

Informix Product Family
Informix
Version 11.70

IBM Informix Migration Guide



Informix Product Family
Informix
Version 11.70

IBM Informix Migration Guide



Note

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

Edition

This edition replaces GC27-3546-05.

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 IBM Corporation 1996, 2014.

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

Contents

Introduction	ix
About this publication	ix
Assumptions about your locale	ix
What's new in migration for Informix, Version 11.70	x
IBM Informix editions	xiii
Example code conventions	xiii
Additional documentation	xiv
Compliance with industry standards.	xiv
Syntax diagrams	xiv
How to read a command-line syntax diagram.	xv
Keywords and punctuation	xvi
Identifiers and names	xvii
How to provide documentation feedback	xvii

Part 1. Overview of migration

Chapter 1. Overview of Informix migration	1-1
The migration process	1-1
Migration effort.	1-1
Migration skills	1-1
Migration plans.	1-1
Types of migration.	1-2
Migration tools	1-3
Upgrading Informix (in-place migration)	1-3
Migrating Informix (non-in-place migration)	1-4
Hardware and operating system requirements	1-5
Fix pack naming conventions	1-6
Paths for migration to the new version	1-6
Chapter 2. Overview of moving data	2-1
Prerequisites before moving data	2-1
Data-migration tools	2-1
High-Performance Loader performance advantages for large databases	2-4
Nonlogging raw tables that speed up data loading time	2-5
When TEXT and BYTE data is scanned, not compressed	2-6
Moving non-Informix data between computers and dbspaces	2-6
Importing data from a non-Informix source	2-6
Importing data with IBM Informix Enterprise Gateway products	2-6
Moving data by using distributed SQL	2-7

Part 2. Migration to and reversion from Version 11.70

Chapter 3. Preparing for migration to Version 11.70	3-1
Preparing for migration	3-1
Reviewing changes in Informix product functionality	3-2
Checking and configuring available space	3-2
Configuring for recovery of restore point data in case an upgrade fails	3-4
Saving copies of the current configuration files	3-5
Saving a copy of the storage manager sm_versions file	3-6
Closing all transactions and shutting down the source database server	3-6
Initiating fast recovery to verify that no open transactions exist.	3-7
Verifying the integrity of the data.	3-7
Verifying that the database server is in quiescent mode	3-8
Making a final backup of the source database server	3-8

Verifying that the source database server is offline	3-8
Pre-migration checklist of diagnostic information	3-9
Migrating from 32-bit to 64-bit database servers	3-10

Chapter 4. Enterprise Replication and migration 4-1

Preparing to migrate with Enterprise Replication	4-1
Migrating with Enterprise Replication	4-2
Converting replication of 9.21 user-defined data types	4-3
Reverting with Enterprise Replication	4-3

Chapter 5. High-availability cluster migration 5-1

Preparing to migrate, upgrade, or revert clusters.	5-1
Upgrading clusters to a new PID or fix pack	5-2
Migrating clusters to a new release	5-3
Upgrading a cluster while it is online	5-4
Errors and warnings generated by the sec2er command	5-9
Reverting clusters.	5-10
Restoring clusters to a consistent point.	5-11
Restoring a cluster from a backup archive	5-12
Restoring a cluster from the HDR secondary server	5-12

Chapter 6. Migrating to Informix Version 11.70 6-1

Migrating to the new version of Informix	6-1
Installing the new version of Informix	6-2
Setting environment variables	6-4
Customizing configuration files	6-4
Adding Communications Support Modules	6-5
Installing or upgrading any DataBlade modules	6-5
Starting the new version of Informix.	6-5
Upgrading the High-Performance Loader onpload database	6-6
Restoring to a previous consistent state after a failed upgrade	6-7
Completing required post-migration tasks	6-7
For ON-Bar, copy the sm_versions file	6-8
Optionally update statistics on your tables after migrating	6-9
Update statistics on some system catalog tables after migrating.	6-9
Review client applications and registry keys	6-9
Verify the integrity of migrated data	6-10
Back up Informix after migrating to the new version	6-10
Tune the new version for performance and adjust queries	6-10
Register DataBlade modules	6-11

Chapter 7. Reverting from Informix Version 11.70 7-1

Preparing to revert.	7-1
Ascertain that reversion is possible and identify reversion requirements	7-2
Check and configure available space for reversion	7-9
Save copies of the current configuration files	7-10
Save system catalog information.	7-10
Verify the integrity of the Version 11.70 data.	7-10
Back up Informix Version 11.70	7-11
Run dummy UPDATE statements	7-11
Remove unsupported features	7-12
Remove new BladeManager extensions	7-13
Reverting from Informix Version 11.70.	7-13
Run the reversion utility	7-14
Restore original configuration parameters.	7-14
Restore original environment variables	7-14
Remove any Communications Support Module settings	7-15
Recompile any Java UDRs that were compiled using JDK 5.0	7-15
Reinstall and start the earlier database server	7-15
Optionally update statistics on your tables after reverting	7-15

Update statistics on some system catalog tables after reverting	7-15
Verify the integrity of the reverted data	7-16
Back up the database server after reversion	7-16
Return the database server to online mode	7-16
Register DataBlade modules	7-16

Part 3. Migration of data between database servers

Chapter 8. Migrating database servers to a new operating system 8-1

Choosing a tool for moving data before migrating between operating systems	8-1
Adjusting database tables for file-system variations	8-2
Moving data to a database server on a different operating system	8-2
Moving data between Informix and Workgroup Edition Version 7.24 on different operating systems	8-2
Adapting your programs for a different operating system.	8-3
Ensuring the successful creation of system databases	8-3

Part 4. Data migration utilities

Chapter 9. The dbexport and dbimport utilities 9-1

Syntax of the dbexport command	9-3
Termination of the dbexport utility	9-5
dbexport errors	9-5
dbexport server-specific information	9-5
dbexport destination options	9-6
Contents of the schema file that dbexport creates	9-7
Syntax of the dbimport command	9-8
Termination of the dbimport utility	9-9
dbimport errors and warnings	9-9
dbimport input-file location options	9-10
dbimport create options.	9-11
Database-logging mode.	9-13
Database renaming	9-13
Changing the database locale with dbimport	9-14
Simple large objects	9-14

Chapter 10. The dbload utility 10-1

Syntax of the dbload command	10-1
Table locking during a load operation	10-3
Rows to ignore during a load operation	10-4
Bad-row limit during a load operation.	10-4
Termination of the dbload utility	10-4
Name and object guidelines for the dbload utility	10-4
Command file for the dbload utility	10-5
Delimiter form of the FILE and INSERT statements	10-5
Character-position form of the FILE and INSERT statements	10-9
Command file to load complex data types	10-13
Using the dbload utility with named row types	10-13
Using the dbload utility with unnamed row types	10-14
Using the dbload utility with collection data types	10-14

Chapter 11. The dbschema utility 11-1

Object modes and violation detection in dbschema output	11-1
Guidelines for using the dbschema utility.	11-2
Syntax of the dbschema command	11-2
Database schema creation	11-5
dbschema server-specific information	11-6
User-defined and complex data types	11-6
Sequence creation.	11-7
Synonym creation.	11-7

Table, view, or procedure creation	11-8
Table information	11-9
Storage space, chunk, and log creation	11-9
Role creation	11-11
Privileges	11-12
Distribution information for tables in dbschema output	11-13
Use dbschema output as DB-Access input	11-15
Inserting a table into a dbschema output file	11-15
Re-creating the schema of a database	11-16

Chapter 12. The LOAD and UNLOAD statements 12-1

Syntax of the UNLOAD statement	12-1
Syntax of the LOAD statement	12-2
Load and unload statements for locales that support multibyte code sets	12-2
Load and unload operations for nondefault locales and GL_DATETIME or DBTIME environment variables	12-2

Chapter 13. The onunload and onload utilities 13-1

Guidelines for when to use the onunload and onload utilities	13-1
Requirements for using the onload and onunload utilities	13-2
How the onunload and onload utilities work	13-3
Syntax of the onunload command	13-3
onunload destination parameters	13-4
Constraints that affect onunload	13-5
Privileges for database or table unloading	13-5
Tables that are unloaded with a database	13-5
Data that is unloaded with a table	13-5
Locking during unload operation	13-5
Logging mode	13-6
Syntax of the onload command	13-6
onload source parameters	13-7
onload create options	13-8
Constraints that affect onload	13-9
Logging during loading	13-10
Movement of simple large objects to a blob space	13-10
Ownership and privileges	13-10
Exclusive locking during a load operation	13-10
Moving a database between computers with the onunload and onload utilities	13-11
Moving a table between computers with the onunload and onload utilities	13-11
Moving a table between dbspaces with the onunload and onload utilities	13-12

Chapter 14. The onmode utility reversion option 14-1

What the onmode -b command does	14-1
Preparation for using the onmode -b command	14-1
Syntax of the onmode -b command	14-1

Chapter 15. The onrestorept utility 15-1

Syntax of the onrestorept command	15-1
---	------

Part 5. New and changed features in Informix servers

Chapter 16. Environment variable changes by version 16-1

Chapter 17. Configuration parameter changes by version 17-1

Configuration parameter changes in Version 11.70	17-9
Configuration parameter changes in Version 11.50	17-11
Configuration parameter changes in Versions 10.00 through 11.10	17-16

Chapter 18. SQL keyword changes by version	18-1
---	-------------

Chapter 19. System catalog and system database changes by version	19-1
--	-------------

Changes for version 11.70	19-1
Changes for version 11.50	19-1
Changes for version 11.10	19-2
Changes for version 10.00	19-3
Changes for version 9.40	19-3
Changes for version 9.30	19-4
Column-width changes in sysmaster tables in Version 9.20 and later versions.	19-5
Data type changes in sysmaster tables in Version 9.20 and later versions	19-8
Changes in treatment of null values in sysmaster tables in Version 9.30.	19-8
Other sysmaster database table and column changes in Version 9.30.	19-9
Remote queries on system catalog tables between Version 7.31 and later versions	19-9
Difference between Version 7.31 and later versions	19-10

Chapter 20. Server library name changes by version	20-1
---	-------------

Part 6. Appendixes

Appendix. Accessibility	A-1
--	------------

Accessibility features for IBM Informix products	A-1
Accessibility features	A-1
Keyboard navigation	A-1
Related accessibility information	A-1
IBM and accessibility.	A-1
Dotted decimal syntax diagrams	A-1

Notices	B-1
--------------------------	------------

Privacy policy considerations	B-3
Trademarks	B-3

Index	X-1
------------------------	------------

Introduction

About this publication

This publication describes how to migrate to a new version of IBM® Informix®, how to revert to the database server that you migrated from, and how to move data manually between databases, servers, and computers.

This publication includes information about how to use:

- The **dbexport**, **dbimport**, **dbload**, **dbschema**, **onload**, and **onunload** data-migration utilities
- The LOAD and UNLOAD SQL statements.
- The **onrestorept** utility, which you can use to restore data if an upgrade fails

This publication does not contain information about using the DSN (data source name) Migration Utility (**dsnigrate.exe**) to migrate from one version of CSDK to another. For information about migrating DSN, see the "DSN Migration Utility" section in the *IBM Informix ODBC Driver Programmer's Manual*.

For information about migrating to previous versions of IBM Informix database servers, see the *Migration Guide* 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.

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 that is used by a language within a given territory and encoding is brought together in a single environment, called a Global Language Support (GLS) locale.

The IBM Informix OLE DB Provider follows the ISO string formats for date, time, and money, as defined by the Microsoft OLE DB standards. You can override that default by setting an Informix environment variable or registry entry, such as **DBDATE**.

If you use Simple Network Management Protocol (SNMP) in your Informix environment, note that the protocols (SNMPv1 and SNMPv2) recognize only English code sets. For more information, see the topic about GLS and SNMP in the *IBM Informix SNMP Subagent Guide*.

The examples in this publication are written with the assumption that you are using one of these locales: en_us.8859-1 (ISO 8859-1) on UNIX platforms or en_us.1252 (Microsoft 1252) in Windows environments. These locales support U.S. English format conventions for displaying and entering date, time, number, and

currency values. They also support the ISO 8859-1 code set (on UNIX and Linux) or the Microsoft 1252 code set (on Windows), which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

You can specify another locale if you plan to use characters from other locales in your data or your SQL identifiers, or if you want to conform to other collation rules for character data.

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

What's new in migration for Informix, Version 11.70

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

The following changes and enhancements are relevant to this publication. For a complete list of what's new in this release, see the release notes or the information center at http://pic.dhe.ibm.com/infocenter/idshelp/v117/topic/com.ibm.po.doc/new_features.htm.

Table 1. What's new for migration in Version 11.70.xC8

Overview	Reference
<p>Server changes</p> <p>Informix 11.70.xC8 includes new and changed configuration parameters and environment variables.</p>	<p>Part 5, "New and changed features in Informix servers"</p>
<p>Temporarily prevent constraint validation</p> <p>You can significantly increase the speed of loading or migrating large tables by temporarily preventing the database server from validating foreign-key referential constraints. You can disable the validation of constraints when you create constraints or change the mode of constraints to ENABLED or FILTERING.</p> <p>When you migrate data, include the -nv option in the dbimport command.</p> <p>The NOVALIDATE keyword prevents the database server from checking every row for referential integrity during ALTER TABLE ADD CONSTRAINT and SET CONSTRAINTS operations on foreign-key constraints. When those statements finish running, the database server automatically resumes referential-integrity enforcement of those constraints in subsequent DML operations.</p> <p>Use this feature only on tables whose enabled foreign-key constraints are free of violations, or when the referential constraints can be validated after the tables are loaded or migrated to the target database.</p>	<p>"Syntax of the dbimport command" on page 9-8</p>

Table 2. What's new for migration in Version 11.70.xC3

Overview	Reference
<p>Converting the Connection Manager configuration file</p> <p>The Connection Manager configuration file in versions of IBM Informix Client Software Development Kit (Client SDK) prior to version 3.70.xC3 are incompatible with the current version of the Connection Manager. You must convert configuration files from versions prior to 3.70.xC3. See Converting older formats of the Connection Manager configuration file to the current format (Administrator's Guide) for more information.</p>	<p>Converting older formats of the Connection Manager configuration file to the current format (Administrator's Guide)</p>
<p>Built-in SQL compatibility functions for string manipulation and trigonometric support</p> <p>The Informix database server supports new built-in SQL string manipulation functions. These functions return either a character string derived from an argument to the function, or an integer that describes a string argument:</p> <ul style="list-style-type: none"> • CHARINDEX() • INSTR() • LEFT() • LEN() • REVERSE() • RIGHT() • SPACE() • SUBSTRING_INDEX() <p>This release also provides two built-in trigonometric support functions. These functions convert the units of angular measurement of a numeric expression argument from radians into degrees, or from degrees into radians:</p> <ul style="list-style-type: none"> • DEGREES() • RADIANS() 	<p>"Preparing for migration" on page 3-1</p>

Table 3. What's new for migration in Version 11.70.xC2

Overview	Reference
<p>Case-insensitive queries on NCHAR and NVARCHAR text strings</p> <p>In previous IBM Informix releases, strings stored in all Informix databases were treated as case-sensitive by database operations. For example, a query for the string "McDavid" returns "McDavid" but not "mcdavid", "MCDAVID", or "Mcdavid". Operations designed to disregard the case of text strings require a bts index or a functional index for each query.</p> <p>In this release a database is still created as case-sensitive by default. However, you can use the NLSCASE INSENSITIVE option with the CREATE DATABASE statement to create a database that ignores the case of text strings. For example, querying "McDavid" returns "McDavid", "mcdavid", "MCDAVID", and "Mcdavid".</p> <p>A case-insensitive database ignores letter case only on NCHAR and NVARCHAR data types, but it treats the other built-in character data types (CHAR, LVARCHAR, and VARCHAR) as case-sensitive. You cannot include both case-sensitive and case-insensitive databases in a distributed query.</p>	<p>Chapter 9, "The dbexport and dbimport utilities," on page 9-1</p>

Table 4. What's new for migration in Version 11.70.xC1

Overview	Reference
<p>Upgrading to Version 11.70</p> <p>If you are migrating from Informix Version 11.50 11.10, 10.0, 9.40, or 7.31, you can migrate directly to Informix Version 11.70.</p>	<p>"Paths for migration to the new version" on page 1-6</p>
<p>Generating storage spaces and logs with the dbschema utility</p> <p>The dbschema utility can now generate the schema of storage spaces, chunks, and physical and logical logs. You can choose to generate either SQL administration API commands or onspaces and onparams utility commands that you can run to reproduce the storage spaces, chunks, and logs. For example, you might want to generate the schema before you use the dbexport and dbimport utilities to move data or before you drop a space that you might want to regenerate at a later date.</p>	<p>"Syntax of the dbschema command" on page 11-2</p> <p>"Storage space, chunk, and log creation" on page 11-9</p>
<p>dbschema utility enhancement for omitting the specification of an owner</p> <p>You can use the new dbschema utility -nw option to generate the SQL for creating an object without specifying an owner.</p>	<p>"Syntax of the dbschema command" on page 11-2</p>
<p>dbexport utility enhancement for omitting the specification of an owner</p> <p>You can use the new dbexport utility -nw option to generate the SQL for creating a database without specifying an owner.</p>	<p>"Syntax of the dbexport command" on page 9-3</p>

Table 4. What's new for migration in Version 11.70.xC1 (continued)

Overview	Reference
<p>Upgrading a high-availability cluster while it is on line</p> <p>You can now upgrade the Informix software on a high-availability cluster without incurring any down time.</p>	<p>“Upgrading a cluster while it is online” on page 5-4</p>
<p>New editions and product names</p> <p>IBM Informix Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products:</p> <ul style="list-style-type: none"> • IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS) • IBM OpenAdmin Tool (OAT) for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS) • IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature 	<p>For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.</p>
<p>New environment variable enables invalid character data to be used by DB-Access, dbexport, and High Performance Loader</p> <p>The IFX_UNLOAD_EILSEQ_MODE environment variable enables DB-Access, dbexport, and High Performance Loader (HPL) to retrieve character data that is invalid for the locale specified in the environment.</p>	<p>“Syntax of the dbexport command” on page 9-3</p>

IBM Informix editions

IBM Informix editions are designed to fit various business needs.

Some of the capability that is described in IBM Informix documentation might not be available for all editions.

The license agreement has the specific restrictions for each edition. To view a license for a particular edition, search for IBM Informix on the following website: <http://www.ibm.com/software/sla/sladb.nsf>.

Example code conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
  WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

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

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

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

Additional documentation

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

You can access Informix technical information such as information centers, technotes, white papers, and IBM Redbooks® publications online at <http://www.ibm.com/software/data/sw-library/>.

Compliance with industry standards

IBM Informix products are compliant with various standards.

IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

Syntax diagrams

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

Table 5. 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.

Table 5. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning
<p>A horizontal line with three vertical bars below it. The bars are labeled ALL, DISTINCT, and UNIQUE from top to bottom.</p>	<pre>---+----ALL-----+--- +--DISTINCT-----+ '---UNIQUE-----'</pre>	Required item with choice. Only one item must be present.
<p>A horizontal line with two vertical bars below it. The bars are labeled FOR UPDATE and FOR READ ONLY from top to bottom.</p>	<pre>---+-----+--- +--FOR UPDATE-----+ '--FOR READ ONLY--'</pre>	Optional items with choice are shown below the main line, one of which you might specify.
<p>A horizontal line with three vertical bars below it. The bars are labeled NEXT, PRIOR, and PREVIOUS from top to bottom.</p>	<pre>.---NEXT----- ---+-----+--- +--PRIOR-----+ '---PREVIOUS-----'</pre>	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 is used by default.
<p>A horizontal line with a comma above it. Below the line are two vertical bars labeled index_name and table_name.</p>	<pre>-----,----- v----- ---+-----+--- +---index_name---+ '---table_name---'</pre>	Optional items. Several items are allowed; a comma must precede each repetition.
<p>A horizontal line with a double arrow pointing right above it and a double arrow pointing left below it. The text 'Table Reference' is in the center.</p>	<pre>>>- Table Reference -<<</pre>	Reference to a syntax segment.
<p>Table Reference</p> <p>A horizontal line with three vertical bars below it. The bars are labeled view, table, and synonym from top to bottom.</p>	<pre> ---+----view-----+-- +-----table-----+ '-----synonym-----'</pre>	Syntax segment.

How to read a command-line syntax diagram

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

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

Creating a no-conversion job

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

```
▶▶ -t table ▶▶
```

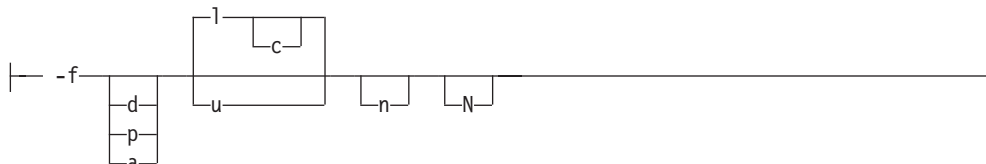
```
▶▶ [ -S server ] [ -T target ] Setting the Run Mode (1) ▶▶
```

Notes:

- 1 See page Z-1

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

Setting the run mode:



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

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

1. Include **onpladm create job** and then the name of the job.
2. Optionally, include **-p** and then the name of the project.
3. Include 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 include 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 include **-f**, optionally include **d**, **p**, or **a**, and then optionally include **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and punctuation

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

A keyword in a syntax diagram 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 other syntax diagrams. A variable in a syntax diagram, an example, or text, is shown in *lowercase italic*.

