_NOLIMIT 3-9
- STMT_CACHE_NUMPOOL 3-9
- STMT_CACHE_SIZE 3-10
- SYSSBSpaceNAME 3-10
- TAPEBLK 3-26
- TBLTBLFIRST 3-7
- TBLTBLNEXT 3-7
- conpload.sh script 5-14
- conploadlegacy.sh script 5-14
- Contact information xxvi
- Conventions
 - command-line xvi
 - documentation xiii

- Conventions (*continued*)
 - sample-code xviii
 - syntax diagrams xv
 - syntax notation xiv
 - typographical xiii
- Converting
 - monitoring status with online.log 5-13
- Converting to Dynamic Server 10.0 with Enterprise Replication 4-1
- CREATE PROCEDURE statement, use of
 - dbschema 10-4
- CREATE SEQUENCE statement 10-4
- CREATE SYNONYM statement
 - use of dbschema 10-4
- CREATE TABLE statement
 - use of dbschema 10-4
- CREATE VIEW statement
 - use of dbschema 10-4
- Creating a dbload command file 9-5
- Customizing
 - configuration file 5-11
- D**
- Data integrity 5-15
- Data type
 - user defined
 - converting from Dynamic Server 9.2x 4-2
- Database
 - ownership, set by onload 13-11
 - verifying integrity 5-16
- Database server
 - platforms and versions 1-1
 - upgrading 3-1, 5-1
- DataBlade API
 - enhancements 3-28, 3-29
- DataBlade module
 - installing and registering 5-12
- dbexport
 - SELECT triggers, disabling 8-1
- dbexport utility 8-1
 - c option 8-3, 8-4
 - d option 8-3
 - q option 8-3
 - ss option 8-3, 8-4
 - V option 8-3
 - X option 8-3
 - defined 2-8, 8-1
 - destination options 8-4
 - schema output 8-6
 - syntax 8-2
- dbimport utility 8-1
 - c option 8-8, 8-9
 - l option 8-13
 - q option 8-8
 - V option 8-8
- dbimport utility (*continued*)
 - X option 8-8
 - create options 8-11
 - database logging mode 8-13
 - defined 2-8, 8-1
 - importing from another computer 2-14
 - input file location options 8-9
 - Interrupt key 8-9
 - locale, changing 8-14
 - syntax 8-7
 - using with GLS 8-8
 - using with NLS 8-14
- dbload utility
 - c command file option 9-1
 - d database option 9-2
 - e errors option 9-2
 - e option 9-3
 - i ignore rows option 9-2
 - i option 9-3
 - k option 9-2
 - l error log file option 9-2
 - n commit interval option 9-2
 - p option 9-2
 - r option 9-2, 9-3
 - s option 9-2
 - V option 9-3
 - X option 9-3
 - compared to LOAD 2-10
 - creating a command file 9-5
 - defined 9-1
 - FILE statement 9-5
 - ignoring rows 9-3
 - importing from another computer 2-14
 - INSERT statement
 - compared to SQL INSERT statement 9-12
 - using 9-5, 9-7
 - Interrupt key 9-4
 - moving data to Workgroup Edition 7-4
 - number errors to allow 9-3
 - speed, increasing 9-4
 - syntax 9-1
 - table locking 9-3
 - using 7-3
 - writing a command file
 - in character-position form 9-12
 - in delimiter form 9-8
- dbschema utility
 - ss option 10-5
 - create schema for a database 6-3, 10-3
 - defined 2-10, 10-1
 - distribution information 10-11
 - moving data to Workgroup Edition 7-4
 - obtaining
 - privilege schema 10-7
 - sequence schema 10-6