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

►—SELECT—*column_name*—FROM—*table_name*—◄

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

How to provide documentation feedback

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

Use one of the following methods:

- Send email to docinf@us.ibm.com.
- In the Informix information center, which is available online at <http://www.ibm.com/software/data/sw-library/>, open the topic that you want to comment on. Click the feedback link at the bottom of the page, complete the form, and submit your feedback.
- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support at <http://www.ibm.com/planetwide/>.

We appreciate your suggestions.

Part 1. Overview of migration

Chapter 1. Overview of Informix migration

Before you upgrade to the new version of Informix, ensure that you understand the migration process, prerequisites, and reversion options.

The migration process

This overview of the migration process describes what you need to know to plan your migration and the resources that you can use to assist you.

Careful planning will ensure minimal impact on your business.

- “Migration effort”
- “Migration skills”
- “Migration plans”
- “Types of migration” on page 1-2
- “Migration tools” on page 1-3

Migration effort

Depending on your environment, the migration process can take a few hours or several weeks.

The migration effort is determined by many factors:

- Your current version of Informix. The older the version, the greater the effort.
- The site architecture and configuration, before and after migration.
- The level of site customization, before and after migration.
- Integration of additional software products.
- To some extent, the size of the database.

Migration skills

Your Informix migration team needs database administration skills, system administration skills, and application programming skills.

The migration team needs:

- Database administration skills, to help migrate custom database extensions.
- System administration skills, to perform various system tasks. These tasks include operating system installation, configuration and maintenance and the installation and configuration of Informix and any additional software products.
- Application programming skills, to create and maintain scripts to evaluate and modify application programs.

If you prefer, highly-skilled IBM Services personnel and business partners are available to assist you in migrating your environment. Contact your IBM representative for further information.

Migration plans

Before you begin to migrate to a new version of the database server, you should plan for migration.

To plan your migration requirements, complete these tasks:

1. Inventory the existing Informix environment assets, such as machines, instances, databases, database customization, custom code, IBM software, and third-party software.
2. Itemize the requirements for the post-migrated environment. New requirements can include upgrading or adding hardware, using new features, or replacing custom-code with new built-in function.
3. Plan the migration activities. Typical activities include:
 - Performing a level-0 backup of the database.
 - Quiescing the database server and preventing connections to the database until migration completes.
Important: Any connection attempts (for example, from cron jobs or monitoring scripts) to the database after you quiesce the database server and during migration will cause migration to fail.
 - Installing the new version of Informix.
 - Migrating database data.
 - Reverting to the previous version.
 - Migrating applications before using them with the new database server.

Depending on your environment, you might need to perform some of these activities more than once. You might not need to restore the level-0 backup; however, if you encounter problems you can always restore the backup of your current server.

Types of migration

There are three ways to migrate to Informix Version 11.70.

Upgrading (In-place migration)

Upgrading is a special case of migration that uses your existing hardware and operating system. You install a new or improved version of the product in a different location from your current version on the same machine. You can copy your configuration file and add new parameters. When you start the new Informix instance, the database data is automatically converted. For example, you can upgrade from Version 11.10 to Version 11.70.

Migrating (Non-in-place migration)

The process of “switching over” your environment from one computer to another. This type of migration requires more planning and setup time compared to upgrading on your existing computer. Non-in-place migration requires that you modify and copy the database schema, user data, and user objects from one server to another server. Use this type of migration if you are moving to Informix Version 11.70 from an early version of Informix that has a different architecture, page size, optimization of dbspaces, and extent allocations.

Migrating from a non-IBM database

The process of moving your data from another database management system (DBMS) such as Oracle or Sybase SQL Anywhere to Informix Version 11.70. This type of migration is especially useful if you are currently using various products. You can consolidate to take advantage of the Informix features and total cost of ownership.

If you have a high-availability cluster with one or more secondary database servers or if you use Enterprise Replication, you follow additional procedures to upgrade your servers.

Related concepts:

Chapter 4, “Enterprise Replication and migration,” on page 4-1

Chapter 5, “High-availability cluster migration,” on page 5-1

Chapter 3, “Preparing for migration to Version 11.70,” on page 3-1

Migration tools

You can choose from various migration tools, depending on the task that you must perform.

- For in-place upgrades: You do not use any migration tools. Simply start the server by using the **oninit** utility. The data from the source database server is converted to the target database server automatically.
- For non-in-place migration: You can use distributed queries to move your data, or you can pick from a number of data transfer tools and utilities, such as:
 - **dbexport** and **dbimport**
 - **onload** and **onunload**
 - **dbload**
 - High Performance Loader (HPL)
 - External tables

Each of these tools and utilities has specific advantages and limitations. Consider all of the variables and pick a tool or utility that is appropriate for your situation.

You can use several different methods to move your data from non-IBM products to Informix.

Related reference:

“Data-migration tools” on page 2-1

Upgrading Informix (in-place migration)

In-place migration upgrades Informix directly to the current version by installing the product in a new directory, copying a few configuration files, and starting the new server to automatically convert your database data.

Upgrading is an in-place migration method that uses your existing test and production hardware. The operating system on those machines must be supported by the new version of the server. Also, you must have enough space for the system database data conversion.

Upgrading consists of these steps:

1. Prepare your system. That includes closing all transactions, verifying the integrity of the data with the **oncheck** utility, and performing a level-0 backup. (If you are using Enterprise Replication or high-availability clusters, you must stop replication and perform required additional tasks.)

While removing in-place alters is not required before upgrading, a best practice is to remove all in-place alters before upgrading. (For information about manually identifying outstanding In-Place ALTER statements using the **oncheck** utility, see this technical document: <http://www.ibm.com/support/docview.wss?uid=swg21144602>.)

2. Install the new product on the machine.

Important: The safest way to upgrade to a new version is to install the new version in another directory. To save space, you can select the product components that you want to install. After you test the database server instance with the similar configuration settings and the client connection information that you use for your current database server, you can remove the old version.

If you do not have space on your machine for two versions, install the new version over the existing version. In this case, you cannot selectively install components; you must install the whole product to block objects from the previous version. Before you choose this approach, make sure that you have the original installation media for the old version, because you will not be able to automatically revert to it.

3. Copy the ONCONFIG file to the target and set parameters that are new for the current release.
4. Start an instance of the new version of Informix. The database data is automatically converted.
5. If you have performance problems run UPDATE STATISTICS and UPDATE STATISTICS FOR PROCEDURE.

This type of migration minimizes the risk of introducing errors. You can always revert from the new server to the old one. In the event of a problem during reversion, you can restore the level-0 backup.

Related concepts:

“Migrating Informix (non-in-place migration)”

“Hardware and operating system requirements” on page 1-5

Chapter 3, “Preparing for migration to Version 11.70,” on page 3-1

Chapter 4, “Enterprise Replication and migration,” on page 4-1

Chapter 5, “High-availability cluster migration,” on page 5-1

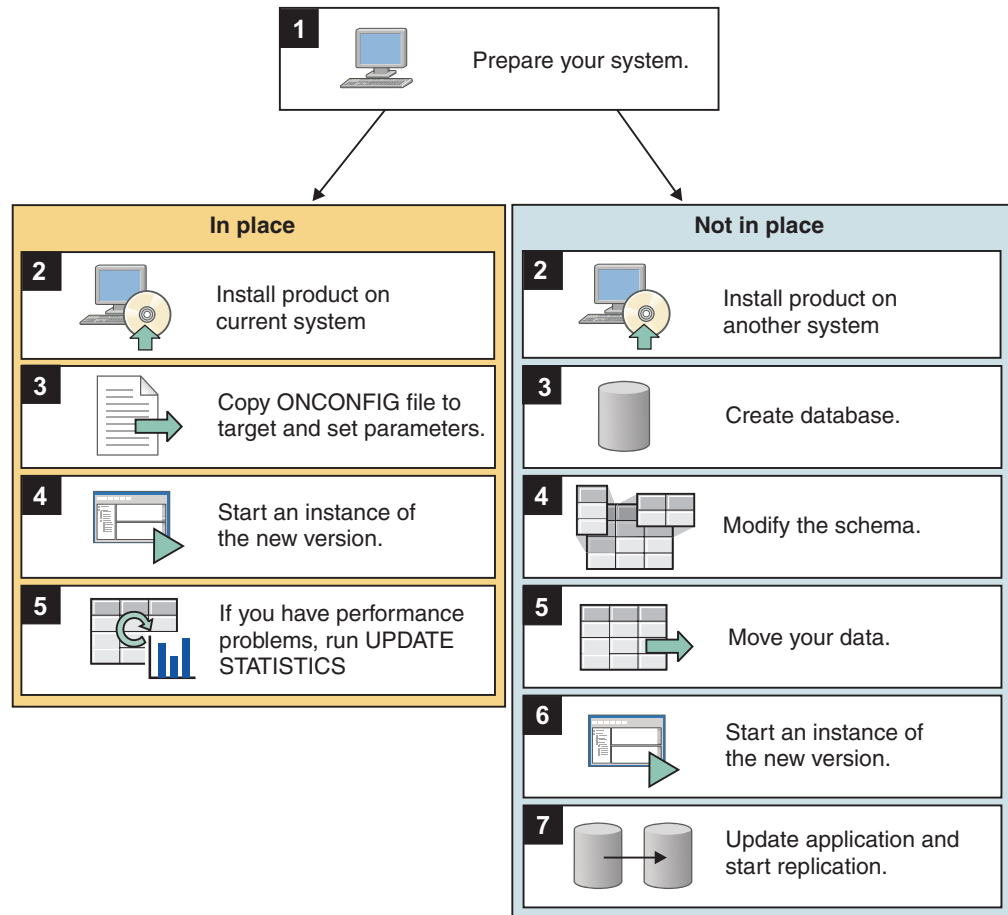
Migrating Informix (non-in-place migration)

Depending on your existing database server setup, you might need to move to a new architecture or a different server. This type of migration is more complicated than in-place migration. It requires more planning and it is more time consuming.

The non-in-place type of migration consists of these steps:

1. Prepare your system. That includes closing all transactions, verifying the integrity of the data with **oncheck**, and performing a level-0 backup. If you are using Enterprise Replication or High-Availability Data Replication, stop replication.
2. Install the new product on a new machine.
3. Create a database with the current schema (**dbschema -d dbname -ss**).
4. Modify the schema for new extent allocations and lock mode changes. If applicable, modify schema for new dbspaces.
5. Move data by using the appropriate utility or tool, or by using distributed queries.
6. Start the Informix Version 11.70 instance.
7. After Informix migration, upgrade applications before running them. (Also, if you use Enterprise Replication or high-availability clusters, you must perform additional tasks.)

The following illustration shows the differences between in-place and non-in-place migration.



Related concepts:

“Upgrading Informix (in-place migration)” on page 1-3

“Hardware and operating system requirements”

Chapter 3, “Preparing for migration to Version 11.70,” on page 3-1

Chapter 4, “Enterprise Replication and migration,” on page 4-1

Chapter 5, “High-availability cluster migration,” on page 5-1

Related reference:

“Data-migration tools” on page 2-1

Hardware and operating system requirements

Ensure that you meet the operating system and hardware requirements for Informix Version 11.70.

Before you upgrade or migrate to the current version of Informix, ensure that the system you choose meets the necessary operating system, hardware, disk, and memory requirements.

For a complete list of supported operating systems and hardware prerequisites for Informix, go to <http://www.ibm.com/support/docview.wss?uid=swg27013343>.

If you are running an earlier version of Informix on an operating system that is no longer supported, you must migrate to a supported operating system.

Your hardware must support the operating systems that Informix Version 11.70 supports, and it must provide enough disk space for the database server and all your other software applications.

Check the machine notes for information about the operating-system patches that you need for successful installation and operation of the database server. Follow any platform-specific instructions in the machine notes.

UNIX, Linux: You might have to change some of the kernel parameters for your UNIX or Linux operating system before you install Informix Version 11.70. Refer to the kernel-configuration instructions for your operating system.

Related concepts:

“Upgrading Informix (in-place migration)” on page 1-3

“Migrating Informix (non-in-place migration)” on page 1-4

Related tasks:

“Preparing for migration” on page 3-1

Fix pack naming conventions

Informix releases and fix packs contain version names that appear in the format `aa.bb.xCn`.

In this format:

- aa = major release number
- bb = minor release number
- x = all operating system platforms, unless one of the following characters appears in the position of x:
 - F = 64-bit on any UNIX, Linux, or Windows platform
 - H = 32-bit build on any HP 11.x platform; also runs on HP 11.x 64-bit
 - J = Java™
 - T = 32-bit on Windows platforms
 - U = 32-bit on any UNIX or Linux platform
- C = GA release
- n = fix pack level

For example, in Version 11.50.xC6, 11 is the major release number, 50 is the minor release number, x means any platform, C means GA release, and 6 means fix pack 6.

Paths for migration to the new version

Direct migration to Informix 11.70 is supported for specific earlier versions of the database server.

Important: As new versions of products are released, some earlier versions of the products are no longer supported. For information about the kind of support available for your product version, go to the *Information Management Product Lifecycle* page at <http://www.ibm.com/software/data/support/lifecycle/>.

Migration paths

If you are migrating from Version 11.50, 11.10, 10.0, 9.40, or 7.31, you can migrate directly to Version 11.70 on the same operating system.

If you are on earlier versions of the database server, consider these options:

- Migrate from Version 9.30 and Version 9.21 to Version 11.50 of the database server before you migrate to Version 11.70. Refer to the migration information that is included in the documentation set for the interim version of the database server.
- Install Version 11.70 in a new location, and then use the **dbexport** and **dbimport** utilities or distributed SQL to move your data into the new database server.

If necessary, you can revert to the version of the database server from which you upgraded. You cannot revert to any other version of the database server.

Related concepts:

Chapter 9, “The dbexport and dbimport utilities,” on page 9-1

“Moving data by using distributed SQL” on page 2-7

Chapter 8, “Migrating database servers to a new operating system,” on page 8-1

Chapter 2. Overview of moving data

If you are installing the new version of the database server on another computer or operating system (non-in-place migration), you can use one of several tools and utilities to move data from your current database server.

For example, suppose you migrated to the current version of Informix and created a few new databases, but decide to revert to the previous version. Before you revert, you can use one of the data-migration tools to save the data you added. After reverting, you can reload the data.

Before you move data, consider these issues:

- Changes in the configuration parameters and environment variables
- Amount of memory and dbospace space that is required
- Organization of the data
- Whether you 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 between database servers on different operating systems, also see Chapter 8, “Migrating database servers to a new operating system,” on page 8-1.

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

Prerequisites before moving data

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*.

Data-migration tools

Informix provides tools, utilities, and SQL statements that you can use to move data from one IBM Informix database to another or from one operating system to another.

You might want to use a data-migration tool when you have different page sizes or code pages. For example, UNIX or Linux and Windows store data in different page sizes.

When your migration involves migrating between 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.

Normally, if you are migrating on the same operating system, you do not need to load and unload data.

You can use the following tools to move data:

- The **dbexport** and **dbimport** utilities
- The **dbload** utility
- The **onunload** and **onload** utilities
- UNLOAD and LOAD statements
- The High-Performance Loader (HPL)
- Nonlogging raw tables

When you import data from non-Informix sources, you can use the following tools:

- The **dbimport** and **dbload** utilities
- The High-Performance Loader (HPL)
- IBM Informix Enterprise Gateway products
- External tables that you create with the CREATE EXTERNAL TABLE statement

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 following table summarizes the characteristics of the methods for loading data and the advantages and disadvantages of each method. The table also shows the database servers on which you can use the tools.

Table 2-1. Comparison of tools for moving data

Tool	Description	Advantages	Disadvantages
dbexport and dbimport utility	Imports or exports a database to a text file that is stored on disk or tape	Can modify the database schema and change the data format Can move data between operating systems Optional logging Can import data from non-Informix sources	Faster performance than the dbload utility, but slower performance than the onload utility Moves the entire database
dbload utility	Transfers data from one or more text files into one or more existing tables	Can modify database schema Can move data between operating systems Optional logging Moderately easy to use Can import data from non-Informix sources	Slower performance than the dbexport , dbimport , and onload utilities
onunload and onload utilities	Unloads data from a database into a file on tape or disk; loads data, which was created with the onunload command, into the database server	Fast performance Optional logging	Only moves data between database servers of the same version on the same operating system Cannot modify the database schema Logging must be turned off Difficult to use

Table 2-1. Comparison of tools for moving data (continued)

Tool	Description	Advantages	Disadvantages
UNLOAD and LOAD statements	Unloads and loads specified rows	<ul style="list-style-type: none"> Can modify database schema Can move data between operating systems Easy to use Optional logging 	Only accepts specified data formats
HPL	Loads data from any ASCII or COBOL file that meets certain format requirements	<ul style="list-style-type: none"> For extremely large databases, has a performance advantage over other IBM Informix data-migration utilities, because it performs I/O and code-set conversions in parallel Can modify database schema Can move data between operating systems Can import data from non-Informix sources 	Requires significant preparation time
Nonlogging raw tables	Loads certain kinds of large tables	Can load very large data warehousing tables quickly	<ul style="list-style-type: none"> Does not support primary constraints, unique constraints, and rollback Requires SQL Not recommended for use within a transaction
External tables	Enables you to read and write from a source that is external to the database server, providing an SQL interface to data in text files managed by the operating system or to data from a FIFO device.	Performs express (high-speed) and deluxe (data-checking) transfers	Requires SQL

If you are choosing a tool for loading data, the questions shown in Figure 2-1 on page 2-4 will help you make a decision.

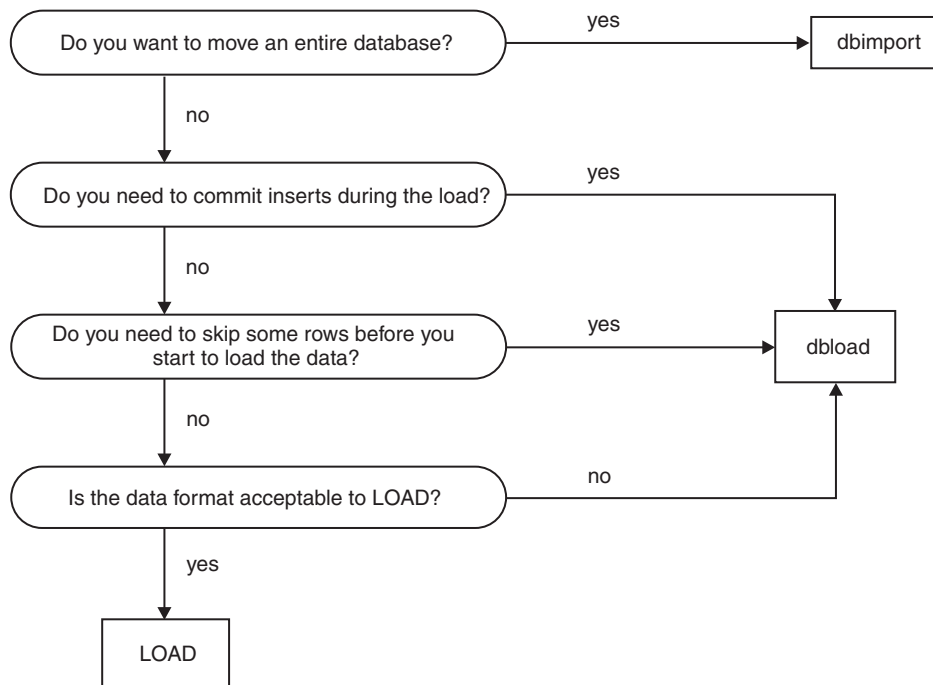


Figure 2-1. Choosing among dbimport, dbload, and LOAD

In addition to the tools that move data, you can use the **dbschema** utility, which gets the schema of a database and redirects the output to a file, so you can provide the file to DB–Access to re-create the database.

Related concepts:

- “Migrating Informix (non-in-place migration)” on page 1-4
- “Migration tools” on page 1-3
- Chapter 9, “The dbexport and dbimport utilities,” on page 9-1
- Chapter 13, “The onunload and onload utilities,” on page 13-1
- Chapter 10, “The dbload utility,” on page 10-1
- Chapter 11, “The dbschema utility,” on page 11-1
- Chapter 12, “The LOAD and UNLOAD statements,” on page 12-1
- “High-Performance Loader performance advantages for large databases”

Related tasks:

- [Moving data with external tables \(Administrator's Guide\)](#)

Related reference:

- [CREATE EXTERNAL TABLE Statement \(SQL Syntax\)](#)

High-Performance Loader performance advantages for large databases

The High-Performance Loader (HPL) utility, which can load data from any ASCII or COBOL file that meets certain format prerequisites, uses parallel processing to perform fast data loading and unloading. However, the HPL requires significant preparation time.

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. 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.

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

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

Related concepts:

“Choosing a tool for moving data before migrating between operating systems” on page 8-1

Related tasks:

 Moving data with external tables (Administrator's Guide)

Related reference:

“Data-migration tools” on page 2-1

 CREATE EXTERNAL TABLE Statement (SQL Syntax)

Nonlogging raw tables that speed up data loading time

If you use a database server utility to load data, you can use nonlogging raw tables in a logging database to speed up the initial loading and validation of data if you are moving data to or from Version 9.21 or any later version.

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.

In a logged database, Informix creates standard tables that use logging by default.

To create a nonlogging table, use the CREATE RAW TABLE statement, or use the ALTER TABLE statement to change the table type from STANDARD to RAW. After the loading of a raw table is complete, you can change the table 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 about 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 about 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 about the ALTER TABLE statement, see the *IBM Informix Guide to SQL: Syntax*.

Informix also supports external tables, which provide an SQL interface to data in text files managed by the operating system or to data from a FIFO device. To create external tables, use the CREATE EXTERNAL TABLE statement. Use the DROP TABLE statement to drop an external table.

Related tasks:

 [Moving data with external tables \(Administrator's Guide\)](#)

Related reference:

 [CREATE EXTERNAL TABLE Statement \(SQL Syntax\)](#)

When TEXT and BYTE data is scanned, not compressed

An Informix database server scans TEXT and BYTE data into an existing table when you load data by using the SQL LOAD statement, the **dbload** utility, the Informix ESQL/C program, the HPL, or external tables.

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

Moving non-Informix data between computers and dbspaces

You can move data between different computers, and you can import data from non-Informix environments. Except when you use the High-Performance Loader (HPL) utility 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 you can specify the delimiter that is used in the data files.

Importing data from a non-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.

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

You can use IBM Informix Enterprise Gateway with DRDA[®] to query a DRDA database and then insert the results into an Informix database.

For example, to import data, run a SELECT statement to select data from the non-Informix database and then run an INSERT statement to insert data into the Informix database. For more information, refer to the *IBM Informix Enterprise Gateway with DRDA User Manual*.

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. For more information, refer to the *IBM Informix Enterprise Gateway with DRDA User Manual*.

Moving data by using distributed SQL

If you want to move data with different binary pages and page sizes across platforms and you have expertise in using distributed SQL, you can use INSERT and SELECT SQL statements to transfer the data.

Important: Do not use INSERT and SELECT statements to move data if the database contains BLOB data types.

Prerequisites: A network connection must exist between database server instances.

To move data using INSERT and SELECT statements with fully qualified table names:

1. Capture the complete database schema from the source database server.
2. Alter the extent sizing and, if necessary, the lock modes on tables from page to row.
3. Create and verify the schema on the target database server.
4. Disable logging on both source and target servers where necessary.
5. Create and run the following scripts:
 - a. Create and run separate scripts for:
 - Disabling select triggers on the source server
 - Disabling indexes, triggers and constraints for each table on the target database server.
 - b. Create and run one script per table for the fully-qualified INSERT and SELECT statements.

For example:

```
INSERT INTO dbname@target:owner.table SELECT *  
FROM dbname@source:owner.table
```

You can run the scripts in parallel. In addition, for larger tables, you can create multiple scripts that can partition the table to run in parallel.

- c. Create and run separate scripts for enabling indexes, triggers and constraints for each table
6. Run UPDATE STATISTICS on system catalog tables and stored procedures and functions on the target database server.
 7. Adjust starting values for all tables that have serial columns on the target database server.
 8. Turn on transaction logging on the source and target database servers.
 9. Return the source and target database servers to multi-user mode.
 10. Validate the data that was transferred to the target database server.

For information about INSERT and SELECT statements, refer to the *IBM Informix Guide to SQL: Syntax*. For information on distributed transactions, refer to the *IBM Informix Administrator's Guide* and the *IBM Informix Administrator's Reference*.

Related concepts:

“Paths for migration to the new version” on page 1-6

Part 2. Migration to and reversion from Version 11.70

Chapter 3. Preparing for migration to Version 11.70

Before you install the new version of Informix, you must prepare the database server environment for migration by performing specified pre-migration tasks. If you are also migrating from 32-bit to 64-Bit database servers, you must perform additional tasks.

Related concepts:

“Upgrading Informix (in-place migration)” on page 1-3

“Types of migration” on page 1-2

“Migrating Informix (non-in-place migration)” on page 1-4

Preparing for migration

Preparing for migration includes gathering information about and backing up your data, so that you can reinstall the previous version of the server and restore your data if you have a migration problem. Preparing for migration is crucial for successful migration.

- Check the IBM Support Portal for the latest patches that you must install before you migrate or upgrade Informix software.
- If you use Enterprise Replication, you must first prepare your replication environment for migration. For more information, see Chapter 4, “Enterprise Replication and migration,” on page 4-1.
- If you are upgrading from Informix Version 9.21 or earlier versions and you have columns for smart large objects (BLOB data types) that were defined with a default value of an empty string, you must redefine the columns to have NULL as the default value.
- If you are upgrading from Informix Version 7.3, you must drop all stored procedures that have embedded CREATE TRIGGER statements, CREATE TABLE statements with fragmentation expressions, ALTER TABLE or CREATE TABLE statements with constraint expressions, and nested CREATE PROCEDURE statements. After migrating to the current database server, you must recreate all stored procedures that were dropped.
- If you are migrating from Informix 7.31, the number of columns of the VARCHAR or NVARCHAR data type per table for Version 11.50 or later versions has been reduced from 231 to 195, within a row size of 32762 bytes and based on a page size of 2K on UNIX or Linux. The same 195-column restriction also applies to BYTE and TEXT columns.

On platforms where the page size is 4K (Windows and AIX®), the limit for the number of columns limit is approximately 450 columns.

Review and complete all tasks that apply:

1. “Reviewing changes in Informix product functionality” on page 3-2.
2. “Checking and configuring available space” on page 3-2.
3. “Configuring for recovery of restore point data in case an upgrade fails” on page 3-4.
4. Renaming user-defined routines (UDRs) that have the following names: CHARINDEX(), LEFT(), RIGHT(), INSTR(), DEGREES(), RADIANS(), REVERSE(), SUBSTRING_INDEX(), LEN(), and SPACE(). These names are reserved for built-in SQL string manipulation functions.

5. Adjusting settings:
 - a. If you use UNICODE, ensure that the GL_USEGLU environment variable on the source server is set to the same value as the GL_USEGLU environment variable on the target server.
 - b. If the source version of the database server contains the IFX_EXTEND_ROLE configuration parameter, which controls authorization to register DataBlade® modules or external UDRs, disable the parameter by setting it to 0 (off).
6. “Saving copies of the current configuration files” on page 3-5.
7. “Saving a copy of the storage manager sm_versions file” on page 3-6.
8. “Closing all transactions and shutting down the source database server” on page 3-6.
9. “Initiating fast recovery to verify that no open transactions exist” on page 3-7.
10. “Verifying the integrity of the data” on page 3-7.
11. “Verifying that the database server is in quiescent mode” on page 3-8.
12. “Making a final backup of the source database server” on page 3-8.

Important: Complete the previous step in case you have to revert to the source database server.

13. “Verifying that the source database server is offline” on page 3-8.

If you use high-availability clusters, you must complete additional preparations. See Chapter 5, “High-availability cluster migration,” on page 5-1.

Related concepts:

“Hardware and operating system requirements” on page 1-5

Related reference:

“Pre-migration checklist of diagnostic information” on page 3-9

Reviewing changes in Informix product functionality

Changes to Informix product functionality might affect your plans for migrating to the latest version of the product.

Changes in functionality in Informix 11.70 can potentially impact your applications, scripts, maintenance processes, and other aspects that are related to your database server environment.

Changes to functionality that was introduced before Informix 11.70 can also affect your plans.

Review the changes that are described in Part 5, “New and changed features in Informix servers,” evaluate the impact to your environment, and plan how to address the changes before migration.

Checking and configuring available space

Before you migrate to the new version of Informix, you must make sure that you have enough available space for the new server, your data, and any other network and data tools that you use.

During migration, Informix drops and then recreates the **sysmaster** database. Depending on which version of Informix you migrate from, the **sysmaster** database in the current version can be significantly larger.

When you migrate to Version 11.70, you need the following space for building **sysmaster**, **sysutils**, and **sysadmin** databases:

- 21892 KB of logical-log spaces (or 10946 pages) for 2 K page platforms
- 26468 KB of logical-log spaces (or 6617 pages) for 4 K page platforms

During migration, a second database, the **sysadmin** database, is created in the **root** dbspace. As you work after migrating, the **sysadmin** database, could grow dramatically. You can move the **sysadmin** database to a different dbspace.

You might need to increase the physical log size to accommodate new features, and you might consider adding a new chunk.

Partition header pages should not be full; key descriptors and other new features might require more space after migration to the new version of Informix.

If your migration fails because there is insufficient space in the partition header page, you must unload your data before you try to migrate again. Then you must manually load the data into the new version.

The root chunk should contain at least ten percent free space when converting to the new version of the server.

In some cases, even if the database server migration 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 Informix.

Add any additional free space to the system prior to the migration. If the dbspaces are nearly full, add space before you start the migration procedure. When you start the new version of Informix on the same root dbspace of the earlier database server, Informix automatically converts the **sysmaster** database and then each database individually.

For a successful conversion of each database, ensure that 2000 KB of free space per database is available in each dbspace where a database resides.

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:

$$X \text{ kilobytes free space} = 2000 \text{ kilobytes} * n$$

The minimum number of databases is 2 (for the **sysmaster** and **sysadmin** databases).

2. Check the amount of free space in each dbspace to determine whether you need to add more space.

You can run 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_avai
  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.

Related concepts:

“Check and configure available space for reversion” on page 7-9

Configuring for recovery of restore point data in case an upgrade fails

By default, the **CONVERSION_GUARD** configuration parameter is enabled and a temporary directory is specified in the **RESTORE_POINT_DIR** configuration parameter. These configuration parameters specify information that Informix can use if an upgrade fails. You can change the default values of these configuration parameters before beginning an upgrade.

Prerequisites: The directory specified in the **RESTORE_POINT_DIR** configuration parameter must be empty before the upgrade begins, but not when recovering from a failed update.

Important:

After a failed upgrade, do not empty the **RESTORE_POINT_DIR** directory before you attempt to run the **onrestorept** utility. The server must be offline after a failed upgrade.

You can change the value of the **CONVERSION_GUARD** configuration parameter or the directory for restore point files before beginning an upgrade. The default value for the **CONVERSION_GUARD** configuration parameter in the **ONCONFIG** file is (2), and the default directory where the server will store the restore point data is **\$INFORMIXDIR/tmp**. You must change this information before beginning an upgrade. You cannot change it during an upgrade.

To change information:

1. If necessary for your environment, change the value of the **CONVERSION_GUARD** configuration parameter.

When the **CONVERSION_GUARD** configuration parameter is set to 2 (the default value), the server will continue the upgrade even if an error related to capturing restore point data occurs, for example, because the server has insufficient space to store the restore point data.

However, if the `CONVERSION_GUARD` configuration parameter is set to 2 and the upgrade to the new version of the server fails, you can use the **onrestorept** utility to restore your data.

However, if you set the `CONVERSION_GUARD` configuration parameter to 2, conversion guard operations fail (for example, because the server has insufficient space to store restore point data), and the upgrade to the new version fails, you cannot use the **onrestorept** utility to restore your data.

2. In the `RESTORE_POINT_DIR` configuration parameter, specify the complete path name for a directory that will store restore point files.

The server will store restore point files in a subdirectory of the specified directory, with the server number as the subdirectory name.

If the `CONVERSION_GUARD` configuration parameter is set to 1 and an upgrade fails, you can run the **onrestorept** utility to restore the Informix instance back to its original state just before the start of the upgrade.

If the `CONVERSION_GUARD` configuration parameter is set to 1 and conversion guard operations fail (for example, because the server has insufficient space to store restore point data), the upgrade to the new version will also fail.

If any restore point files from a previous upgrade exist, you must remove them before you begin an upgrade.

Even if you enable the `CONVERSION_GUARD` configuration parameter, you should still make level 0 backup of your files in case you need to revert after a successful upgrade or in case a catastrophic error occurs and you cannot revert.

Saving copies of the current configuration files

Save copies of the configuration files that exist for each instance of your source database server. Keep the copies available in case you decide to use the files after migrating or you need to revert to the source database server.

Although you can use an old `ONCONFIG` configuration file with Informix Version 11.70, you should use the new Version 11.70 `ONCONFIG` file, or at least examine the file for new parameters. For information about Version 11.70 changes to the `ONCONFIG` file, see Chapter 17, “Configuration parameter changes by version,” on page 17-1.

Configuration files that you might have are listed in Table 3-1.

Table 3-1. Configuration files to save from the source database server

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

Table 3-1. Configuration files to save from the source database server (continued)

UNIX or Linux	Windows
\$INFORMIXDIR/etc/termcap	

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

UNIX or Linux:

`$INFORMIXDIR/etc/ixbar.servernum`

Windows:

`%INFORMIXDIR%\etc\ixbar.servernum`

If you are using a different directory as `INFORMIXDIR` for the new database server, copy `sm_versions` to the new `$INFORMIXDIR/etc`, 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 migration.

Saving a copy of the storage manager `sm_versions` file

Before you migrate 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.

If you are using a different directory as `INFORMIXDIR` for the new database server, copy `sm_versions` to the new `$INFORMIXDIR/etc`, 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 migration.

For information about how to install and use the Informix Storage Manager, see the *IBM Informix Storage Manager Administrator's Guide*.

Closing all transactions and shutting down the source database server

Before migrating, terminate all database server processes and shut down your source database server. This lets users exit and shuts down the database server gracefully. If you have long running sessions, you must also shut those down.

Inform client users that migration time is typically five to ten minutes. However, if migration fails, you must restore from a level-0 backup, so ensure that you include this possibility when you estimate how long the server will be offline.

Before you migrate 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 migration.

To let users exit and shut down the database server gracefully

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

7. Run the **onmode -yuk** command to shut down the system.

If you need to perform an immediate shutdown of the database server, run these commands:

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

Initiating fast recovery to verify that no open transactions exist

A shutdown procedure does not guarantee a rollback of all open transactions. To guarantee that the source database server has no open transactions, put the source database server in quiescent mode and initiate fast recovery.

Run 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 must run the **oninit -s** command before you initialize the new version of Informix. If any transactions remain when you try to initialize the new database server, the following error message appears when you try to initialize the new database server, and the server goes offline:

```
An open transaction was detected when the database server changed log versions.  
Start the previous version of the database server in quiescent mode and then shut  
down the server gracefully, before migrating to this version of the server.
```

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

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

Only after proper shutdown can you bring the new database server (Informix Version 11.70) through the migration path. Any transaction that is open during the migration causes an execution failure in fast recovery.

Verifying the integrity of the data

After verifying that no open transactions exist, verify the integrity of your data by running the **oncheck** utility. You can also verify the integrity of the reserve pages, extents, system catalog tables, data, and indexes. If you find any problems with the data, fix the problems before you make a final backup of the source database server.

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

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


Alternatively, to obtain the database names, run the **oncheck -cc** command without any arguments and filter the result to remove unwanted lines, as shown in this example:


```
oncheck -cc | grep "ting database"
```

Table 3-2 lists the **oncheck** commands that verify the data integrity.

Table 3-2. 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

Related concepts:

 The oncheck Utility (Administrator's Reference)

Verifying that the database server is in quiescent mode

Before you make a final backup, verify that your source database server is in quiescent mode.

Run the **onstat -** command to verify that the database server is in quiescent mode.

The first line of the onstat output shows the status of your source database server. If the server is in quiescent mode, the status line includes this information:

```
Quiescent -- Up
```

Making 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.

Important: You must also make a final backup of 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. Removing the **ixbar** file 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.

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

Verifying that the source database server is offline

Before you install the new database server, verify that the source database server is offline. You must do this 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.

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

Pre-migration checklist of diagnostic information

Before you migrate to a newer version of Informix, gather diagnostic information, especially if you have large, complex applications. This information will be useful to verify database server behavior after migration. This information will also be useful if you need help from IBM Software Support.

If you have problems, you or IBM Software Support can compare the information that you gather with information obtained after migration.

The following table contains a list of the diagnostic information that you can gather. You can print the checklist. Then, after you get the information specified in each row, check the second column of the row.

Table 3-3. Checklist of information to get before migrating

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

Related tasks:

“Preparing for migration” on page 3-1

Migrating from 32-bit to 64-bit database servers

If you are migrating from a 32-bit version of Informix to a 64-bit version of Informix or reverting from a 64-bit version of Informix, 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.

For 32- to 64-bit migrations, change SHMBASE and STACKSIZE according to the **onconfig.std** configuration file for the new version.

All UDRs and DataBlade modules that were built in 32-bit mode must 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 32-bit mode, make sure that proper size and alignment of the data structures are used to work correctly on a 64-bit computer after recompiling in 64-bit mode. For more information, refer to the machine notes.

Migrating from 32-bit to 64-bit with collection types that use the SMALLINT data type

If you are moving your database from a 32-bit computer to a 64-bit computer and your database contains collection types that use the SMALLINT data type, you must take extra steps to prevent memory corruption. Collection types are the ROW, LIST, SET, and MULTiset data types. This restriction applies if you are upgrading from an older version of Informix on 32-bit to the current version of Informix on 64-bit, or if you are moving from 32-bit to 64-bit on the current version of Informix.

To migrate a database with SMALLINT collection types from 32-bit to 64-bit, use one of the following methods:

- Export and import the data.
 1. Export the data from the 32-bit computer.
 2. Import the data onto the 64-bit computer.
- Drop and recreate specific collection types and database objects.
 1. Drop the collection types that use the SMALLINT data type and all other database objects that reference them (such as tables, columns, SPL routines, triggers, indexes, and so on).
 2. Recreate the collections types and all the other necessary database objects.

Chapter 4. Enterprise Replication and migration

You must coordinate the migration of all servers that are involved in data replication.

These topics describe the additional tasks that you must perform when migrating to and reverting from Informix Version 11.70 if you are running Enterprise Replication.

Related concepts:

“Upgrading Informix (in-place migration)” on page 1-3

“Types of migration” on page 1-2

“Migrating Informix (non-in-place migration)” on page 1-4

Preparing to migrate with Enterprise Replication

If you use Enterprise Replication, you must do replication-related tasks to prepare for migration.

You must do all migration operations as user **informix**, unless otherwise noted.

To prepare for migration with Enterprise Replication:

1. If you are migrating from Version 7.31, remove all replicate groups.
In Version 9.3, replicate groups were replaced by replicate set (**replset**) commands.
2. Stop applications that are performing replicable transactions.
3. Make sure that the replication queues are empty by running the following commands:
 - Run **onstat -g grp** to ensure that the Enterprise Replication grouper does not have any pending transactions. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission.
 - Run **onstat -g rqm** to check for queued messages.
4. Replace old event class identifiers (IDs) for Enterprise Replication used in your alarm program with the new event class IDs. Event class IDs for Enterprise Replication were changed in version 10.0 and new event alarms were added in subsequent releases. In version 11.70 and later versions, event alarms have unique IDs in addition to event class IDs.
5. Shut down Enterprise Replication by running the **cdr stop** command.

Now you can complete the steps in “Preparing for migration” on page 3-1 and, if necessary, in “Migrating from 32-bit to 64-bit database servers” on page 3-10.

Related reference:

 [Enterprise Replication Event Alarms \(Enterprise Replication Guide\)](#)

Migrating with Enterprise Replication

If you use Enterprise Replication, you must complete replication-related tasks when you migrate to a new version of Informix.

Prerequisites:

- Complete the steps in “Preparing to migrate with Enterprise Replication” on page 4-1.
- Complete the steps in “Preparing for migration” on page 3-1.
- Perform all migration operations as user **informix**.
- All servers in the Enterprise Replication domain must be available.

To migrate with Enterprise Replication:

1. Perform the tasks that are described in “Migrating to the new version of Informix” on page 6-1, including starting the new version of the server.
2. If the `CDR_QDATA_SBSPACE` configuration parameter is not set, you must set it by specifying the sbspaces for Enterprise Replication to use for storing spooled row data.
3. For each node involved in Enterprise Replication, back up the **syscdr** databases by using the **dbexport -ss** command or the **dbschema -ss** command and the `UNLOAD` statement, or by a combination of these methods. The **-ss** option prevents backup tables from using default extent sizes and row-level locking, which is not an appropriate lock mode with Enterprise Replication.
4. Make sure that no replicable transactions occur before Enterprise Replication starts.
5. If you are upgrading to a new release, 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. Do not run the script when you migrate between fix packs of the same release.

Run the following command:

```
% sh concdr.sh from_version 11.70
```

The `from_version` values that you can use are:

11.50, 11.10, 10.00, 9.40, and 7.31.

6. Wait for one of the following messages:
'syscdr' conversion completed successfully.
'syscdr' conversion failed.
For details about the conversion, see either of the following files:
 - `$INFORMIXDIR/etc/concdr.out`
 - `%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 by running the **cdr start** command.

Important: After you convert to the new version of Informix with Enterprise Replication, do not drop the **syscdr** database. If **syscdr** is dropped, you cannot revert to the older database server with Enterprise Replication because the data required to carry out the reversion is stored in the **syscdr** database.

Related reference:

 [cdr modify grid \(Enterprise Replication Guide\)](#)

Converting replication of 9.21 user-defined data types

If you are migrating from Informix Version 9.21 and use Enterprise Replication, you must complete additional tasks if you have user-defined data types (UDTs).

Informix Version 9.21 has limited support for the replication of UDTs. To take advantage of the UDT replication that is available in newer versions of the server, the user-defined routines (UDRs) for a UDT must contain **streamwrite()** and **streamread()** functions.

After you install the new version of Informix, implement the **streamwrite()** and **streamread()** functions for any currently replicated UDTs on all database servers within the enterprise.

Because you cannot migrate directly from version 9.21 to Version 11.70, use the **dbexport** and **dbimport** utilities or distributed SQL to move your data into the new database server.

Reverting with Enterprise Replication

If you use Enterprise Replication, you must complete replication-related tasks when you revert from the new version of Informix.