- dbschema utility (*continued*)
 - obtaining (*continued*)
 - synonym schema 10-7
 - owner conventions 10-4
 - specifying a table, view, or procedure 10-9
 - specifying the role schema 10-10
 - using 7-3
- DBSERVERALIASES configuration parameter 3-10
- DbSPACE
 - backing up 3-4
 - moving tables to another dbSPACE 13-13
- DD_HASHMAX configuration parameter 3-9
- DD_HASHSIZE configuration parameter 3-9
- DEF_TABLE_LOCKMODE configuration parameter 3-8
- Default locale xii
- Delimiter form of FILE and INSERT statements 9-6
- Dependencies, software x
- Detached index 3-20
- Diagnostic information to gather 3-2
- Diagnostics table
 - displayed by dbschema utility 2-11, 10-14
- Directory
 - installation 5-10
- Disabilities, visual
 - reading syntax diagrams A-1
- Disabling
 - High-Availability Data Replication 5-8
- Display schema for a database 10-3
- Distributed transactions
 - limitation on using routines 3-21
- Distribution information for tables 10-11
- Documentation conventions xiii
- Documentation Notes xx
- Documentation set of all manuals xxii
- Documentation, types of xix
 - machine notes xx
 - online manuals xxii
 - printed manuals xxii
- Dotted decimal format of syntax diagrams A-1
- DR_IDXAUTO configuration parameter 3-7
- DRAUTO configuration parameter 3-19
- DRDA database 2-14
- DS_HASHSIZE configuration parameter 3-9
- DS_NONPDQ_MEM configuration parameter 3-7
- DS_POOLSIZE configuration parameter 3-9
- Dynamic Server
 - 10.0
 - authentication policies 5-12, 6-10
 - backing up 5-16, 6-8
 - bringing down 5-9
 - bringing up 5-12
 - DataBlade modules, installing 5-12
 - installing 5-9
 - performance tuning 5-16

- Dynamic Server (*continued*)
 - 10.0 (*continued*)
 - release notes 3-3
 - saving configuration file 6-8
 - space requirements 5-3
 - 7.3x
 - using dbexport and dbimport 2-8
 - 9.x or later
 - moving data between computers 2-8
 - onload and onunload 2-7
 - using dbexport and dbimport 2-8
 - using onload and onunload 2-7
 - new features 3-24
 - platforms and versions 1-1
 - reverting 6-11
 - reverting from current version 6-2
- Dynamic Server 10.0
 - initializing 5-12
 - starting 5-12
- Dynamic Server 7.3x
 - moving data between computers 2-8
- Dynamic Server 9.x or later
 - importing data 2-14
 - moving data between computers 2-14
 - using dbimport 2-14
 - using dbload 2-14
- DYNAMIC_LOGS configuration parameter 3-9

E

- en_us.8859-1 locale xii
- ENCRYPT_CDR configuration parameter 3-8
- ENCRYPT_CIPHER configuration parameter 3-8
- ENCRYPT_MAC configuration parameter 3-8
- ENCRYPT_MACFILE configuration parameter 3-8
- ENCRYPT_SWITCH configuration parameter 3-8
- Enterprise Gateway Manager 2-14
- Enterprise Gateway with DRDA 2-14
- Enterprise Replication
 - enhancements 3-27, 3-29
- Environment variable
 - AFDEBUG 3-5
 - CDR_LOGDELTA 3-5
 - CDR_PERFLOG 3-5
 - CDR_RMSCALEFACT 3-5
 - CDR_ROUTER 3-5
 - CLIENT_LOCALE 8-2
 - DB_LOCALE 8-2, 8-15
 - DBCENTURY 8-1
 - DBDATE 8-1
 - DBTEMP 8-14
 - IFX_DEF_TABLE_LOCKMODE 3-5
 - IFX_EXTDIRECTIVES 3-5
 - IFX_LONGID 3-6
 - IFX_NO_TIMELIMIT_WARNING 3-5
 - IFX_ONPLOAD_AUTO_UPGRADE 3-5, 5-14

Environment variable (*continued*)

IFX_UPDDESC 3-6
INFORMIXSERVER 5-10
INFORMIXSQLHOSTS 5-10
JAR_TEMP_PATH 3-5
JAVA_COMPILER 3-5
JVM_MAX_HEAP_SIZE 3-6
ONCONFIG 5-10
PATH 5-10
reverting Dynamic Server 6-10
SERVER_LOCALE 8-2
STDIO 3-5
STMT_CACHE 3-6
TEMP 8-14
TMP 8-14
USETABLENAME 3-5

Environment variables xxiii

new in Dynamic Server 10.0 3-5
new in Dynamic Server 9.20 3-6
new in Dynamic Server 9.21 3-5
new in Dynamic Server 9.30 3-5
new in Dynamic Server 9.40 3-5