Prerequisites:

- Perform all reversion operations as user **informix**.
- Enterprise Replication must be running, or you must delete the replication server before you revert.
- All servers in the Enterprise Replication domain must be available.
- Replication queues must be nearly empty.

During reversion to an earlier version of Informix with Enterprise Replication:

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

This procedure describes reverting to Informix 11.50, 11.10, or 10.00. To revert to an earlier version, first revert to an intermediate version, such as version 11.50 or 10.0.

If you are reverting from version 11.70.xC3 or a later version to version 11.70.xC1 or 11.70.xC2, you do not need to follow the additional Enterprise Replication migration procedures that are described in this topic.

If you want to revert to a version earlier than when Enterprise Replication was defined on this server, you must remove Enterprise Replication from this server before reverting. For more information, see *Deleting a Replication Server* (Enterprise Replication Guide). After you remove Enterprise Replication, you can revert your server using the instructions at Chapter 7, "Reverting from Informix Version 11.70," on page 7-1.

To revert from Version 11.70 with Enterprise Replication:

1. Stop applications that are doing replicable transactions.
2. Remove Enterprise Replication features from the current release that cannot be reverted. For more information, see “Reversion requirements and limitations” on page 7-2.
3. Run the **onstat -g cat repls** command to check whether if Enterprise Replication is in alter mode. If so, run the **cdr alter --off** command.
4. Delete shadow replicates.
5. Make sure that control and TRG send queues are empty:
 - Run **onstat -g grp** to ensure that the Enterprise Replication grouper does not have any pending transactions.
 - Run **onstat -g rqm** to check for queued messages.
6. Shut down Enterprise Replication with the following command:
`cdr stop`
7. Back up the **syscdr** databases with **dbschema** or UNLOAD.
8. If you are doing an in-place reversion, copy the `revcdrintrepl.sql` script and the `revcdrintrepl.sh` or `revcdrintrepl.bat` script from the `$INFORMIXDIR/etc/conv` directory on UNIX or the `%INFORMIXDIR%\etc\conv` directory on Windows, to a directory that is not under the **INFORMIXDIR** directory. These scripts are used to revert the internal replicates. You run the script after you revert to the earlier version.
9. Run the reversion script, named `revcdr.sh`, in the `$INFORMIXDIR/etc/conv` directory on UNIX, or `revcdr.bat`, in the `%INFORMIXDIR%\etc\conv` directory on Windows:

```
% sh revcdr.sh 11.70 to_version
```


Valid `to_version` values are 11.50, 11.10, and 10.00. This script does a reversion test followed by the actual Enterprise Replication reversion.
10. If the reversion test or actual reversion fails, check the file `$INFORMIXDIR/etc/revtestcdr.out` or `revcdr.out`. Attempt reversion after you resolve the problems that are reported.
11. Perform database server reversion tasks, as described in “Reverting from Informix Version 11.70” on page 7-13.
12. Run **onmode -l** and **onmode -c** to prevent the database server from failing when you start Enterprise Replication.
13. Start Enterprise Replication by running the **cdr start** command.
14. If you are doing an in-place reversion, copy the `revcdrintrepl.sql` script and the `revcdrintrepl.sh` or `revcdrintrepl.bat` script to the `$INFORMIXDIR/etc/conv` directory on UNIX or the `%INFORMIXDIR%\etc\conv` directory on Windows.
15. Run the `revcdrintrepl.sh` script on UNIX, or the `revcdrintrepl.bat` script on Windows.

Related tasks:

“Preparing to revert” on page 7-1

Related reference:

 [cdr modify grid \(Enterprise Replication Guide\)](#)

 [Conversion and Reversion Messages for Enterprise Replication \(Administrator's Reference\)](#)

Chapter 5. High-availability cluster migration

You must coordinate the migration and reversion of all servers that are involved in high-availability clusters.

High-availability clusters are based on technology that is sometimes referred to as Multi-node Active Clusters for High Availability (MACH).

Related concepts:

“Upgrading Informix (in-place migration)” on page 1-3

“Types of migration” on page 1-2

“Migrating Informix (non-in-place migration)” on page 1-4

Preparing to migrate, upgrade, or revert clusters

If you use high-availability clusters, you must coordinate the migration of all of the servers that are involved in a cluster, and you must perform additional steps when preparing to migrate.

Prerequisites:

- You must perform all migration operations as user **informix**.
- If you are migrating to a new version of the server, complete all steps in “Preparing for migration” on page 3-1.

To prepare for migration or reversion with clusters:

1. Install the target server (in a different location from where the source database server is installed) on all of the servers in the cluster. Do not install the target server over the source server.
2. When upgrading, install the new version of IBM Informix Client Software Development Kit (Client SDK), which contains the Connection Manager that you must use with the new version of the server. If you only want to use the new Connection Manager in the new version of Client SDK, install Client SDK in a new location and use the Connection Manager from that location.
3. Copy the configuration files (the **onconfig** and **sqlhosts** files) to the target installation directory (for example, **\$INFORMIXDIR/etc**) on all of the servers in the cluster.
4. Install any user-defined objects or DataBlade modules (that are used on the source server) onto all of the servers in the cluster.
5. Back up your primary server. You can perform this step in either of the following ways:
 - Back up all logs. Then use **ON-Bar** or **ontape** to make a level-0 backup on the primary source server.
 - Alternatively, if you have a high-availability cluster with a High-availability Data Replication (HDR) secondary server, you can use the HDR secondary server as a standby server for any contingencies that occur while you upgrade the primary server. However, if it is necessary to use an HDR secondary server as a standby server for contingencies, do not perform updates on the standby server while migration or reversion is in progress,

because the updates cannot be replicated and will be lost. Additionally, nonlogged objects on the primary server will not exist on the secondary server.

Do not use RS secondary servers as backup servers, because transactions could be lost.

Upgrading clusters to a new PID or fix pack

If you are upgrading clusters from one PID or fix pack to a new PID or fix pack and the migration does not involve actual conversion or reversion, you must complete additional tasks when you upgrade.

However, you do not need to recreate the secondary servers after you upgrade the primary database server.

Prerequisites:

- Verify that you are upgrading to a new PID or fix pack in which standard conversion procedures are not necessary. If you are upgrading to a new PID or fix pack that requires you to complete standard conversion procedures or if you are upgrading to a new release, instead of following the procedures in this topic, go to “Migrating clusters to a new release” on page 5-3.
- Complete the steps in “Preparing to migrate, upgrade, or revert clusters” on page 5-1.
- Perform all migration operations as user **informix**.

To upgrade clusters to a new PID or fix pack:

1. Stop the Connection Manager by issuing the **oncmsm -k *connection_manager_name*** command.
2. If you are using a High-availability Data Replication (HDR) secondary server as a backup server in case of contingencies:
 - a. Quiesce the primary server by issuing an **onmode -sy** command to prevent user connections to the server.
 - b. Force a checkpoint by issuing an **onmode -c** command on the primary server.
3. Stop secondary servers in the cluster in the following order:
 - a. If you have remote standalone (RS) servers, stop them by issuing the **onmode -ky** command.
 - b. If you have shared disk (SD) servers, stop them by issuing the **onmode -ky** command.
 - c. If you have an HDR secondary server, stop it issuing the **onmode -ky** command.
4. Stop the primary server by issuing the **onmode -ky** command.
5. On each server, set the INFORMIXDIR environment variable to the full path name for the target installation.
6. Ensure that all of the necessary configuration files are available in the target installation.
7. Start the servers in the cluster and perform additional tasks in the following order:
 - a. Start the primary server by running an **oninit** command.
 - b. Wait for primary server to be in online (multi-user) mode.
 - c. Start the Connection Manager by running an **oncmsm** command.

- d. Start the HDR secondary server by running an **oninit** command.
- e. Start SD servers by running an **oninit** command.
- f. Start RS servers by running an **oninit** command.

Migrating clusters to a new release

If you have a high-availability cluster with one or more secondary database servers and are migrating to a new release or if you are upgrading to a new PID or fix pack that requires you to complete standard conversion procedures, you must complete additional tasks when you migrate.

Prerequisites:

- Complete the steps in “Preparing for migration” on page 3-1.
- Complete the steps in “Preparing to migrate, upgrade, or revert clusters” on page 5-1.
- Perform all migration operations as user **informix**.

When you migrate clusters, you need to migrate only the primary database server.

The server automatically removes secondary servers when you migrate or revert. After migration or reversion on the primary server is complete, you must recreate all High-availability Data Replication (HDR), RS, and SD secondary servers in a high-availability cluster.

When you migrate clusters, be sure to stop and start the servers in the cluster in the order shown in the following procedure.

To migrate to a new release with high-availability clusters:

1. Stop the Connection Manager by issuing the **oncmsh -k** *connection_manager_name* command.
2. If you are using a High-availability Data Replication (HDR) secondary server as a backup server in case of contingencies:
 - a. Quiesce the primary server by issuing an **onmode -sy** command to prevent user connections to the server.
 - b. Force a checkpoint by issuing an **onmode -c** command on the primary server.
3. Stop the secondary servers in the cluster in the following order:
 - a. If you have remote standalone (RS) secondary servers, stop them by issuing the **onmode -ky** command.
 - b. If you have shared disk (SD) servers, stop them by issuing the **onmode -ky** command.
 - c. If you have an HDR secondary server, stop it by issuing the **onmode -ky** command.
4. Stop the primary server by issuing the **onmode -ky** command.
5. On each server, set the INFORMIXDIR environment variable to the full path name for the target installation.
6. Ensure that all of the necessary configuration files are available in the target installation.
7. Optional: Enable quick reversion to a consistent restore point if the migration fails. Do this by setting the **CONVERSION_GUARD** and

RESTORE_POINT_DIR configuration parameters. (For more information, see “Configuring for recovery of restore point data in case an upgrade fails” on page 3-4.)

8. Start the primary server by issuing an **oninit** command.
9. Ensure that the conversion to the target server was successful and that the server is in multi-user mode.
10. Start the Connection Manager by issuing an **oncmsm** command.
11. If you are migrating from pre-11.10 versions of Informix and need SD secondary servers on the primary server in a shared-disk cluster, set the primary server by issuing the **onmode -d set SDS primary primary_server_name** command.
12. Start SD secondary servers by issuing **oninit** commands.
13. Start the servers in the cluster and perform additional tasks in the following order:
14. Back up all logs. Then use ON-Bar or **ontape** to make a level-0 backup on the primary server to use to reestablish the RS and HDR secondary servers if necessary.
15. If you have RS secondary servers:
 - a. Add RS entries on the primary server by issuing **onmode -d add RSS rss_server_name** commands.
 - b. Start RS secondary servers with level-0 restore operations from the level 0 backup that was made on the primary server after migration.
 - c. On RS secondary servers, run the **onmode -d RSS primary_server_name** command, and wait for the "RSS secondary server operational" message to appear after each command.
16. If you have an HDR secondary server:
 - a. Reestablish the pair on the primary server by issuing an **onmode -d primary hdr_secondary_server_name** command.
 - b. Start the HDR secondary server with level-0 restore operations from the level 0 backup that was made on the primary server after migration.
 - c. On the HDR secondary server, run the **onmode -d secondary primary_server_name** command, and wait for the "HDR secondary server operational" message to appear after each command.
17. Perform any additional standard migration tasks described in “Migrating to the new version of Informix” on page 6-1 and in “Completing required post-migration tasks” on page 6-7.
18. If you have Connection Manager configuration files that you created with a version of IBM Informix Client Software Development Kit (Client SDK) that is prior to version 3.70.xC3, you must convert the files. You must convert them because the older files are incompatible with the current version of the Connection Manager. For more information, see Converting older formats of the Connection Manager configuration file to the current format.

The migration of all servers in the cluster is now complete.

Upgrading a cluster while it is online

You can upgrade the Informix software in a high-availability cluster without incurring any down time.

To upgrade Informix software on a high-availability cluster, you first convert the primary server and the secondary to standalone Enterprise Replication (ER)

servers. You then upgrade the software on the secondary server, stop ER, and then clone the server using the **ifxclone** command. You can perform the upgrade while the servers are actively processing client requests because ER supports replication between different versions of the server software.

See the following link for additional information about upgrading clusters:
<http://www.ibm.com/developerworks/data/library/techarticle/dm-1012rollingupgrade/index.html>.

The following prerequisites apply when upgrading software on a cluster:

- Non-logged databases are not supported.
- Raw or unlogged tables are not supported.
- Typed tables are not supported unless the typed table contains a primary key.
- UDTs that do not support ER are not supported.
- The CDR_QDATA_SBSpace configuration parameter must be set on both the primary and secondary servers.
- The `sqlhosts` file must define a server group.
- The primary and secondary servers must belong to different groups.
- For versions of Informix software earlier than 11.50.xC7, converting a primary and secondary server pair to ER is not supported if a table does not have a primary key.
- For versions of Informix software earlier than 11.10, the **sec2er** command is not supported.

If you are upgrading Informix software version 11.50.xC7 or later, the **sec2er** command adds a primary key to any table that does not already have one defined. For large tables, adding the primary key can take a long time, during which you will not see any server activity. In addition, the **sec2er** command requires exclusive access to the table while adding the primary key and user transactions will be blocked from accessing the table. You might want to manually create primary keys on any large table before running the **sec2er** command. If you have tables that were created with the `DELIMIDENT` environment variable set, and the tables do not have primary keys, then you must manually create the primary keys for those tables before running the **sec2er** command.

There are different steps involved in the upgrade process depending on whether you are using the Connection Manager:

- For high-availability clusters that use the Connection Manager to redirect user connections:

There are two options you can choose based on your requirements. You can:

- Add a new Connection Manager instance to manage user connections while the cluster is upgraded. This involves configuring a new 3.70 Connection Manager instance that supports ER and has corresponding changes to the `sqlhosts` file or other connection mechanisms for user applications. If users already have a Connection Manager group support infrastructure to manage their user connections, they can easily add the new Connection Manager for ER to their existing Connection Manager group to ensure that no user connection downtime occurs during the upgrade process.
- Use your existing cluster Connection Manager or Connection Manager groups throughout the upgrade process, without making any changes to the Connection Manager configuration, applications, or application connection mechanisms. This option has a 10-second down time for user connections, but

if that is acceptable, you can avoid the overhead of adding a new Connection Manager instance and the configuration changes that go with it.

- For clusters not using the Connection Manager to redirect user connections: Users must take steps to move user connections to the appropriate servers during the upgrade process.

Performing a server upgrade when the Connection Manager is not in use

In this example, the terms *server1* and *server2* refer to server names rather than machine names.

Some additional steps are required to upgrade Informix software:

1. On the primary server (**server1**), perform a check to see whether the servers can be split into ER servers by running the following command:

```
cdr check sec2er -c server1 --print server2
```

The command examines the primary and secondary servers and determines if it is possible to convert them to ER. The command displays warnings and errors that explain conditions that may prevent the servers from converting to ER. The `--print` option prints the commands that will be run when the **cdr start** command runs. You should fix any warnings or errors and then run the command again before performing the next step.

2. Run the following command from an Informix 11.70 or later server:

```
cdr start sec2er -c server1 server2
```

The **sec2er** command converts the primary and secondary servers into standalone servers and configures and starts Enterprise Replication. Enterprise Replication keeps the data on the servers synchronized; however, any table created after the **sec2er** command is run will not be replicated.

3. On the former secondary server (*server2*), upgrade the Informix software. The steps to upgrade the server are as follows:

- a. Stop replication by running the following command:

```
cdr stop
```

- b. Back up the logical logs:

```
ontape -a
```

- c. Stop the server that contains the older version of the Informix software:

```
onmode -kuy
```

- d. Log on to the server with the newly installed Informix software.

- e. Start the server and let the conversion complete successfully :

```
oninit
```

- f. Run the `concdr.sh` script to convert the `syscdr` database from the old software version to the new version:

```
concdr.sh old_version new_version
```

- g. Start replication on the former secondary server (**server 2**) after it has been upgraded:

```
cdr start
```

Because ER supports replication between dissimilar versions of the server software, the upgraded secondary server (**server2**) replicates data with the former primary server (**server1**), so that data updates are replicated on both servers.

4. Move client application connections from the former primary server (**server1**) to the upgraded server (**server2**).

5. On the primary server (**server1**) use the **onmode -k** command to take the database server to offline mode.

```
onmode -k
```

6. On the former secondary server (**server2**) run the following command to stop ER:

```
cdr stop
```

7. You can now clone the upgraded server to set up the other secondary servers in your cluster. Clone the newly upgraded server (**server2**) by running the **ifxclone** utility on server1. Use the **-d** (disposition) parameter to create a standalone, RSS, or HDR secondary server. In the following examples, assume that the TCP/IP address for **server1** is 192.168.0.1 on port 123, and the address for **server2** is 192.168.0.2 on port 456.

- To create a standalone server:

```
ifxclone -T -S server2 -I 192.168.0.2 -P 456 -t server1  
-i 192.168.0.1 -p 123
```

- To create an RS secondary server specify the disposition using the **-d** option:

```
ifxclone -T -S server2 -I 192.168.0.2 -P 456 -t server1  
-i 192.168.0.1 -p 123 -d RSS
```

- To create an HDR secondary server, specify a disposition of RSS then promote the RSS server to HDR:

```
ifxclone -T -S server2 -I 192.168.0.2 -P 456 -t server1  
-i 192.168.0.1 -p 123 -d RSS  
onmode -d secondary primary_server_name
```

At this point, the cluster is running on the upgraded server. Clients can move applications from server2 if necessary.

Performing a server upgrade when the Connection Manager is in use

Refer to the following steps when clients are using the Connection Manager without a Connection Manager group defined in the existing setup.

For this example, assume that the following Connection Manager configuration file is defined:

```
NAME cm1  
LOG 1  
LOGFILE /tmp/cm1.log  
  
CLUSTER cluster1  
{  
  INFORMIXSERVER ids1,ids2  
  SLA oltp DBSERVERS=primary  
  SLA webapp DBSERVERS=HDR  
  SLA report DBSERVERS=(primary,HDR)  
  FOC ORDER=ENABLED  
}
```

1. On the primary server (**server1**), perform a check to see whether the servers can be split into ER servers by running the following command:

```
cdr check sec2er -c server1 --print server2
```

When the above command is run, the primary and secondary servers are examined to determine whether it is possible to convert them to ER. The command displays warnings and errors that explain conditions that might prevent the servers from converting to ER. The **-print** option prints the commands that will be run when the **cdr start sec2er** command runs. You should fix any warnings or errors and then run the command again before performing the next step.

2. Reload the Connection Manager so that it directs all client connections to the primary server. Here is the revised Connection Manager configuration file:

```

NAME cm1
LOG 1
LOGFILE /tmp/cm1.log

CLUSTER cluster1
{
  INFORMIXSERVER ids1,ids2
  SLA o1tp DBSERVERS=primary
  SLA webapp DBSERVERS=primary,HDR
  SLA report DBSERVERS=primary,HDR
  FOC ORDER=ENABLED
}

```

3. Run the following command from an Informix 11.70 or later server:

```
cdr start sec2er -c server1 server2
```

The **sec2er** command converts the primary and secondary servers into standalone servers and configures and starts Enterprise Replication. Enterprise Replication keeps the data on the servers synchronized; however, any table created after the **sec2er** command is run will not be replicated.

4. On the former secondary server (**server2**), upgrade the Informix software. Because ER supports replication between dissimilar versions of the server software, the upgraded secondary server (**server2**) replicates data with the former primary server (**server1**), so that data updates are replicated to both servers.
5. Move client application connections from the former primary server (**server1**) to the upgraded server (**server2**).
 - a. Create a new Connection Manager configuration file for ER. The following shows a sample ER Connection Manager configuration file. The SLA names are same as for **cm1**:

```

NAME cm2
LOG 1
LOGFILE /tmp/cm2.log
MACRO list=g_server2,g_server1

REPLSET replset_1
{
  INFORMIXSERVER g_server1,g_server2
  SLA o1tp DBSERVERS=${list}
  SLA webapp DBSERVERS=${list}
  SLA report DBSERVERS=${list}
}

```

The ER Connection Manager must define a replicate set that includes all replicates that are generated by the **sec2er** command. You can see the list of replicates by running the following command:

```
cdr list repl
```

You create a replicate set by running the following command:

```
cdr def replset replset_name repl1 repl2 ...
```

In the above example, *repl1* and *repl2* are replicates created by the **sec2er** command.

- b. Halt the **cm1** Connection Manager instance and load the **cm2** instance. Performing the above step ensures that client connections are redirected to **group_2** (because **server2** belongs to **group_2**).

Here is a sample `sqlhosts` file:

#dbservername	nettype	hostname	servicename	options
g_server1	group	-	-	i=10
ids1	onsoctcp	host1	port1	g=g_server1
g_server2	group	-	-	i=20
ids2	onsoctcp	host2	port2	g=g_server2
oltp	onsoctcp	host1	port3	
webapp	onsoctcp	host1	port4	
report	onsoctcp	host1	port5	

- On the primary server (**server1**) use the **onmode -k** command to take the database server to offline mode.

```
onmode -k
```

- You can now clone the upgraded server to the other secondary servers in your cluster.

At this point in the upgrade process, the high-availability cluster is running on the upgraded server.

- Shut down the **cm2** Connection Manager instance and start the **cm1** instance.