Error messages xxi

Example

calculating free space 5-5

EXT_DIRECTIVES configuration parameter 3-7

Extents, checking 5-16

Extracting schema information 7-1

F

Fast recovery

initiating 5-6

FAST_RESTART_CKPT_FUZZYLOG configuration parameter 3-7

FAST_RESTART_PHYSLOG configuration parameter 3-7

File

oncfg 5-5

onconfig 6-8

ONCONFIG 5-5, 5-11

sqlhosts 5-5, 5-11

ttermcap

saving 5-5

termcap 5-5

FILE statement

character-position form 9-10

delimiter form 9-6

syntax for

character-position form 9-11

delimiter form 9-6

with dbload 9-5

Fixed and Known Defects File xx

Formula

free space 5-4

Free space required 5-4

G

Gateway, importing from another computer 2-14

Global Language Support xii

dbimport utility 8-8

locales 8-2

new features 3-27

using onload and onunload 2-7, 13-9

GLS.

See Global Language Support.

GRANT statement

role privileges 10-8

use of dbschema 10-4

H

Help xxii

High-Availability Data Replication

disabling 5-8

disabling during reversion 6-9

enabling 5-16

High-Performance Loader

custom-code shared library 3-21

ipload utility 2-12

loading ASCII or COBOL data 2-12

onpladm utility 2-12

using 2-12

HPL_DYNAMIC_LIB_PATH configuration

parameter 3-8, 5-11

HPL.

See High-Performance Loader.

HPLAPIVERSION configuration parameter 3-8

I

IBM Informix Server Administrator 3-22

IBM Informix Storage Manager 3-35

migrating catalogs 3-36

IFX_DEF_TABLE_LOCKMODE environment variable 3-5

IFX_EXTDIRECTIVES environment variable 3-5

IFX_EXTEND_ROLE configuration parameter 3-7

IFX_LISTEN_TIMEOUT configuration parameter 3-7

IFX_LONGID environment variable 3-6

IFX_NO_TIMELIMIT_WARNING environment variable 3-5

IFX_ONPLOAD_AUTO_UPGRADE environment variable 3-5, 5-14

IFX_UPDDESC environment variable 3-6

Importing

non-Informix data 2-14

Index

checking 5-16

dbload utility 9-3

detached 3-20

rebuilding 5-13

Industry standards, compliance with xxv

Informix Dynamic Server documentation set xxii

- INFORMIXDIR directory 5-10
- INFORMIXDIR/bin directory xii
- INFORMIXSERVER environment variable 5-10
- INFORMIXSQLHOSTS environment variable 5-10

Initializing

- Dynamic Server 10.0 5-12

INSERT statement

- character-position form 9-10
- delimiter form 9-6
- syntax for
 - character-position form 9-11
 - delimiter form 9-6
- with dbload 9-5

Installation

- directory 5-10
- enhancements 3-29

Installation Guides xix

Installing

- Dynamic Server 10.0 5-9
- storage manager 3-35

Interrupt key

- with dbimport 8-9
- with dbload 9-4

ipload utility 2-12

ISO 8859-1 code set xii

J

- JAR_TEMP_PATH environment variable 3-5

- Java UDRs 6-7

- JAVA_COMPILER environment variable 3-5

- JDKVERSION configuration parameter 3-11

- JVM_MAX_HEAP_SIZE environment variable 3-6

- JVPJAVAHOME configuration parameter 3-11

- JVPJAVALIB configuration parameter 3-11

- JVPJAVAVM configuration parameter 3-11

K

- Kernel parameters on UNIX or Linux 5-9

Keywords

- in syntax diagrams xvii

L

- LBU_PRESERVE configuration parameter 3-11

- Level-0 backup 5-9, 5-16, 6-11

- after moving data 13-12

Linux

- migrating on 1-4

LOAD SQL statement

- for locales that support multibyte code sets 11-2
- syntax 11-2

Loading

- ASCII files 2-4, 2-14

- binary data 2-5, 2-7, 13-9

- COBOL data 2-4, 2-12

Locale xii

Locking

- set by onload 13-11

Logging

- onload and onunload 13-12

Logical log

- backup 5-12

- out of space 5-12

- restoring 5-16

- space required for Dynamic Server 10.0 5-3

- space required for Dynamic Server 9.40 5-4

- LOGSMAX configuration parameter 3-11

- LRU_MAX_DIRTY configuration parameter 3-10, 3-19

- LRU_MIN_DIRTY configuration parameter 3-11, 3-19

- LRUS configuration parameter 3-10, 3-19

- LTAPEBLK configuration parameter 3-10, 3-26

- LTAPEDEV configuration parameter

- onunload/onload 13-12

- LTAPESIZE configuration parameter 3-10

- LTXEHWM configuration parameter 3-11

- LTXHWM configuration parameter 3-11

M

- Machine notes xx

- MAX_INCOMPLETE_CONNECTIONS configuration parameter 3-7

Migrating

- between 32-bit and 64-bit database servers 3-36

Migrating a database

- using dbexport and dbimport 7-3

- using UNLOAD, dbschema, and LOAD 7-3

Migrating a database to

- overview 2-1

- same platforms 2-5

- Workgroup Edition 7-4

Migration

- before you begin 3-1

- checklist 3-2

- diagnostic information that you need before

- upgrading 3-2

- on Linux 1-4

- on same operating system 1-4

- on UNIX 1-4

- on Windows 2000 1-6

- on Windows NT 1-5

- on Windows XP 1-5

- onload utility 13-2, 13-5

- onunload utility 13-1

- paths 1-1, 1-7

Migration procedure

- between UNIX and Windows NT 7-2

- general guidelines for 3-3

Mode

- checking 5-8

- Monitoring conversion status 5-13

- Moving data
 - blobspaces 13-13
 - choosing a migration method 1-1
 - constraints 2-4
 - oncheck utility 13-12
 - steps for using onunload and onload 13-11

N

- Name space
 - case-sensitive 3-21
- Native Language Support (NLS)
 - populating with dbimport 8-14
- New features
 - 7.30 features in Dynamic Server 9.20 3-34
 - Java Virtual Machine 1.3 support 3-30
 - MaxConnect support in Version 9.21 3-31
 - of Dynamic Server 10.0 3-24
 - of Dynamic Server 9.20 3-31, 3-33
 - of Dynamic Server 9.21 3-30
 - of Dynamic Server 9.30 3-29
 - of Dynamic Server 9.40 3-26, 3-29
 - Universal Server 9.1x features in Dynamic Server 9.20 3-33
- NOAGE configuration parameter 3-11
- Non-Informix data, importing 2-14
- NUMAIOVPS configuration parameter 3-11
- NUMCPUVPS configuration parameter 3-11

O

- ON-Bar 5-6, 5-15
- ON-Bar utility
 - backing up
 - Dynamic Server 10.0 5-16
 - backing up Dynamic Server 5-9
 - backing up Dynamic Server 10.0 6-8
- ON-Bar Utility
 - command changes between Version 7.x and 9.x or later 3-20
- oncfg file in \$INFORMIXDIR/etc 5-5
- oncfg file in %INFORMIXDIR%/etc 5-5
- oncheck utility
 - cc database_name option 5-8, 5-16
 - cD database_name option 5-8, 5-16
 - ce option 5-8, 5-16
 - cI database_name option 5-8, 5-16
 - cr option 5-8, 5-16
 - cs sbspace_name option 5-16
 - cS sbspace_name option 5-16
 - before moving data 13-12
 - rebuilding table indexes 5-13
 - verifying database integrity 5-7, 5-15
- ONCONFIG environment variable 5-10
- onconfig file
 - saving 6-8
- ONCONFIG file 5-5, 5-11