- On the former secondary server (**server2**) run the following command to stop ER:

```
cdr stop
```

Errors and warnings generated by the **sec2er** command

The **sec2er** command checks several conditions before converting a primary and secondary server pair to an ER system. The following conditions are checked by the **sec2er** command; ER conversion will take place only if the following conditions are met:

- The group definition must use the **i=** option.
- The CDRID option must be the same on both the primary and secondary servers (The CDRID is the unique identifier for the database server in the Options field of the `sqlhosts` file).
- The `sqlhosts` files on the primary server must match the `sqlhosts` file on the secondary server. The **sec2er** command checks only the lines in the `sqlhosts` files that must to match to support ER.
- The database must not contain a typed table without a primary key.
- User-defined types (UDT) must have ER support.
- Tables must not be protected with label-based access control (LBAC).
- A secondary server must be defined.
- An **sbspace** for a stable queue must exist.
- You must be running Informix version 11.00 or later.

The following warnings might occur. Warnings do not always indicate a problem but should be addressed. A warning is generated if any of the following are true:

- The CDR_SERIAL configuration parameter is not set.
- The values for the CDR_SERIAL configuration parameter are the same on both the primary and secondary servers. Identical values can cause conflicts.
- The database has sequence generators. Because sequence generators are not replicated, if you replicate tables using sequence objects for update, insert, or delete operations, the same sequence values might be generated on different servers at the same time, leading to conflicts.
- The database is not logged.
- A table is not logged.
- The DBSPACE is more than 70-percent full.

If the **cdr start sec2er** command fails or is interrupted, you might see a message that is similar to this message:

```
ERROR: Command cannot be run on pre-11.70 instance if ER is already running
```

If you receive this error, and you do not have the Connection Manager running, remove replication by running the **cdr delete server** command on both servers and then run the **cdr start sec2er** command again. If you have the Connection Manager running, then perform the following steps:

1. Shut down the Connection Manager.
2. Shut down ER using one of the following commands:
 - **cdr delete server** *group*
 - **cdr delete server** *server*
3. Start the Connection Manager.
4. Restart the **sec2er** command:

```
cdr start sec2er
```

Reverting clusters

If you have a high-availability cluster, you must complete additional tasks when you revert from the new version of Informix. You must revert only the primary database server.

The server automatically removes secondary servers during reversion. After reversion on the primary server is complete, you must recreate all HDR, RS, and SD secondary servers in a high-availability cluster.

Prerequisites:

- Determine if you can revert. See information in “Ascertain that reversion is possible and identify reversion requirements” on page 7-2.
- Complete the steps in “Preparing to migrate, upgrade, or revert clusters” on page 5-1.
- Perform all reversion operations as user **informix**.

When you revert clusters, be sure to stop and start the servers in the cluster in the order shown in the following procedure.

To revert high-availability clusters:

1. Stop the Connection Manager by issuing the **oncmism -k** *connection_manager_name* command.
2. If you are using a High-availability Data Replication (HDR) secondary server as a backup server in case of contingencies:
 - a. Quiesce the primary server by issuing an **onmode -sy** command to prevent user connections to the server.
 - b. Force a checkpoint by issuing an **onmode -c** command on the primary server.
3. Stop the servers in the cluster and perform the following tasks in the following order:
 - a. If you have remote standalone (RS) servers, stop them by issuing the **onmode -ky** command.
 - b. If you have shared disk (SD) servers, stop them by issuing the **onmode -ky** command.

- c. If you have a High-availability Data Replication (HDR) secondary server, stop it by issuing the **onmode -ky** command.
 - d. Revert the standard server by issuing an **onmode -b target_IDS_version** command.
 - e. Verify that reversion was successful and the server was stopped. If the reversion was not successful, check the message log for error messages, take appropriate action, and restart reversion.
4. On each server, set the INFORMIXDIR environment variable to the full path name for the target installation.
 5. Ensure that all of the necessary configuration files are available in the target installation.
 6. Perform any additional database server reversion tasks, as described in “Reverting from Informix Version 11.70” on page 7-13.
 7. Start the primary server by issuing an **oninit** command.
 8. Start the Connection Manager by issuing an **oncmsm** command
 9. Start SD secondary servers by issuing **oninit** commands.
 10. Back up all logs. Then use ON-Bar or **ontape** to make a level-0 backup on the primary server to use to reestablish the RS and HDR servers if necessary.
 11. If you have RS secondary servers:
 - a. Add RS entries on the primary server by issuing **onmode -d add RSS rss_server_name** commands.
 - b. Start the RS secondary servers with level-0 restore operations from the level 0 backup that was made on the primary server after reversion.
 - c. On the RS secondary servers, run the **onmode -d RSS primary_server_name** command, and wait for the "RSS secondary server operational" message to appear after each command.
 12. If you have an HDR secondary server:
 - a. Reestablish the HDR pair on the primary server by issuing an **onmode -d primary hdr_secondary_server_name** command.
 - b. Start the HDR secondary server with level-0 restore operations from the level 0 backup that was made on the primary server after reversion.
 - c. On the HDR secondary server, run the **onmode -d secondary primary_server_name** command, and wait for the "HDR secondary server operational" message to appear after each command.

The reversion of all servers in the cluster is now complete.

Related tasks:

“Preparing to revert” on page 7-1

Restoring clusters to a consistent point

You can restore the primary server in a high-availability cluster to a consistent point after a failed upgrade.

Prerequisites:

- Before you began the upgrade, you must have enabled quick reversion, according to information in “Configuring for recovery of restore point data in case an upgrade fails” on page 3-4.
- The server must be offline.

To restore the primary server in a cluster to a consistent point after a failed upgrade:

Run the **onrestorept** utility. For more information, see “Restoring to a previous consistent state after a failed upgrade” on page 6-7.

Alternatively, if you backed up your primary server or you prepared for using the High-availability Data Replication (HDR) secondary server as a backup server before you upgraded and the upgrade fails, you can take other steps to restore the cluster. See “Restoring a cluster from a backup archive” or “Restoring a cluster from the HDR secondary server.”

Restoring a cluster from a backup archive

If you backed up the primary server before you migrated or reverted the cluster, you can restore the primary server from the backup archive if migration or reversion fails. After you restore the primary server, you must recreate the other servers in the high-availability cluster.

Prerequisite: You made a level-0 backup archive of the primary server before migration or reversion.

To restore a cluster from a level-0 backup archive:

1. Point your **INFORMIXDIR**, **PATH**, and any other relevant environment variables to the directory in which the original version of Informix was installed before you migrated or reverted.
2. Using the level-0 backup archive, perform a full restore of your primary server.
3. Recreate the rest of your high-availability cluster.

Restoring a cluster from the HDR secondary server

You can restore the primary server from the High-availability Data Replication (HDR) secondary server that you prepared to use as a backup server before you migrated or reverted the cluster. After you restore the primary server, you must recreate the other servers in the high-availability cluster.

Prerequisite: You prepared the HDR secondary server to use as a contingency backup server, according to information in “Preparing to migrate, upgrade, or revert clusters” on page 5-1.

To restore a cluster from the HDR secondary server:

1. Start the HDR secondary server by running an **oninit** command.
2. Change the secondary server to the primary server by running the **onmode -d make primary *hdr_server_name*** command.
3. If the server is in quiescent mode, change it to multi-user mode by running an **onmode -m** command.
4. Make a level-0 backup using the ON-Bar or **ontape** utility.
5. Recreate the rest of the high-availability cluster.

Chapter 6. Migrating to Informix Version 11.70

When you migrate to a new version of Informix, you must complete required migration and post-migration tasks.

Migrating to the new version of Informix

After you prepare your databases for migration, you can migrate to the new version of Informix.

- Read the release notes and the machine notes for any new information.
- Complete the steps in “Preparing for migration” on page 3-1.

To upgrade a Informix non-root installation, you must run the installation program as the same user who installed the product being upgraded.

If you are migrating the database server from a version that does not support label-based access control, users who held the DBA privilege are automatically granted the SETSESSIONAUTH access privilege for PUBLIC during the migration process. For more information about SETSESSIONAUTH, see the *IBM Informix Guide to SQL: Syntax*. For information about label-based access control, see the *IBM Informix Security Guide*.

Important: Do not connect applications to a database server instance until migration has successfully completed.

Review and complete all tasks that apply:

1. “Installing the new version of Informix” on page 6-2.
2. “Setting environment variables” on page 6-4.
3. “Customizing configuration files” on page 6-4.
4. “Adding Communications Support Modules” on page 6-5.
5. “Installing or upgrading any DataBlade modules” on page 6-5.
6. “Starting the new version of Informix” on page 6-5.

When the migration starts, the `online.log` displays the message `Conversion from version <version number> Started..` The log continues to display start and end messages for all components. When the migration of all components is complete, the message `Conversion Completed Successfully` appears. For more information about this log, see “Migration status messages” on page 6-3.

After migration, see “Completing required post-migration tasks” on page 6-7 for information about preparing the new server for use.

If the log indicates that migration failed, you can either:

- Install the old database server and restore your database from a level-0 backup.
- Run the **onrestorept** utility to back out of the upgrade and restore files to a consistent state without having to restore from a backup. You can run this utility only if you set the configuration parameters that enable the utility. See “Restoring to a previous consistent state after a failed upgrade” on page 6-7.

If the conversion of the High-Performance Loader **onpload** database failed, upgrade the **onpload** database. For more information, see “Upgrading the High-Performance Loader onpload database” on page 6-6.

Installing the new version of Informix

Install and configure the new version of Informix.

If possible, migrate on a database server dedicated to testing your migration before you migrate on your production database server.

Tip: Install a new version of the database server in a new directory and then copy the appropriate configuration files (such as the ONCONFIG and sqlhosts files) from the older version directory to the new directory, modifying them as necessary. If you do not have space on your machine to accommodate the new copy of the Informix software, verify that you have the installation media for the older version of Informix or back up your older directory. You can either remove the older software (but not the configuration files) from the Informix directory or install the new version over the older version. Older versions of files may not be cleaned up if you install over an older version of Informix.

Refer to your installation guide for detailed installation prerequisites, options, and procedures.

Platform considerations

For a Windows installation, you must be a member of the Administrators group.

For a UNIX or Linux administrative installation, you must be logged on as **root**.

For a Mac OS X administrative installation, you are prompted for administrator credentials.

For a Mac OS X, UNIX or Linux non-root installation, you must be logged on as the user who is running **informix**.

If you install a new instance into a directory that already contains an instance and there is insufficient free space, the install program will request that you confirm removal of the older product before extracting the new one. Otherwise, if sufficient free space exists, the install program will add or replace files without deleting the existing instance.

Important: If you install over the existing product, you cannot revert after migration, and you must install the earlier product from the original installation media.

When you change from the old server to the new server, change the INFORMIXDIR environment variable to ensure that it points to the location of the installed database server. If you do not do this, the older version of the database server will start when you reboot.

During subsequent installations, the behavior of the install program depends on the target directory, as follows.

- If you install from the media into the same install location, only the files chosen for replacement or installation will be installed.
- If you are installing into a different location on the hard disk, you must verify that you have enough free disk space tests before file extraction.

After you install a new instance in a new directory, complete data migration and verification.

The installation script installs Informix into the **INFORMIXDIR** directory specified for user **root** on UNIX or Linux.

On Windows, the install application suggests a default **INFORMIXDIR** directory, which you can change by typing a different path. You do not need to create the directory before installation.

Important: Monitor the database server message log, **online.log**, for any error messages. If you see an error message, solve the problem before you continue the migration procedure.

For detailed installation prerequisites, options, and procedures, see your Informix installation guide.

If the installation program alerts you that the destination path is not secure, you have several options for how to proceed. For more information, see Secure a nonsecure Informix installation path in the *IBM Informix Installation Guide for UNIX, Linux, and Mac OS X*.

For information about configuring the new server, see configuration information in the *IBM Informix Administrator's Guide*.

Tips for installing on the server on Windows


If you are installing on Windows:


1. Be sure to choose to install to a different directory.
2. Do not initialize the server when installing.
3. Copy the ONCONFIG file to the target database server and set parameters that are new for the current release.
4. Start the new server without initializing it. See Initializing and starting the database server in your *IBM Informix Installation Guide for Windows*.
5. Monitor the **online.log** for the Conversion Successful message.
6. Once the upgrade has completed successfully, you can remove the old instance. When you run the uninstaller make sure that you select this option:
Retains all databases, but removes server binaries.

Related concepts:

 [Installing Informix and client products on Windows \(Installation Guide \(Windows\)\)](#)

Related tasks:

 [Installing Informix and client products quickly with defaults \(UNIX and Linux\) \(Installation Guide \(UNIX\)\)](#)

 [Installing Informix with selected features and client products \(UNIX and Linux\) \(Installation Guide \(UNIX\)\)](#)

 [Installing Informix on Mac OS X \(Installation Guide \(UNIX\)\)](#)

Migration status messages


When the migration starts, the **online.log** displays the message "Conversion from version *<version number>* Started." The log continues to display start and end messages for all components.

When conversions of all components are complete, the message "Conversion Completed Successfully" displays. This message indicates that the migration process completed successfully, but it does not guarantee that each individual database was converted successfully. The message log might contain more information about the success or failure of the migration of each individual database. If migration of a particular database fails, then try to connect to the database to find out the exact cause of the failure.

At the end of the migration of each individual database, Informix runs a script to update some system catalog table entries. The message log includes messages that are related to this script. The success or failure of the script does not prevent the usage of a database.

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

Related reference:

 Conversion and reversion error messages (Administrator's Reference)

Setting environment variables

After you install the current version of Informix, verify that the **INFORMIXDIR**, **INFORMIXSERVER**, **ONCONFIG**, **PATH**, and **INFORMIXSQLHOSTS** (if used) environment variables are set to the correct values.

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.

The setting of the **GL_USEGLU** environment variable must match between the source and target server during migration.

Important: Before you start the Version 11.70 database server, you must set the **DBONPLOAD** environment variable to the name of the **pload** database if the name is not **onpload**, the default name.

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

Customizing configuration files

When you initialize the new version of Informix, which contains a new **onconfig.std** file, use the same configuration that the old database server used. After you observe the performance of new version, you can examine the new file for new configuration parameters that you might want to use and can you start to use the new and changed configuration 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 migration. For more details, see "Starting the new version of Informix" on page 6-5. After the migration, change the value of **ALARMPROGRAM** to **log_full.sh**.

If the **ALARMPROGRAM** configuration parameter is set to the script **alarmprogram.sh**, set the value of **BACKUPLOGS** in **alarmprogram.sh** to **N**.


Important: To facilitate migration (and reversion), use the same values for your new database server for `ROOTOFFSET`, `ROOTSIZE`, and `ROOTPATH` that you used for the old 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.

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 example, the location of the shared library might be `$INFORMIXDIR/lib/ipldd11a.SOLIBSUFFIX`, where `SOLIBSUFFIX` is the shared-library suffix for your operating system.

For information about how to configure Informix, see your *IBM Informix Administrator's Guide*. For information about how to tune the configuration parameters, see the *IBM Informix Performance Guide*.

Related reference:

Chapter 17, "Configuration parameter changes by version," on page 17-1

 onconfig Portal: Configuration parameters by functional category (Administrator's Reference)

Adding Communications Support Modules

For communications with clients, you can optionally use a Communications Support Module (CSM) with the current version of Informix. After you install the CSM components, create entries in the `concsm.cfg` file and in the options field of the `sqlhosts` file to configure the CSM.

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 install and configure the CSM.

For information about how to set up the CSM, see the *IBM Informix Administrator's Guide*.

Installing or upgrading any DataBlade modules

After you install the new version of Informix, you might need to install or upgrade any DataBlade modules that you want to add to the database server.

Register the DataBlade modules after you initialize the database server.

When you install Informix, the TimeSeries and Spatial extensions are installed and registered automatically. You do not need to perform any actions to upgrade these DataBlade modules, nor do you need to unload and load spatial or time series data during migration. The other built-in extensions (basic text search, node data type, binary data types, large object locator, MQ messaging, and Informix web feature service) are also installed, upgraded, and registered automatically.

Note: If the `sysadmin` database does not exist or the Scheduler is turned off, automatic registration does not occur. You must register each extension that you want to use by running the `SYSBlDPrepare()` function.

Starting the new version of Informix

After installing the new database server, start the server. Do not perform disk-space initialization, which overwrites whatever is on the disk space.

Prerequisite: If you installed Informix as user **root**, you must switch to user **informix** before starting the server.

Important: Informix 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, Informix stops 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 migration. 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 migration, you must back it up with **ontape**, the only backup tool you can use at this point. Issue the **ontape -a** command.

Start the new version of Informix for the first time by running **oninit** command on UNIX or by using the **Service** control application on Windows.

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

Upgrading the High-Performance Loader **onpload** database

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

Starting with Version 9.40.xC3, Informix has a new version of the **onpload** database with longer column lengths. The **onpload** database now requires slightly more disk space than it did before Version 9.40.xC3.

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

To upgrade the **onpload** database:

1. If you are upgrading from a version of Informix 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 you are upgrading from a version of the database server that is prior to Version 9.40.xC3, you must also perform one of the following tasks:
 - Run the **conpload.sh** script, as shown in this example:

```
conpload.sh 9.40 11.50
```
 - Set the new environment variable **IFX_ONPLOAD_AUTO_UPGRADE** to 1 for the upgrade to happen automatically the first time you start 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 start an HPL utility before upgrading the **onpload** database, then you receive an error stating that the **onpload** database must be converted.

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

Restoring to a previous consistent state after a failed upgrade

If the **CONVERSION_GUARD** configuration parameter is enabled and an upgrade fails, you can run the **onrestorept** command to undo the changes that are made during the upgrade and restore Informix to a consistent state.

The following prerequisites must be met:

- The directory that is specified by the **RESTORE_POINT_DIR** configuration parameter contains the files that were stored during the failed upgrade attempt.
- The server must be offline after a failed upgrade.

To restore the server to a previous consistent state after a failed upgrade:

1. Run the **onrestorept** command.

Important: Do not start the server until the **onrestorept** utility finishes running. Starting the server before the **onrestorept** utility finishes running can damage the server, requiring the server to be restored from a backup copy.

Optionally, use the **-y** option if you want the utility to respond **y** (yes) to every prompt that appears while the **onrestorept** command runs.

If you do not specify **-y**, you must respond to every prompt that appears. Valid responses are **y**, **Y**, **n**, or **N**. For example, if you do not specify **-y**, you can decide whether to proceed with the upgrade whenever the prompt **OK to proceed (Y/N)** appears.

2. If the upgrade is successful, run the **onrestorept -c** command to remove the files in the directory that is specified in the **RESTORE_POINT_DIR** configuration parameter.

You can also run the **onrestorept -V** command to display the version of the current server and the software serial number.

Example

The following command restores files, displaying prompts while the command runs:

```
onrestorept -y
```

After you run the **onrestorept** command, you can resume work in the source version of the server, find the problem that caused the failed upgrade, or begin the upgrade again.

Completing required post-migration tasks

After you migrate, you must complete a series of post-migration tasks to prepare the new version of the server for use.

To complete post-migration tasks:

1. For ON-Bar, copy the previous **sm_versions** file. For more information, see “For ON-Bar, copy the **sm_versions** file” on page 6-8.
2. Optionally run **UPDATE STATISTICS** on your tables (not system catalog tables) and on UDRs that perform queries, if you have performance problems

after migrating. For more information, see “Optionally update statistics on your tables after migrating” on page 6-9.

3. Run UPDATE STATISTICS on some system catalog tables
4. Review client applications and registry keys. For more information, see “Review client applications and registry keys” on page 6-9.
5. Verify the integrity of the data. For more information, see “Verify the integrity of migrated data” on page 6-10.
6. Make an initial backup of the new version of Informix . For more information, see “Back up Informix after migrating to the new version” on page 6-10.
7. Tune the new version of Informix for performance and adjust queries as necessary. For more information, see “Tune the new version for performance and adjust queries” on page 6-10.
8. If you use Enterprise Replication, perform Enterprise Replication migration tasks. For more information, see “Migrating with Enterprise Replication” on page 4-2.
9. If you use high-availability clusters, perform additional migration tasks. For more information, see Chapter 5, “High-availability cluster migration,” on page 5-1.
10. Register any DataBlade modules that you installed. For more information, see “Register DataBlade modules” on page 6-11.

After you migrate and start using the new version, refer to Chapter 7, “Reverting from Informix Version 11.70,” on page 7-1 in case you need to revert.

Repeat the migration and post-migration procedures for each instance of Informix Version 11.70 that you plan to run on the computer.

If a serious error occurs during the migration, you might need to return to the previous version of the server, restore from a level-0 backup, and then correct the problem before restarting the migration tasks.

Important: Do not connect applications to a database server instance until the migration has successfully completed.

After successful migration to Informix Version 11.70, you might want to modify configuration files and environment variables to take advantage of Informix Version 11.70 features.

Related reference:

Chapter 17, “Configuration parameter changes by version,” on page 17-1

Chapter 16, “Environment variable changes by version,” on page 16-1

For ON-Bar, copy the sm_versions file

After migration, copy your previous sm_versions file for the ON-Bar backup and restore system to run.

If you are using the same version of ISM, copy the same sm_versions file from your old database server to the new database server installation.

If you are using other storage managers, copy your previous sm_versions file from the old \$INFORMIXDIR/etc directory to the new \$INFORMIXDIR/etc directory.

If you are migrating from Version 7.31 or moving data from that version to the current version, unload the contents of the **sysutils:bar_version** table.

Optionally update statistics on your tables after migrating

Optionally run UPDATE STATISTICS on your tables (not system catalog tables) and on UDRs that perform queries, if you have performance problems after migrating to the new version of Informix.

An unqualified UPDATE STATISTICS statement that does not specify a table and column scope clause and a resolution clause updates all tables and all UDRs that are written in SPL.

You do not need to run UPDATE STATISTICS statements on C or Java UDRs.

Update statistics on some system catalog tables after migrating

After migrating successfully to Informix Version 11.70, run UPDATE STATISTICS on some of the system catalog tables in your databases.

If you are migration from a Version 7.31 or 7.24 database server or moving data from that version to the current version, be sure to run UPDATE STATISTICS on the following system catalog tables in Informix Version 11.70:

- sysblobs** system catalog table
- syscolauth** system catalog table
- syscolumns** system catalog table
- sysconstraints** system catalog table
- sysdefaults** system catalog table
- sysdistrib** system catalog table
- sysfragauth** system catalog table
- sysfragments** system catalog table
- sysindices** system catalog table
- sysobjstate** system catalog table
- sysopclstr** system catalog table
- sysprocauth** system catalog table
- sysproceduresysroleauth** system catalog table
- sys synonyms** system catalog table
- sys syntable** system catalog table
- sysstabauth** system catalog table
- sys tables** system catalog table
- sys triggers** system catalog table
- sys users** system catalog table

Because you cannot migrate directly from a Version 7.31 or 7.24 database server to Version 11.70, use the **dbexport** and **dbimport** utilities or distributed SQL to move your data into the new database server.

Review client applications and registry keys

After you migrate 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.

If necessary, recompile or modify client applications.

Verify that the client-application version you use is compatible with your database server version. If necessary, 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.

Verify the integrity of migrated data

Open each database with DB-Access and use `oncheck` to verify that data was not corrupted during the migration process.

You can also verify the integrity of the reserve pages, extents, system catalog tables, data, indexes, and smart large objects, as Table 6-1 shows.


Table 6-1. 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>

If the `oncheck` utility finds any problems, the utility prompts you to respond to corrective action that it can perform. If you respond Yes to the suggested corrective action, run the `oncheck` command again to make sure the problem has been fixed.

The `oncheck` utility cannot fix data that has become corrupt. If the `oncheck` utility is unable to fix a corruption problem, you might need to contact Technical Support before you proceed.

Related concepts:

 [The oncheck Utility \(Administrator's Reference\)](#)

Back up Informix after migrating to the new version

Use a 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 old 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 old database server for your new database server.

Tune the new version for performance and adjust queries

After backing up the new server, you can tune the database server to maximize performance. If your queries are slower after the upgrade, there are steps you can take to adjust your configuration and queries.

If your queries are slower after an upgrade, find out what changed that affects your configuration and adjust your configuration and queries as necessary:

- Compare the default values in the new `onconfig.std` file to values in your previous installation, and make any necessary adjustments (see New configuration parameters).
- If you created sample queries for comparison, use them to analyze the performance differences between the old and new database servers and to determine if you need to adjust any configuration parameters or the layout of databases, tables, and chunks.
- If you changed your applications, check to see if the changes led to slower performance.
- If you changed your hardware, operating system, or network settings, determine if you need to adjust any related settings or environment variables.
- Make sure that you ran necessary update statistics after upgrade:
 - Drop data distributions if necessary when upgrading
 - Optionally update statistics on your tables after migrating
 - Update statistics on some system catalog tables after migrating

Related concepts:

 [Improving individual query performance \(Performance Guide\)](#)

Register DataBlade modules

You must register any DataBlade modules that you installed.

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 *IBM Informix DataBlade Module Installation and Registration Guide*.

Related concepts:

 [DataBlade Module Installation and Registration Guide \(DataBlade Module Installation Guide\)](#)

Chapter 7. Reverting from Informix Version 11.70

You can revert to the version of the database server from which you migrated. When you run the reversion utility, you specify the target server for reversion and then Informix checks your database. If necessary, Informix might tell you to drop new objects, before automatically converting your data into the target server.

If Informix cannot revert a database, Informix prevents reversion.

Normally, reversion takes only a few minutes.

If you used the new features of Version 11.70, reversion time is longer, because you must prepare your database and data for reversion, and you must remove the features that are not supported in the earlier version of the server. The more work you complete in the new version, the more time consuming the reversion. See “Preparing to revert” before you revert.

If you did not use any of the new features of Version 11.70 and you did not complete much work using the new server, you can run the reversion utility and modify the values of the configuration parameters. See “Reverting from Informix Version 11.70” on page 7-13.

Preparing to revert

If you used Version 11.70, you must prepare your system for reversion to the pre-migration version of the database server.

Review and complete all tasks that apply:

1. “Ascertain that reversion is possible and identify reversion requirements” on page 7-2.
2. “Check and configure available space for reversion” on page 7-9.
3. “Save copies of the current configuration files” on page 7-10.
4. “Save system catalog information” on page 7-10.
5. “Verify the integrity of the Version 11.70 data” on page 7-10.
6. “Back up Informix Version 11.70” on page 7-11.
7. Export or save your data.
8. To prevent reversion issues:
 - a. “Run dummy UPDATE statements” on page 7-11. This step locates and removes pending in-place alter operations.
 - b. If you have empty tables with no extents, drop those tables.
 - c. Defragment partitions by using the SQL administration API.
9. “Remove unsupported features” on page 7-12.
10. If you are reverting to the following versions, drop indexes if necessary:
 - Informix Version 7.3: You must drop any index with a key size that is greater than 254.
 - Informix Version 9.4: You must drop any index whose key size is greater than 390.

The maximum key size in Versions Version 11.70, 11.10, and 10.00 is 3200.

11. "Remove new BladeManager extensions" on page 7-13.

If you use high-availability clusters or Enterprise Replication, you must complete extra tasks before you can revert to the pre-migration version of Informix.

Related tasks:

- "Reverting clusters" on page 5-10
- "Reverting with Enterprise Replication" on page 4-3
- "Reverting from Informix Version 11.70" on page 7-13

Related reference:

 defragment argument: Dynamically defragment partition extents (SQL administration API) (Administrator's Reference)

Ascertain that reversion is possible and identify reversion requirements

You can revert from Informix Version 11.70 to the database server from which you migrated, if you have not added any extensions to the Version 11.70 database server and you are not reverting from a newly created instance. You must review your database schema to determine if reversion is possible.

See "Reversion requirements and limitations" for limitations on reversion to previous databases and prerequisite steps you must take before you revert.

To review the database schema to determine if reversion is possible:

1. Run the **dbschema** utility command.
For example, run the following command to display information about the database **db1**:

```
dbschema -d db1 -ss
```
2. Determine if the schema file contains SQL statements that the earlier database server does not support.
3. Determine if the database contains features, such as long identifiers, that the earlier database server does not support. See Part 5, "New and changed features in Informix servers."
4. Determine if any new SPL routines have been created in Informix Version 11.70 or if any routines were imported using **dbimport**.
5. Determine if tables or indexes using expression fragmentation had expressions changed or new fragments added.
6. Identify any new triggers, procedures, or check constraints.

Reversion requirements and limitations

If you used the new database server, you must review a list of reversion requirements and limitations, and then complete any prerequisite tasks before you revert. If the reversion restrictions indicate that you must drop objects from the database, you can unload your data and then reload it in the prior database server.

Reversion requirements and limitations are described in the following tables:

- Table 7-1 on page 7-3 Requirements and limitations when reverting to any version of the server
- Table 7-2 on page 7-3 Requirements and limitations when reverting to a specific version of the server

Table 7-1. Requirements and limitations when reverting to any version of the server

Reversion requirement or limitation
Revert only to the version from which you migrated: If you need to revert, you must revert to the Informix version that was your source version before you migrated to Version 11.70.
New databases created in the new version of the server: If you created a new database in the new version of the server, you cannot revert the database back to an earlier version of the server. If the data is required, you can unload the data and reload it in the prior version of the server.
New procedures, expression-based fragmented tables, expression-based fragmented indexes, check constraints, and triggers: These cannot be reverted. You must remove any new procedures, fragmented tables, expression-based fragmented indexes, check constraints, and triggers. Note: Expression-based fragmentation includes fragment by expression, fragment by interval, and fragment by list.
New built-in routines: These cannot be reverted.
New configuration parameters: These cannot be reverted.
New or outstanding in-place alters: In-place ALTER TABLE statements performed in the new version of the server must not be outstanding against any table.
If a table has an incomplete new in-place ALTER operation, you must 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, the reversion process lists all the tables that need dummy updates. You must perform a dummy update on each of the tables in the list before you can revert to the older database server. For more information, see “Run dummy UPDATE statements” on page 7-11.
Important: Any in-place alter that was completed in a version that is before the current version will successfully revert and dummy updates are not necessary for them.

Table 7-2. Requirements and limitations when reverting to a specific version of the server

Reversion requirement or limitation	If reverting to this server or earlier servers
LATERAL keyword in SPL routines: You must drop all SPL routines that include the LATERAL keyword.	12.10.xC1, 11.70.xC7 or earlier
Enterprise Replication key specified by the --anyUniqueKey or --key options: You must remove any Enterprise Replication replicates that include the --anyUniqueKey or --key options to define the replication key.	11.70.xC7
Serial processing of Enterprise Replication replicates: You must remove any Enterprise Replication replicates that include the --serial option.	11.70.xC7
Time series containers and tables with large page sizes: You cannot revert time series tables and containers that have a page size other than the default size. You must drop all time series tables and containers with large pages sizes before you revert. (You can unload the tables and drop the tables and containers before you revert, and then later recreate the containers and reload the tables.)	11.70.xC2
Time series containers and time series data type names longer than 18 bytes: You cannot revert time series containers and time series data types that have names that are longer than 18 bytes. You must drop all tables that have time series data types with long names and drop all time series containers that have long names before you revert. (You can unload the tables and drop the tables and containers before you revert, and then later recreate the containers and reload the tables.)	11.70.xC2
Time series virtual tables: You must drop all time series virtual tables that were created by the TSCreateExpressionVirtualTab procedure and returned by this query before you revert: SELECT a.tabname FROM systables a, systabamdata b WHERE LENGTH(b.am_param) > 256 AND a.tabid = b.tabid;	11.70.xC2

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
Informix Warehouse Accelerator reversion requirements: If you use the Informix Warehouse Accelerator and need to revert, see Reversion requirements for an Informix warehouse server and Informix Warehouse Accelerator (Informix Warehouse Accelerator Guide) in the <i>IBM Informix Warehouse Accelerator Administration Guide</i> .	11.70.xC2
Databases created as NLSCASE INSENSITIVE: You cannot revert databases that were created as NLSCASE INSENSITIVE. You must drop all databases that were created with the NLSCASE INSENSITIVE property before you revert.	11.70.xC1
Forest of trees indexes: If you created forest of trees indexes, you must drop them before you revert.	11.50
A dbspace that exceeds 2147483647 pages: If the total size of a dbspace exceeds 2147483647 base pages (for example, 4 terabytes for a 2K page size, 8 terabytes for a 4K page size), reversion will fail. If this happens, you must reorganize your dbspaces and chunks so that the total size of an individual dbspace does not exceed 2147483647 base pages.	11.50
Interval and list fragmentation strategies: Any table or index with interval or list fragmentation strategy must be dropped before reversion.	11.50
The sysfragdist table and related schema changes: The <code>sysfragdist</code> system catalog table and changes to the schema or to the encoding of other system catalog tables to support fragment level statistics and fragmentation strategies will be dropped during reversion.	11.50
Sequence objects: You can revert these unless the server does not have the space for allocating extents for new sequence objects. If the server flags these objects because it cannot revert them, you must use the <code>DROP SEQUENCE</code> statement to drop from the database any sequence objects that the database server flags.	11.50
External tables and empty tables: You can revert these unless the server does not have the space for allocating extents for new external and empty tables. If the server cannot revert these objects, you must drop them before reverting. If you are reverting to a version of the server that is before version 11.50.xC6, you must drop all external tables before reversion. The <code>SYSEXTTERNAL</code> , <code>SYSEXTCOLS</code> , and <code>SYSEXTDFILES</code> system catalog tables also will be dropped during reversion.	11.50
Disabled foreign key indexes: You must drop these and recreate the affected constraints without the index disabled option before you can revert.	11.50
MULTI_INDEX, STAR_JOIN, and related query optimizer directives: These are not supported after reversion.	11.50
Disabled replication server: Before reverting, you must enable or delete the replication server.	11.50
Master server for quality of data: Before reverting, you must stop monitoring the quality of data or define a new master server for quality of data.	11.50
Grid member: Before reverting, you must remove the server from the grid.	11.50
ERKEY shadow columns: Before reverting, you must drop the ERKEY shadow columns by running the <code>ALTER TABLE</code> statement with the <code>DROP ERKEY</code> clause.	11.50
UDRs that use the SET ENVIRONMENT RETAINUPDATELOCKS syntax: Before reverting, you must drop these UDRs.	11.50.xC5
MERGE statements that include the Delete clause: Before reverting, you must drop these routines.	11.50.xC5
Reversion if you have high-availability clusters: Before reverting, see “Reverting clusters” on page 5-10.	11.50.xC5
MERGE statements: Before reverting , you must drop any routines that use the <code>MERGE</code> statement.	11.50.xC4

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
SELECT statements that include the CONNECT BY clause: Before reverting you must drop any routines that use queries or subqueries that include the CONNECT BY clause, and drop any views that are defined by SELECT statements that include the CONNECT BY clause. After reversion, SYS_CONNECT_BY_PATH() is not supported as a built-in routine.	11.50.xC4
ifx_replcheck shadow column: Before reverting, you must drop the ifx_replcheck shadow column.	11.50.xC4
Compressed tables and compressed table fragments: You must uncompress or drop compressed tables and fragments before reverting.	11.50.xC3
UDRs that use methods or SQL statements that reference savepoints: You must drop these UDRs, because they include new SQL syntax that earlier Informix versions do not support. (Before you can compile these UDRs, you must rewrite their error-handling code, so that no savepoint objects are referenced.)	11.50.xC2
New indexes in sbspaces: If you built indexes in sbspaces so you could search the sbspaces with the Basic Text Search DataBlade module, you must drop the indexes before reverting.	11.50.xC2
Version columns in tables: If you have version columns in tables, you must remove them.	11.10
BIGINT and BIGSERIAL columns: If you have any BIGINT or BIGSERIAL columns, you must modify or remove them.	11.10
Extended data types or attributes based on BIGINT and BIGSERIAL data types: If you have these, you must remove them.	11.10
Casts based on BIGINT and BIGSERIAL data types: If you have these, you must remove them.	11.10
Components installed with the custom installation option: If you installed components with the custom installation option, you can uninstall a component only if you are not breaking any component dependencies.	11.10
JAVA UDRs that have been compiled using newer versions of JDK: These must be recompiled with older JDK versions. For details, see “Recompile any Java UDRs that were compiled using JDK 5.0” on page 7-15.	11.10
<p>Subqueries in DELETE and UPDATE statements: If a condition with a subquery in the WHERE clause of DELETE or UPDATE references the same table that the DELETE or UPDATE statement modifies, before you revert, you must rewrite the INSERT or DELETE operation as two separate SQL statements:</p> <ul style="list-style-type: none"> • A SELECT statement that returns qualifying rows of the original table to a temporary table • A DELETE or INSERT statement that modifies the original table by inserting or deleting rows that match rows in the temporary table 	11.10
Returned data type from CONCAT and other SQL string-manipulation functions: Because these built-in functions now support promotion of their return value to longer data types, some operations on VARCHAR or NVARCHAR values might fail with overflow error -881 after reversion.	11.10
Automatic update statistics feature: Informix versions that are earlier than version 11.10.xC1 do not support the Scheduler. Therefore, the functionality of the Automatic Update Statistics feature, which is implemented by the Scheduler, is not available after reversion. To enforce any Automatic Update Statistics policies that you intend to apply to your databases, you must manually issue the corresponding UPDATE STATISTICS statements after reversion to Version 10.00 or to an earlier version.	10.00

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
<p>ANSI joins in distributed queries: Distributed queries that use ANSI-compliant LEFT OUTER syntax for specifying joined tables and nested loop joins run more efficiently in Version 10.00.UC4 than in earlier releases. This occurs through sending the query to each participating database server for operations on local tables of those servers. If you revert from Version 10.00.UC4 or later to an earlier release that does not support this implementation of the ANSI-compliant syntax, such queries might show reduced performance, because the database server instance from which the query originates will perform the joins locally.</p>	10.00.xC4
<p>The INDEX_SJ and AVOID_INDEX_SJ optimizer directives: When queries explicitly use the new INDEX_SJ and AVOID_INDEX_SJ optimizer directives, these directives have no effect when the query runs. You must run UPDATE STATISTICS on stored procedures to force re-compilation of stored procedures.</p> <p>In addition, reversion removes the effect of these directives in SAVE EXTERNAL DIRECTIVES statements and on output from the SET EXPLAIN statement. If you revert to a version of the database server that supports the sysdirectives system catalog table, but does not support the AVOID_INDEX_SJ or INDEX_SJ directives, user informix must delete any active row of sysdirectives that includes AVOID_INDEX_SJ or INDEX_SJ in the directives column.</p>	10.00
<p>sysdbopen() and sysdbclose() procedures: Earlier versions of the database server do not support these procedures, and any UDRs with these names are not automatically started.</p>	10.00
<p>UDRs include a collection-derived table in the FROM clause of a query: These will not work correctly after reversion to an earlier release.</p>	10.00
<p>Multiple BEFORE, FOR EACH ROW and AFTER triggers for the same INSERT, UPDATE, DELETE, or SELECT event on a table or view, and trigger routine UDRs: Before reverting, you must drop any of the following triggers and UDRs, if they exist in the database:</p> <ul style="list-style-type: none"> • Delete triggers defined on the same table or defined on a view as another Delete trigger • Insert triggers defined on the same table or defined on a view as another Insert trigger • Update triggers defined on the same table or view (or on the same subset of columns) as another Update trigger • Select triggers defined on the same table or same subset of columns as another Select trigger • Trigger routines defined with the FOR TRIGGER keywords • Triggers that use the DELETING, INSERTING, SELECTING, or UPDATING operators in their triggered action 	10.00
<p>Cross-server operations on BOOLEAN, LVARCHAR, or DISTINCT columns: If you revert to a database server version that does not support cross-server operations on BOOLEAN, LVARCHAR, or DISTINCT columns in databases, applications that use this feature will fail.</p>	10.00
<p>sysadmin database: This database is automatically dropped during reversion.</p>	10.00
<p>Queries that use SKIP and LIMIT: Version 10.00.xC3 supports queries that use the keywords SKIP and LIMIT. A query that uses either of these keywords will fail with an error after reversion to any earlier version of the database server.</p>	10.00
<p>FIRST clause with an ORDER BY clause: Version 10.00.xC3 supports the ORDER BY clause of the SELECT statement in a subquery whose result set is a collection-derived table (CDT), but only in conjunction with the SKIP keyword or the FIRST keyword (or its keyword synonym LIMIT) in the Projection clause of the same SELECT statement. Queries that use this syntax will fail with an error after reversion to an earlier version of the database server.</p>	10.00
<p>Label-based access control (LBAC): Before reverting, you must drop any security policy from tables. In addition, because IDSSecurityLabel is a new built-in type for Version 11.10, you must remove any columns of that type before you can revert to versions that are earlier than Version 11.10.</p>	10.00

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
Support for Distributed Relational Database Architecture™ (DRDA): Informix drops stored procedures for metadata that the database server created automatically. You cannot manually drop these built-in stored procedures.	10.00
UDRs and applications that use the TRUNCATE Keyword or the am_truncate() method: You cannot revert these to a pre-10.00 version of Informix. During reversion, you must drop or revise any routines that use the TRUNCATE statement, including any newly registered Virtual-Table Interface or Virtual-Index Interface purpose functions.	9.40
Column-level encryption: If your tables contain encrypted data, do not revert to a version of the server that does not contain encryption support (any version before 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 the database server. Informix does not record whether encrypted data is stored in a database.	9.40
XA data sources and XA data source types: If you create any XA data sources and XA data source types, you must drop these.	9.40
DataBlade user-defined routines (UDRs) that include 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.	9.40
New or changed 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.	9.40
Multiple INOUT parameter support: You must drop any new UDRs that were created using INOUT parameters.	9.40
Default roles: During reversion, the defrole column is dropped from the sysusers table. You must revoke default roles from users before reverting.	9.40
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-10.00 version of the database server. Informix drops the partition column from the sysfragments table during reversion.	9.40
External optimizer directives: You cannot revert if external optimizer directives have been created.	9.40
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-10.00 version of Informix.	9.40
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.	9.40
Procedures, functions, and triggers created with new version: You must drop all triggers, procedures, and functions created with the new version before reverting.	9.40
Maximum chunk size: 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.	9.40
Key length of B-tree indexes: All indexes must be B-tree indexes with a total key length less than or equal to 390.	9.40 (not applicable for 7.31)
Maximum file size: No files larger than 2 GB can be stored in or used by the database server.	9.30
TAPESIZE and LTAPESIZE limitation: The TAPESIZE or LTAPESIZE configuration parameters cannot be set to 0.	9.30
ALARMPROGRAM limitation: The ALARMPROGRAM configuration parameter cannot be set to the alarmprogram.sh file.	9.30