- oninit utility
 - s option 5-7
 - starting Dynamic Server 5-13
- ONLIDX_MAXMEM configuration parameter 3-7
- Online help xxii
- Online manuals xxii
- Online notes xix, xx
- online.log 5-13
- onload and onunload utilities 2-6, 13-13
- onload utility
 - constraints on use 13-9
 - create options 13-5, 13-7
 - logging status 13-10
 - moving a database 13-11
 - moving a table 13-12, 13-13
 - moving locales 2-7, 13-9
 - moving to another dbspace 13-13
 - ownership and privileges 13-11
 - specifying tape parameters 13-6
 - steps for using 13-11
 - syntax 13-5
 - using between computers 2-5, 2-6
- onmode utility
 - b option 6-9, 12-2
 - d standard option 5-8
 - ky option 5-6
 - m option 6-11
 - sy option 5-6
 - yuk option 5-7
 - reverting from the current Dynamic Server version 6-9
 - shutting down 5-6
 - shutting down Dynamic Server 5-6
- onmode-b command 12-1
- onpladm utility 2-12
- onpload database
 - reverting 6-8
 - upgrading 5-14
- onstat utility 5-8
- ontape utility
 - a option 5-12
 - backing up
 - Dynamic Server 10.0 5-16
 - Dynamic Server source database server 5-9
 - backing up Dynamic Server 10.0 6-8
- onunload utility
 - constraints on use 13-3, 13-9
 - locking 13-5
 - logging mode 13-5
 - moving a database 13-11
 - moving a table 13-12, 13-13
 - moving locales 2-7, 13-9
 - moving to another dbspace 13-13
 - ownership and privileges 13-4
 - steps for using 13-11

- onunload utility (*continued*)
 - syntax 13-2
 - system catalog tables 13-4
 - use without tape parameters 13-2
 - using 2-5, 2-6
 - what is included with a
 - database 13-4
 - table 13-4
- Operating system
 - reconfiguring 5-9
- OPT_GOAL configuration parameter 3-9
- Optical storage manager library 3-21
- OPTICAL_LIB_PATH configuration parameter 3-11, 5-11
- Owner, in dbschema 10-4

P

- PATH environment variable 5-10
- PC_HASHSIZE configuration parameter 3-9
- PC_POOLSIZE configuration parameter 3-9
- Performance comparison 5-16
- Performance tuning 5-16
- Planning
 - data migration overview 2-1
- Platforms, moving data between
 - compatible computers 2-5, 2-8, 2-14
- PLOG_OVERFLOW_PATH configuration parameter 3-8
- Printed manuals xxii
- Privileges 10-8
 - access privileges for tables 13-4
 - required for onload 13-11
 - required for onunload 13-4

Q

- Quiescent mode 5-8

R

- Recompiling Java UDRs 6-7
- Reconfiguring the operating system 5-9
- Registering DataBlade modules 5-12
- Release notes
 - Dynamic Server 10.0 3-3
 - location of 3-3
- Release Notes xx
- Reserve pages, checking 5-16
- Reversion utility 6-9
- Reverting
 - chunk reserve pages 6-8
 - determining if it is possible 6-2
 - disabling High-Availability Data Replication 6-9
 - from Dynamic Server 10.0 6-1, 6-2, 6-11
 - from Dynamic Server 10.0 with Enterprise
 - Replication to Dynamic Server 7.31 4-3, 4-5
 - from Dynamic Server 10.0 with Enterprise
 - Replication to Dynamic Server 9.2x 4-3, 4-5

- Reverting (*continued*)
 - from Dynamic Server 10.0 with Enterprise
 - Replication to Dynamic Server 9.30 4-3, 4-4
 - removing 10.0 features 6-9
 - restrictions for 6-7
 - restrictions for reverting to prior versions 6-3
 - using the onmode -b command 12-1
- revpload.sh script 6-8
- revploadlegacy.sh script 6-8
- ROOTOFFSET configuration parameter 5-11
- ROOTPATH configuration parameter 5-11
- ROOTSIZE configuration parameter 5-11
- Running the reversion utility 6-9

S

- Sample-code conventions xviii
- SBSPACETEMP configuration parameter 3-9
- Screen reader
 - reading syntax diagrams A-1
- SELECT triggers, disabling with dbexport 8-1
- SERIAL data type, treatment by dbschema 10-4
- Server Studio JE 3-22
- SET ENVIRONMENT OPTCOMPIND 3-25
- SHMADD configuration parameter 3-19
- SHMVIRTSIZE configuration parameter 3-19
- Simple large objects
 - moving with onload 13-13
- sm_versions.std file
 - renaming to sm_versions 5-15
- Software dependencies x
- Space
 - checking what is available 5-3
 - configuring 5-3
 - for sysmaster database 5-3
- SQL
 - enhancements 3-27
- SQL code xviii
- SQL reserved words 3-12
- SQL statement
 - CREATE SEQUENCE 10-4
 - FILE 9-5
 - INSERT 9-5
 - UPDATE STATISTICS
 - data distributions 10-11
- SQLHOSTS Connectivity Information 3-23
- sqlhosts file, UNIX
 - changing name or path 5-11
 - csm option 5-12, 6-10
 - save a copy when migrating 5-5
- Starting
 - Dynamic Server 10.0 5-12
 - target database server after reversion 6-10
- STDIO environment variable 3-5
- STMT_CACHE configuration parameter 3-9
- STMT_CACHE environment variable 3-6