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
LRU_MAX_DIRTY and LRU_MIN_DIRTY limitation: If reverting, set these configuration parameters to integers. (These configuration parameters were removed in Version 10.0.)	9.30
UDRs with IN or OUT parameters: UDRs must not use multiple IN or OUT parameters. Drop all such UDRs before reversion.	9.30
UDRs with named return parameters: UDRs and stored procedures must not use named return parameters.	9.30
Sequence objects: Sequence objects must not be in use. Drop all sequences before reversion.	9.30
Triggers created with the INSTEAD OF clause: Triggers created with the INSTEAD OF clause must not be in use. Drop all such triggers before reversion.	9.30
Multiple collations: Multiple collations among indexes, stored procedures, triggers, and constraints must not be in use.	9.30
Functional index limitations: Functional indexes cannot contain more than 16 parameters.	9.30
High-data availability replication and Enterprise Replication in the same server: High-Data Availability Replication and Enterprise Replication cannot co-exist on the same database server.	9.30
LVARCHAR limitation: The LVARCHAR(<i>n</i>) data types must not be in use if <i>n</i> is not equal to 2042.	9.30
Extensions added to sever: You cannot revert to an earlier database server from a database server that has had extensions added unless you remove the extensions. You must remove any new data types or routines that you created either explicitly or by registering a different version of a DataBlade module. To be able to revert, downgrade any DataBlade module to the version that was registered before reversion and explicitly drop any data types and routines that were created outside of any DataBlade registration. For information about how to use DataBlade modules, see the DataBlade documentation.	9.30
UDR limitations: New user-defined or SPL routines must not be created in new 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 before reversion.	9.30
New Trigger limitations: New triggers must not be defined in the upgraded databases.	9.30
New fragment expressions and check constraints: New fragment expressions and check constraints must not exist in the databases. To revert, convert fragmented tables to nonfragmented tables by detaching fragment expressions. You cannot use ALTER TABLE or ALTER INDEX statements to change fragment strategies that existed before the migration to Informix 9.40.	9.30
item_nvarchar limitation: Reversion fails if, for an index, the value of item_nvarchar is 255 or higher.	7.31
New log files: If Informix uses a newly added log file, you cannot reset the status of the file to "newly added" after reversion to the earlier database server.	7.31
PER_STMT_EXEC or PER_STMT_PREP in a DataBlade module: A DataBlade module that uses the PER_STMT_EXEC or PER_STMT_PREP memory duration cannot be used with the earlier database server.	7.31
Select triggers: Select triggers must not be in use.	7.31
User-defined statistics: User-defined statistics must not be in use.	7.31

Table 7-2. Requirements and limitations when reverting to a specific version of the server (continued)

Reversion requirement or limitation	If reverting to this server or earlier servers
<p>Long identifiers: Long identifiers or long user names must not be in use.</p> <p>Before reversion, make sure that the R-tree indexes do not use long identifiers as indexed column names, operator class (opclass) names, or operator class function names.</p> <p>Also, make sure that the following disk structures do not use long identifiers:</p> <ul style="list-style-type: none"> • Dbspace tblspaces (owner and database name length) • Tblspace tblspaces (owner and table space name length) • Dbspaces (owner and dbspace name length) and chunks (path length) 	7.31
<p>Storage space names: Each storage space must not have a name that is more than 18 characters long.</p>	7.31
<p>New routine languages: New routine languages must not be defined in the upgraded databases. In addition, new language authorizations must not have been completed in the upgraded databases.</p>	7.31
<p>New operator classes, casts, and extended types: New operator classes, casts, or extended types must not be defined in the new database server.</p>	7.31
<p>Table limitations:</p> <ul style="list-style-type: none"> • Databases cannot have tables whose primary access method is a user-defined access method. • Databases cannot have typed tables. • Tables cannot have any user-defined type columns. • Tables cannot have named row types with default values. 	7.31
<p>Index limitations:</p> <ul style="list-style-type: none"> • You cannot revert detached indexes to Version 7.31. To enable reversion to Version 7.31, retain the Version 7.31 attached index behavior by setting the environment variable DEFAULT_ATTACH in the application environment. • All indexes must be B-tree indexes with a total key length less than or equal to 255. • Tables cannot have any functional or Virtual Index Interface (VII) indexes. • Semi-detached indexes must not be in the databases. • Indexes created with Informix 10.00 and an opclass that supports nearest-neighbor search cannot be reverted to the earlier database server. 	7.31
<p>Extensibility features: 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.</p>	7.31

If you migrated to an interim version of the database server before you migrated to Version 11.70 and you need to revert, see additional reversion requirements and limitations in the *IBM Informix Migration Guide* that is included in the documentation set for the interim version of the database server.

Check and configure available space for reversion

You must be sure you have enough space for reversion to the source database server.

For Informix Versions 11.70, 11.50, 11.10, 10.00, and 9.40, tblspace **tblspace** pages can be allocated in non-root chunks. If the root chunk is full and tblspace **tblspace** pages were allocated in non-root chunks, make sure you have enough space in the root chunk of the target database server.

If the reversion is to a Version 9.30 or to an earlier database server, all tablespace reserved pages are written to the root chunk. The default number of tablespace reserved pages in Version 9.30 is 250 pages (with a size 2k or 4k each, depending on the operating system).

To determine how many pages were allocated and where they were allocated, run **oncheck -pe** and look for the word TBLSpace. This space must be available on the device where the root chunk will be located.

For information about space requirements for Informix Version 11.70, see “Checking and configuring available space” on page 3-2.

Related concepts:

“Checking and configuring available space” on page 3-2

Save copies of the current configuration files

Save copies of the ONCONFIG and **concsm.cfg** files for when you migrate to Informix Version 11.70 again.

Informix uses the **concsm.cfg** file to configure CSMs.

Save system catalog information

If your current database server instance uses secure-auditing masks or external spaces, and you want to preserve the associated catalog information, you must unload these system catalog tables before you revert.

Run the following command to unload the system catalog tables:

```
$INFORMIXDIR/etc/smi_unld
```

When the **smi_unld** utility finishes unloading the information, the utility displays instructions for reloading the information. Save these instructions. After you complete the reversion and bring up your database server, you can reload the data that you preserved. Follow the instructions given with the **smi_unld** utility for reloading the information. Typically, you run the following command:

```
$INFORMIXDIR/etc/smi_load $INFORMIXDIR/etc/
```

Verify the integrity of the Version 11.70 data

Verify the integrity of your Version 11.70 data, if you did not do this after you migrated.

To verify the integrity of your data, run the following commands:

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

If the **oncheck** utility finds any problems, the utility prompts you to respond to corrective action that it can perform. If you respond Yes to the suggested corrective action, run the **oncheck** command again to make sure the problem has been fixed.

The **oncheck** utility cannot fix data that has become corrupt. If the **oncheck** utility is unable to fix a corruption problem, you might need to contact Technical Support before you proceed.

You will also need to verify the integrity of your data after you revert.

Back up Informix Version 11.70

Before you begin the reversion, make a complete level-0 backup of Informix Version 11.70.

For more information, see the *IBM Informix Backup and Restore Guide*.

Run dummy UPDATE statements

Run dummy UPDATE statements on outstanding in-place alters before you revert to a previous version of the database server. An in-place alter is outstanding when data pages still exist with the prior definition.

Run an **oncheck -pT tablename** command to see if you have outstanding in-place alters. If you have outstanding in-place alters, run dummy UPDATE statements on them before you revert to a previous version of the database server.

Dummy UPDATE statements force any outstanding in-place alters to complete by updating the rows in the affected tables. To generate a dummy UPDATE statement, create an UPDATE statement in which a column in the table is set to its own value. This forces the row to be updated to the latest schema without actually changing column values. Because the database server always alters rows to the latest schema, a single pass through the table that updates all rows completes all outstanding in-place alters.

The dummy UPDATE statement differs from a standard UPDATE statement because it does not change the data. A standard UPDATE statement usually changes the value of the affected row.

For example, to create a dummy update, specify:

```
UPDATE tabl SET col1=col1 WHERE 1=1 ;
```

You must ensure that the column selected is a numeric data type (for example, INTEGER or SMALLINT) and not a character data type.

If a table is large, a single update of the whole table can cause a long transaction. To avoid a long transaction, update the table in pieces, by ranges of some column, with this statement:

```
... WHERE {id_column} BETWEEN {low_value} AND {step_value}
```

For example, specify:

```
UPDATE tabl SET col1=col1 WHERE col1 BETWEEN 1 AND 100;  
UPDATE tabl SET col1=col1 WHERE col1 BETWEEN 101 AND 200;
```

Ensure that the UPDATE statements include the entire data set.

If the table is replicated with Enterprise Replication, the database server will replicate all updated rows unnecessarily. To avoid replication, update the table as follows:

```
BEGIN WORK WITHOUT REPLICATION;  
...  
COMMIT WORK;
```

When all the pending in-place alters are removed, run the **oncheck -pT tablename** command. In the output of the command, check information in the "Version"

section. The number of data pages should match with current version. Also, all other table versions should have count=0 for the number of data pages that the version is accessing.

For example, if you run **oncheck -pT testdb:tab1** after a dummy update is complete, you might see information similar to the information in this segment of sample output:

TBLspace Report for testdb:root.tab1

```

Physical Address          1:860
Creation date            06/23/2011 14:23:08
TBLspace Flags          800801    Page Locking
                                TBLspace use 4 bit bit-maps

Maximum row size        29
Number of special columns 0
Number of keys          0
Number of extents      1
Current serial value    1
Current SERIAL8 value  1
Current BIGSERIAL value 1
Current REFID value    1
Pagesize (k)           2
First extent size      8
Next extent size       8
Number of pages allocated 8
Number of pages used   4
Number of data pages   3
  << Number of data pages used is 3 >>
Number of rows         6
Partition partnum     1048981
Partition lockid      1048981

```

```

Extents
  Logical Page    Physical Page    Size Physical Pages
              0              1:1895      8              8

```

TBLspace Usage Report for testdb:root.tab1

Type	Pages	Empty	Semi-Full	Full	Very-Full
Free	4				
Bit-Map	1				
Index	0				
Data (Home)	3				
Total Pages	8				

Unused Space Summary

```

Unused data slots          177

```

Home Data Page Version Summary

Version	Count
3 (oldest)	0
4	0
5 (current)	3

<< Other version should show data page count=0>>
 << Other version should show data page count=0>>
 << Current should always match the number of data pages>>

Remove unsupported features

Before you revert, remove all features that your older database server does not support.

Important: Do not remove objects that you did not create, such as the boot scripts (boot90.sql and boot901.sql) in the system catalog because the reversion utility uses them.

For a list of features that you must remove before reversion, see “Ascertain that reversion is possible and identify reversion requirements” on page 7-2.

Remove new BladeManager extensions

When BladeManager or SQL Registration are used to register an extension in a database, the ifxmngt DataBlade module, which manages extensions, is registered first. If you need to revert from Version 11.70, and you ran BladeManager against a database, you must remove all BladeManager extensions.

To remove the BladeManager extensions, you must use BladeManager to unregister all DataBlade modules and then run the following BladeManager command:

```
unprep database_name
```

Reverting from Informix Version 11.70

After preparing to revert, run the reversion utility and prepare to use the original database server.

Prerequisite: Complete the steps in “Preparing to revert” on page 7-1. Preparation includes determining if reversion is possible and preparing your database for reversion.

Review and complete all tasks that apply:

1. “Run the reversion utility” on page 7-14.
2. “Restore original configuration parameters” on page 7-14.
3. “Restore original environment variables” on page 7-14.
4. “Remove any Communications Support Module settings” on page 7-15.
5. “Recompile any Java UDRs that were compiled using JDK 5.0” on page 7-15.
6. “Reinstall and start the earlier database server” on page 7-15.
7. “Optionally update statistics on your tables after reverting” on page 7-15.
8. “Verify the integrity of the reverted data” on page 7-16.
9. “Back up the database server after reversion” on page 7-16.
10. “Return the database server to online mode” on page 7-16.
11. If you use high-availability clusters, perform additional tasks that are described in “Reverting clusters” on page 5-10.

If you are reverting to a version that is earlier than Version 11.10, the database server automatically drops the **sysadmin** database.

Attention: 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.

Related tasks:

“Preparing to revert” on page 7-1

Run the reversion utility

After preparing to revert, run the reversion utility by using an **onmode -b** command.

Important: You must revert to the version of Informix that was your source database before you migrated. If you revert to a different version of the server, you will corrupt data.

Informix Version 11.70 must be running when you run the reversion utility. If the reversion utility detects and lists any remaining features that are specific to Informix Version 11.70, you must remove those features before reversion can complete.

To see a list of all of the versions to which you can revert using an **onmode -b** command, type **onmode -b**.

To run the reversion utility, type

```
onmode -b version_number
```

For examples of the **onmode -b** command, see “Syntax of the **onmode -b** command” on page 14-1.

When you revert to the older version, Informix displays messages that tell you when reversion begins and ends.

When the reversion is complete, Informix is offline. The reversion utility drops the Informix Version 11.70 system catalog tables and restores compatibility so that you can access the data with the earlier version of the database server. The reversion utility does not revert changes made to the layout of the data that do not affect compatibility.


Related concepts:

“Preparation for using the **onmode -b** command” on page 14-1

“What the **onmode -b** command does” on page 14-1

Related reference:

“Syntax of the **onmode -b** command” on page 14-1

 Conversion and reversion error messages (Administrator's Reference)

Restore original configuration parameters

Replace the Informix Version 11.70 ONCONFIG configuration file with the ONCONFIG file that you saved before you migrated. 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 Chapter 17, “Configuration parameter changes by version,” on page 17-1.

Restore original environment variables

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

Remove any Communications Support Module settings

If your Informix Version 11.70 instance used CSMs, edit the `sqlhosts` file to remove any `csm` option settings that are not supported in the older database server.

If you do not do this, the older database server will return an invalid `sqlhosts` options error.

You must also delete the `concsm.cfg` file if the older database server does not support CSMs.

Recompile any Java UDRs that were compiled using JDK 5.0

After you revert and before you start the earlier sever, recompile JAVA UDRs that were compiled using Version 5.0 of the Java Development Kit (JDK) with a JDK version that is earlier than or equal to the version included with the earlier server.

What you do depends on whether your application uses external JAR and class files or JAR files installed on the server:

- If your application uses external JAR and class files (for example, JAR and class files that are listed in `JVPCLASSPATH`), just recompile the files.
- If your application uses JAR files installed in the server (for example, through 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.

Reinstall and start the earlier database server

Reinstall and configure the earlier version of the database server.

Refer to the instructions in your *IBM Informix Installation Guide* and your *IBM Informix Administrator's Guide*.

Run the `oninit -s` command to start the earlier database server in quiescent mode.

Important: Do **not** use the `oninit -i` command.

Optionally update statistics on your tables after reverting

Optionally run `UPDATE STATISTICS` on your tables (not system catalog tables) and on UDRs that perform queries, if you have performance problems after reverting to the previous version of the database server or to a database server on a different operating system.

An unqualified `UPDATE STATISTICS` statement that does not specify a table and column scope clause and a resolution clause updates all tables and all UDRs that are written in SPL.

You do not need to run `UPDATE STATISTICS` statements on C or Java UDRs.

Update statistics on some system catalog tables after reverting

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

For reversion to a 7.31 database server from Informix Version 11.70, run `UPDATE STATISTICS` on the following system catalog tables in the 7.31 database server:

sysblobs
syscolauth
syscolumns
sysconstraints
sysdefaults
sysdistrib
sysfragauth
sysfragments
sysindices
sysobjstate
sysopclstr
sysprocauth
sysproceduresysroleauth
syssynonyms
sysstable
systabauth
systables
systriggers
sysusers

Verify the integrity of the reverted data

Before you allow users to access the databases, check the integrity of the reverted data.

Follow the steps in “Verifying the integrity of the data” on page 3-7.

Back up the database server after reversion

After you complete the reversion, use ON-Bar or **ontape** to make a level-0 backup of the database server to which you reverted.

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 database server to online mode

To bring the old database server online, run the **onmode -m** command.

Then users can access the data.

Register DataBlade modules

You must register any DataBlade modules that your databases require. Built-in database extensions automatically revert to the version included in the database server to which you reverted.

If you are reverting to IBM Informix version 11.70.xC1 or earlier and using the basic text search extension, you must run the following statement in each applicable database before you register or use basic text searching:

```
UPDATE sysblobjects SET drop_can_fail = "t" WHERE bld_id LIKE 'bts.%';
```

If you are reverting to version 11.70.xC1 or later, you do not need to register built-in database extensions because they are registered automatically.

For information on registering DataBlade modules and database extensions, see the *IBM Informix DataBlade Module Installation and Registration Guide*.

Related concepts:

 [DataBlade Module Installation and Registration Guide \(DataBlade Module Installation Guide\)](#)

Part 3. Migration of data between database servers

Chapter 8. Migrating database servers to a new operating system

When you migrate to a new operating system, you must choose a tool for migrating your data, you might need to make some adjustments to your tables, and you must review environment-dependent configuration parameters and environment variables.

Related concepts:

“Paths for migration to the new version” on page 1-6

Choosing a tool for moving data before migrating between operating systems

If you are migrating between different operating systems, you must choose a method for exporting and importing data. The tool 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. The tools that you can use include:

- The **dbexport** and **dbimport** utilities, which you can use to move an entire database
- The UNLOAD and LOAD statements, which move selected columns or tables (The LOAD statement does not change the data format.)
- The **dbload** utility, which you can use to change the data format
- The **onunload** utility, which unloads data in page-sized chunks, and the **onload** utility, which moves data to an identical database server on a computer of the same type
- The High-Performance Loader (HPL), which moves selected columns or tables or an entire database
- Enterprise Replication, which you can use to transfer data between Informix on one operating system and Informix on a second operating system.

For an overview of all of these data-migration tools, a comparison of tools, and information about which versions of the database server do not support all of the tools, see “Data-migration tools” on page 2-1.

Related concepts:

Chapter 9, “The dbexport and dbimport utilities,” on page 9-1

Chapter 13, “The onunload and onload utilities,” on page 13-1

Chapter 10, “The dbload utility,” on page 10-1

Chapter 11, “The dbschema utility,” on page 11-1

Chapter 12, “The LOAD and UNLOAD statements,” on page 12-1

“High-Performance Loader performance advantages for large databases” on page 2-4

Related tasks:

 Moving data with external tables (Administrator's Guide)

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 3 GB table, but your operating system allows only 2 GB files, break up your table into separate files before you migrate. For more information, see your *IBM Informix Administrator's Guide*.

An Informix storage space can reside on an NFS-mounted file system using regular operating-system files. For information about the NFS products you can use to NFS mount a storage space for an Informix database server, check product compatibility information.

Moving data to a database server on a different operating system

You can move 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 or **ontape** to make a final level-0 backup. For more information, refer to your *IBM Informix Backup and Restore 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.

Moving data between Informix and Workgroup Edition Version 7.24 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 Informix and Workgroup Edition Version 7.24 and earlier versions, 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 about UNLOAD, LOAD, **dbload**, and **dbschema**, see Chapter 10, “The dbload utility,” on page 10-1 and Chapter 11, “The dbschema utility,” on page 11-1. For information about how to use DB-Access, see the *IBM Informix DB-Access User’s Guide*.

For more information about using UNLOAD, dbschema, and LOAD to or from Version 7.31 or an earlier version or for information about moving data to or from Version 7.31 on a different operating system, see the Version 7.31 or earlier *Migration Guide*, installation information, and release notes.

Adapting your programs for a different operating system

When you change to a different operating system, you must review and, if necessary, adjust your environment-dependent configuration parameters and environment variables.

Certain database server configuration parameters and environment variables are environment-dependent.

For details, see the information about configuration parameters in the *IBM Informix Administrator’s Guide* and the *IBM Informix Administrator’s Reference* and the information about environment variables in the *IBM Informix Administrator’s Guide* and the *IBM Informix Guide to SQL: Reference*.

Ensuring the successful creation of system databases

The first time the database server is brought online, the **sysmaster**, **sysutils**, **sysuser**, and **sysadmin** databases are built. After moving to a database server on a different operating system, 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*.

Part 4. Data migration utilities

Chapter 9. The **dbexport** and **dbimport** utilities

The **dbexport** and **dbimport** utilities import and export a database and its schema to disk or tape.

The **dbexport** utility unloads an entire database into text files and creates a schema file. You can unload the database and its schema file either to disk or tape. If you prefer, you can unload the schema file to disk and unload the data to tape. You can use the schema file with the **dbimport** utility to re-create the database schema in another IBM Informix environment, and you can edit the schema file to modify the database that **dbimport** creates.

The **dbimport** utility creates a database and loads it with data from text files on tape or disk. 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.

Attention: When you import a database, use the same environment variable settings and configuration parameter settings that were used when the database was created.

- If any environment variables or configuration parameters that affect fragmentation, constraints, triggers, or user-defined routines are set differently than they were when these database objects were created originally, the database that is created by the **dbimport** utility might not be an accurate reproduction of the original.
- Incompatible settings are likely to occur if you move data from an earlier version of the database server to a newer version. Over time, some configuration parameters or environment variables are deprecated, or their default values are changed. For example, assume that attached indexes were created by default in the original database. In the current version of the database server, detached indexes are created by default. If you want to maintain the original behavior, you can set the **DEFAULT_ATTACH** environment variable to 1 before you run the **dbimport** utility.

Requirements or limitations apply in the following cases:

Compressed data

The **dbexport** utility uncompresses compressed data. You must recompress the data after you use the **dbimport** utility to import the data.

Date values

Use four-digit years for date values. The date context for an object includes the date that the object was created, the values of the **DBCENTURY** and **DBDATE** environment variables, and some other environment variables. If the date context during import is not the same as when these objects were created, you might get explicit errors, you might not be able to find your data, or a check constraint might not work as expected. Some of these problems do not generate errors.