- STMT_CACHE_HITS configuration parameter 3-9
- STMT_CACHE_NOLIMIT configuration parameter 3-9
- STMT_CACHE_NUMPOOL configuration parameter 3-9
- STMT_CACHE_SIZE configuration parameter 3-10
- Storage manager 3-35
- Stored Procedure Parameter Limit 3-21
- stores_demo database xii
- superstores_demo database xii
- Syntax diagrams
 - conventions for xiv
 - keywords in xvii
 - reading in a screen reader A-1
 - variables in xvii
- Syntax segment xvi
- sysindexes
 - changes to 3-14
- sysmaster database 5-3
 - and logical logs 5-12
 - changes to 3-14, 3-18
 - space required to build 5-3
 - when created 7-4
- SYSSBSPACENAME configuration parameter 3-10
- System catalogs
 - boot scripts 6-1
 - changes to 3-14
 - checking tables 5-16
- System requirements
 - database x
 - software x
- sysutils database 7-4

T

- TAPEBLK configuration parameter 3-11, 3-26
- TAPEDEV configuration parameter, with onunload and onload 13-12
- TAPESIZE configuration parameter 3-11
- TBLTBLFIRST configuration parameter 3-7
- TBLTBLNEXT configuration parameter 3-7
- tctermcap file
 - saving 5-5
- termcap file in \$INFORMIXDIR/etc 5-5
- TOC Notes xx
- Tools, migration 2-4
- Transactions
 - checking for open ones 5-6
- Tuning
 - Dynamic Server 10.0 5-16
- Typographical conventions xiii

U

- ulimit 8-4
- UNIX
 - migrating on 1-4

- UNLOAD SQL statement
 - for locales that support multibyte code sets 11-2
 - moving data 7-4
 - syntax 2-9, 11-1
 - using 7-3
- UPDATE STATISTICS statement
 - data distributions 10-11
 - using after a successful conversion 5-15
 - using after reversion 6-10
- Upgrading, ISM 3-35
- User-defined data type, converting from Dynamic Server 9.2x 4-2
- Users, types of x
- USETABLENAME environment variable 3-5
- Utilities
 - dbexport 2-8, 7-1, 8-1
 - dbexport syntax 8-2
 - dbimport 2-8, 7-1, 8-1
 - dbimport syntax 8-7
 - dbload 7-1, 9-1
 - dbschema 2-10, 7-1, 10-1
 - onload 7-2
 - onload and onunload 2-6, 13-13
 - onpladm 2-12
 - onpload 2-12
 - onunload 7-2
- Utilities for data migration
 - dbexport 7-1, 8-1
 - dbimport 7-1, 8-1
 - dbload 7-1, 9-1
 - dbschema 7-1, 10-1
 - onload 7-2, 13-6
 - onunload 7-2, 13-2

V

- Validated storage manager 3-35
- VARCHAR Column Limit 3-21
- Variables, in syntax diagrams xvii
- Verifying data integrity 5-15
- Verifying storage-manager validation 3-35
- Versions
 - database servers 1-1
- View, display description with dbschema 10-3
- Visual disabilities
 - reading syntax diagrams A-1

W

- Windows 2000
 - migrating 1-6
- Windows NT
 - migrating 1-5
- Windows XP
 - migrating on 1-5
- Workgroup Edition 7.24
 - moving data between computers 2-8

Workgroup Edition 7.24 (*continued*)
 using onload and onunload 2-8

Workgroup Edition 7.3x
 moving data between computers 2-8
 using onload and onunload 2-8

Writing a dbload command file
 in character-position form 9-12
 in delimiter form 9-8



Printed in USA

G251-2293-00



Spine information:

IBM DB2 IBM Informix **Version 10.0**

IBM Informix Migration Guide