Tip: By default, the **dbexport** utility exports dates in four-digit years unless environment variables are set to values that would override that format.

High-availability clusters

You cannot use the **dbexport** utility on HDR secondary servers or shared disk (SD) secondary servers.

The **dbexport** utility is supported on a remote standalone (RS) secondary server only if the server is set to stop applying log files. Use the **STOP_APPLY** configuration parameter to stop application of log files.

The **dbimport** utility is supported on all updatable secondary servers.

Label-based access control (LBAC)

When you export data that is LBAC-protected, the data that is exported is limited to the data that your LBAC credentials allow you to read. If your LBAC credentials do not allow you to read a row, that row is not exported, but no error is returned. To export all the rows, you must be able to see all the rows.

NLSCASE mode

Whether the NLSCASE mode of your source database is SENSITIVE or INSENSITIVE, you can reduce the risk of case-sensitivity issues by always migrating to a target database that has the same NLSCASE mode as the source database. For tables that include columns with case-variant NCHAR and NVARCHAR data values (for example, 'IBM', 'ibm', 'Ibm'), you might encounter the following differences after migration:

- ORDER BY and sorting operations can produce a different ordering of qualifying rows in query results, compared to the result of the same query before migration.
- Unique indexes and referential constraints with which the data were compliant before the migration might have integrity violations in the new database, if any index or constraint key column contains case-variant forms of the same character string.
- Queries with predicates that apply conditional operators to NCHAR or NVARCHAR values might return different results from the same data after migration.

Nondefault database locales

If the database uses a nondefault locale and the **GL_DATETIME** environment variable has a nondefault setting, you must set the **USE_DTEENV** environment variable to the value of 1 so that localized DATETIME values are processed correctly by the **dbexport** and **dbimport** utilities.

SELECT triggers on tables

You must disable SELECT triggers before you export a database with the **dbexport** utility. The **dbexport** utility runs SELECT statements during export. The SELECT statement triggers might modify the database content.

Virtual tables for the Informix MQ extension

The **MQCreateVtiRead()**, **MQCreateVtiReceive()**, and **MCQCreateVtiWrite()** functions create virtual tables, and map them to the appropriate IBM WebSphere® MQ message queue. When the **dbexport** utility unloads data, it removes the messages from WebSphere MQ queues. Before you use the **dbexport** utility, drop any MQ virtual tables. After you

load the database with the **dbimport** utility, you can create the tables in the target database by using the appropriate functions.

Related concepts:

“Paths for migration to the new version” on page 1-6

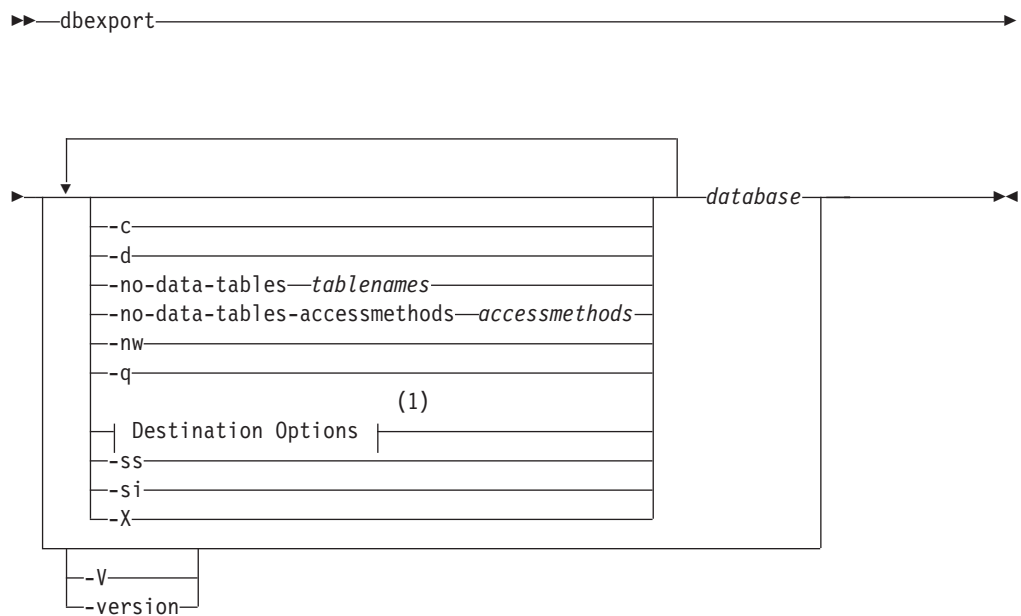
“Choosing a tool for moving data before migrating between operating systems” on page 8-1

Related reference:

“Data-migration tools” on page 2-1

Syntax of the dbexport command

The **dbexport** command unloads a database into text files that you can later import into another database. The command also creates a schema file.



Notes:

1 See “dbexport destination options” on page 9-6

Element	Purpose	Key Considerations
-c	Makes dbexport complete exporting unless a fatal error occurs	References: For details on this option, see “dbexport errors” on page 9-5.
-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> .
-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 “dbexport server-specific information” on page 9-5.

Element	Purpose	Key Considerations
-si	Excludes the generation of index storage clauses for non-fragmented tables The -si option is available only with the -ss option.	References: For details on this option, see “dbexport server-specific information” on page 9-5.
-X	Recognizes HEX binary data in character fields	None.
-no-data-tables	Prevents data from being exported for the specified tables. Only the definitions of the specified tables are exported.	Accepts a comma-separated list of names of tables for which data will not be exported. Default behavior: Only the definition of the tsinstanceTable table is exported, not the data. The data and definitions of all other tables are exported.
-no-data-tables-accessmethods	Prevents data from being unloaded using the specified access methods.	Accepts a comma-separated list of names of access methods. Tables using those access methods are not unloaded. Default value: ts_rts_vtam, ts_vtam Tables using ts_rts_vtam and ts_vtam access methods are not unloaded.
-nw	Generates the SQL for creating a database without the specification of an owner	None.
-V	Displays the software version number and the serial number	None.
-version	Extends the -V option to display additional information about the build operating system, build number, and build date	None.
database	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, see Database Name.

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 9-13.

You can set the IFX_UNLOAD_EILSEQ_MODE environment variable to enable **dbexport** to use character data that is invalid for the locale specified in the environment.

You can use delimited identifiers with the **dbexport** utility. The utility detects database objects that are keywords, mixed case, or have special characters, and the utility 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 standard output unless you specify the **-q** option.

During 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.

Example

The following command exports the table definitions but no data for all the tables in the customer database.

```
dbexport -no-data-tables -no-data-tables-accessmethods customer
```

Example

The following command generates the schema and data for the customer database without the specification of an owner:

```
dbexport customer -nw
```

Related reference:

 [IFX_UNLOAD_EILSEQ_MODE environment variable \(SQL Reference\)](#)

Termination of the dbexport utility

You can stop the **dbexport** utility at any time.

To cancel **dbexport**, press your Interrupt key.

The **dbexport** utility asks for confirmation before it terminates.

dbexport errors

The **dbexport -c** option tells **dbexport** to complete exporting unless a fatal error occurs.

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

- **dbexport** is unable to open the specified tape.
- **dbexport** finds bad writes to the tape or disk.
- Invalid command parameters were used.
- **dbexport** cannot open the database or there is no system permission for doing so.
- A subdirectory with the name specified during invocation already exists

dbexport server-specific information

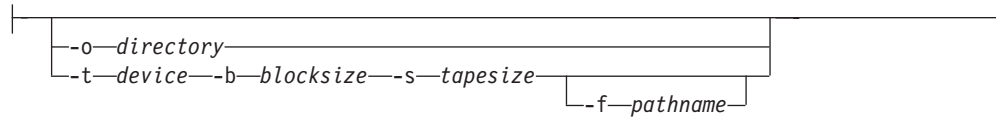
The **dbexport -ss** option generates server-specific information. This option specifies initial- and next-extent sizes, fragmentation information if the table is fragmented, the locking mode, the dbspace for a table, the blobspace for any simple large objects, and the dbspace for any smart large objects.

The **dbexport -si** option, which is available only with the **-ss** option, does not generate index storage clauses for non-fragmented tables.

dbexport destination options

The **dbexport** utility supports disk and tape destination options.

Destination options:



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 name of the path where you want the schema file stored, if you are storing the data files on tape.	The path name can be a complete path name or a file name. If only a file name is given, the file is stored in the current directory. If you do not use the -f option, the SQL source code is written to the tape.
-o <i>directory</i>	Specifies the directory on disk in which dbexport creates the <i>database.exp</i> directory. This directory holds the data files and the schema file that dbexport creates for the database.	The specified directory must exist.
-s <i>tapesize</i>	Specifies, in kilobytes, the amount of data that you can store on the tape.	To write to the end of the tape, set the value to 0. If you do not specify 0, the maximum size is 2 097 151 KB.
-t <i>device</i>	Specifies the path name of the tape device where you want the text files and, possibly, the schema file stored.	You cannot 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 database servers on UNIX or Linux, the command is:

```
dbexport //finland/reports
```

The following command exports the database **stores_demo** to tape with a block size of 16 KB and a tape capacity of 24 000 KB. The command also writes the schema file 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
```

Windows Only

For Windows, the following command exports the database **stores_demo** to tape with a block size of 16 KB and a tape capacity of 24 000 KB. 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
```

Contents of the schema file that dbexport creates

The **dbexport** utility creates a schema file. This 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.

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 blob space 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 runs the **dbimport** command also has DBA privileges for the database.

The schema file that **dbexport** creates contains comments, which are 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.

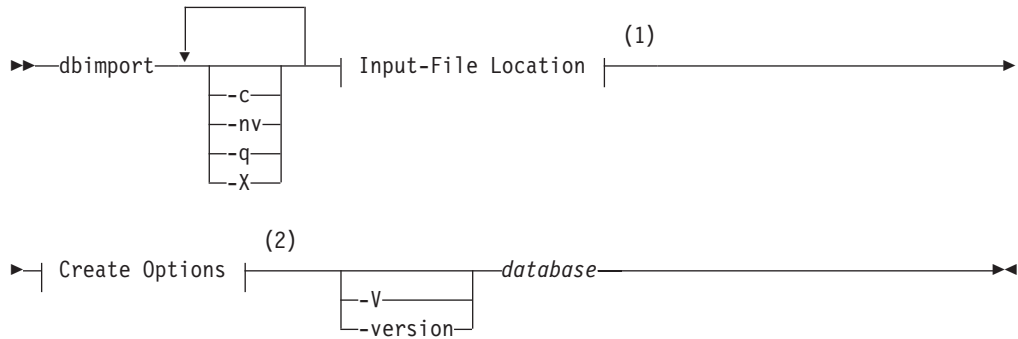
The number of rows must match in the unload file and the corresponding unload comment in the schema file. If you change the number of rows in the unload file but not the number of rows in the schema file, a mismatch occurs.

Attention: Do not delete any comments in the schema file, and do not change any existing comments or add any new comments. If you change or add comments, the **dbimport** utility might stop or produce unpredictable results.

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** is successful.

Syntax of the dbimport command

The **dbimport** command imports previously exported data into another database.



Notes:

- 1 See “dbimport input-file location options” on page 9-10
- 2 See “dbimport create options” on page 9-11

Element	Purpose	Key Considerations
-c	Completes importing data even when certain nonfatal errors occur	For more information, see “dbimport errors and warnings” on page 9-9.
-nv	While the dbimport -nv command is running, tables with foreign-key constraints that ALTER TABLE ADD CONSTRAINT creates in enabled or filtering mode are not checked for violations, as if you had also specified NOVALIDATE	By bypassing the checking of referential constraints, this option can reduce migration time for very large tables that already conform to their foreign-key constraints. The NOVALIDATE mode does not persist after the ALTER TABLE ADD CONSTRAINT statement has completed.
-q	Suppresses the display of error messages, warnings, and generated SQL data-definition statements	None.
-V	Displays the software version number and the serial number	None.
-version	Extends the -V option to display additional information about the build operating system, build number, and build date	None.
-X	Recognizes HEX binary data in character fields	None.
<i>database</i>	Declares the name of the database to create	If you want to use more than the simple name of the database, see Database Name.

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

- All input files are on disk.
- All input files are on tape.
- The schema file is 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 an imported Informix database server:

- Specify the dbspace where the database will reside
- Create an ANSI-compliant database with unbuffered logging
- Create a database that supports explicit transactions (with buffered or unbuffered logging)
- Create an unlogged database
- Create a database with the NLS case-insensitive property for NCHAR and NVARCHAR strings.
- Process all ALTER TABLE ADD CONSTRAINT and SET CONSTRAINTS statements in the .sql file of the exported database that define enabled or filtering referential constraints so that any foreign-key constraints that are not specified as DISABLED are in ENABLED NOVALIDATE or in FILTERING NOVALIDATE mode.

Note: If you specify the **-nv** option, the .sql file of the exported database is not modified, but any foreign-key constraints that ALTER TABLE ADD CONSTRAINT or SET CONSTRAINTS statements enable are processed without checking each row of the table for violations. The ENABLED, or FILTERING WITH ERROR, or FILTERING WITHOUT ERROR constraint mode specifications are implemented instead as the ENABLED NOVALIDATE, or FILTERING WITH ERROR NOVALIDATE or FILTERING WITHOUT ERROR NOVALIDATE modes. After the foreign-key constraints have been enabled without checking for violations, their modes automatically revert to whatever the .sql file specified so that subsequent DML operations on the tables enforce referential integrity.

The user who runs the **dbimport** utility 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 server versions that support GLS.

Termination of the dbimport utility

You can stop the **dbimport** utility at any time.

To cancel the **dbimport** utility, press your Interrupt key .

The **dbimport** utility asks for confirmation before it terminates.

dbimport errors and warnings

The **dbimport -c** option tells the **dbimport** utility to complete exporting unless a fatal error occurs.

If you include the **-c** option in a **dbimport** command, **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

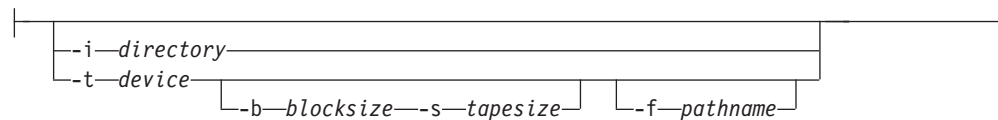
The **dbimport** utility creates a file of messages called **dbimport.out** in the current directory. This file contains any error messages and warnings that are related to **dbimport** processing. The same information is also written to the standard output unless you specify the **-q** option.

dbimport input-file location options

The input-file location specifies the location of the *database.exp* directory, which contains the files that the **dbimport** utility imports.

If you do not specify an input-file location, **dbimport** searches for data files in the directory *database.exp* under the current directory and for the schema file in *database.exp/database.sql*.

dbimport input-file location:



Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies, in kilobytes, the block size of the tape device	If you are importing from tape, you must use the same block size that you used to export the database. If you do not use the -b option, the default block size is 1.
-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	If you use the -f option to export a database, you typically use the same path name that you specified in the dbexport command. If you specify only a file name, dbimport looks for the file in the <i>.exp</i> subdirectory of your current directory. If you do not use the -f option, the SQL source code is written to the tape.
-i <i>directory</i>	Specifies the complete path name on disk of the <i>database.exp</i> directory, which holds the input data files and schema file that dbimport uses to create and load the new database. The directory name must be the same as the database name.	This directory must be the same directory that you specified with the dbexport -o option. If you change the directory name, you also rename your database.

Element	Purpose	Key Considerations
<code>-s tapesize</code>	Specifies, in kilobytes, the amount of data that you can store on the tape	To read to the end of the tape, specify a tape size of 0. If you are importing from tape, you must use the same tape size that you used to export the database. The maximum size is 2 097 151 KB. If you do not use the <code>-s</code> option, the default value is 0 (read to the end of the tape).
<code>-t device</code>	Specifies the path name of the tape device that holds the input files	You cannot specify a remote tape device.

Examples showing input file location on UNIX or Linux

To import the **stores_demo** database from a tape with a block size of 16 KB and a capacity of 24 000 KB, issue this command:

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

The schema file is read from `/tmp/stores_demo.imp`.

To import the **stores_demo** database from the `stores_demo.exp` directory under the `/work/exports` directory, issue this command:

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

The schema file is assumed to be `/work/exports/stores_demo.exp/stores_demo.sql`.

Examples showing input file location on Windows

To import the **stores_demo** database from a tape with a block size of 16 KB and a capacity of 24 000 KB, issue this command:

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

The schema file is read from `C:\temp\stores_demo.imp`.

To import the **stores_demo** database from the `stores_demo.exp` directory under the `D:\work\exports` directory, issue this command:

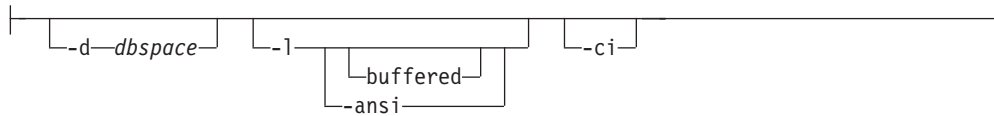
```
dbimport -c -i D:\work\exports stores_demo
```

The schema file is assumed to be `D:\work\exports\stores_demo.exp\stores_demo.sql`.

dbimport create options

The **dbimport** utility supports options for creating a database, specifying a dbspace for that database, defining logging options, and optionally specifying ANSI/ISO-compliance or NLS case-insensitivity (or both) as properties of the database.

Create options:



Element	Purpose	Key Considerations
-ansi	Creates an ANSI/ISO-compliant database in which the ANSI/ISO rules for transaction logging are enabled. Otherwise, the database uses explicit transactions by default.	If you omit the -ansi option, the database uses explicit transactions. Additional Information: For more information about ANSI/ISO-compliant databases, see the <i>IBM Informix Guide to SQL: Reference</i> .
-ci	Specifies the NLS case-insensitive property. Otherwise, the database is case-sensitive by default.	Additional Information: See the <i>IBM Informix Guide to SQL: Syntax</i> and <i>IBM Informix Guide to SQL: Reference</i> descriptions of the NLS case-insensitive property.
-d dbspace	Specifies the dbspace where the database is created. .	If this is omitted, the default location is the root dbspace
-l	Establishes unbuffered transaction logging for the imported database. If the -l flag is omitted, the database is unlogged,	References: For more information, see “Database-logging mode” on page 9-13.
-l buffered	Establishes buffered transaction logging for the imported database. If -l is included but buffered is omitted, the database uses unbuffered logging.	References: For more information, see “Database-logging mode” on page 9-13.

If you created a table or index fragment containing partitions in Informix Version 10.00 or a later version of the Informix database server, 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.

Example showing dbimport create options (UNIX or Linux)

To import the **stores_demo** database from the `/usr/informix/port/stores_demo.exp` directory, issue this command:

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

The new database is ANSI/ISO-compliant.

The next example similarly imports the **stores_demo** database from the `/usr/informix/port/stores_demo.exp` directory. The imported database uses buffered transaction logging and explicit transactions. The **-ci** flag specifies *case insensitivity* in queries and in other operations on columns and character strings of the NCHAR and NVARCHAR data types:

```
dbimport -c stores_demo -i /usr/informix/port -l buffered -ci
```

The **-ansi** and **-ci** options for database properties are not mutually exclusive. You can specify an ANSI/ISO-compliant database that is also NLS case-insensitive, as in the following example of the **dbimport** command:

```
dbimport -c stores_demo -i /usr/informix/port -l -ansi -ci
```

Example showing dbimport create options (Windows)

To import the `stores_demo` database from the `C:\USER\informix\port\stores_demo.exp` directory, issue this command:

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

The imported database is ANSI/ISO-compliant and is case-sensitive for all built-in character data types.

Database-logging mode

Because the logging mode is not retained in the schema file, you can specify logging information when you use the **dbimport** utility to import a database.

You can specify any of the following logging options when you use **dbimport**:

- ANSI-compliant database with unbuffered logging
- Unbuffered logging
- Buffered logging
- No logging

For more information, see “dbimport create options” on page 9-11.

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

- Omitting any of the **-l** options is equivalent to omitting the WITH LOG clause.
- The **-l** option is equivalent to the WITH LOG clause.
- The **-l buffered** option is equivalent to the WITH BUFFERED LOG.
- The **-l -ansi** option is equivalent to the WITH LOG MODE ANSI clause, and implies unbuffered logging.

Related reference:

 CREATE DATABASE statement (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.

You can use the RENAME DATABASE statement to change the database name.

Alternative ways to change the database name

The following examples show alternative ways to change the database name. In this 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 a 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 a 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
```

Changing the database locale with dbimport

You can use the **dbimport** utility 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.

Simple large objects

When the **dbimport**, **dbexport**, and DB-Access utilities process simple-large-object data, they create temporary files for that data in a temporary directory.

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.

Attention: 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.

Chapter 10. The dbload utility

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.

Prerequisites: If the database contains label-based access control (LBAC) objects, the **dbload** utility can load only those rows in which your security label dominates the column-security label or the row-security label. If the entire table is to be loaded, you must have the necessary LBAC credentials for writing all of the labeled rows and columns. For more information about LBAC objects, see the *IBM Informix Security Guide* and the *IBM Informix Guide to SQL: Syntax*.

You cannot use the **dbload** utility on secondary servers in high-availability clusters.

When you use the **dbload** utility, you can manipulate a data file that you are loading or access a database while it is loading. When possible, use the LOAD statement, which is faster than **dbload**.

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.

Related concepts:

“Choosing a tool for moving data before migrating between operating systems” on page 8-1

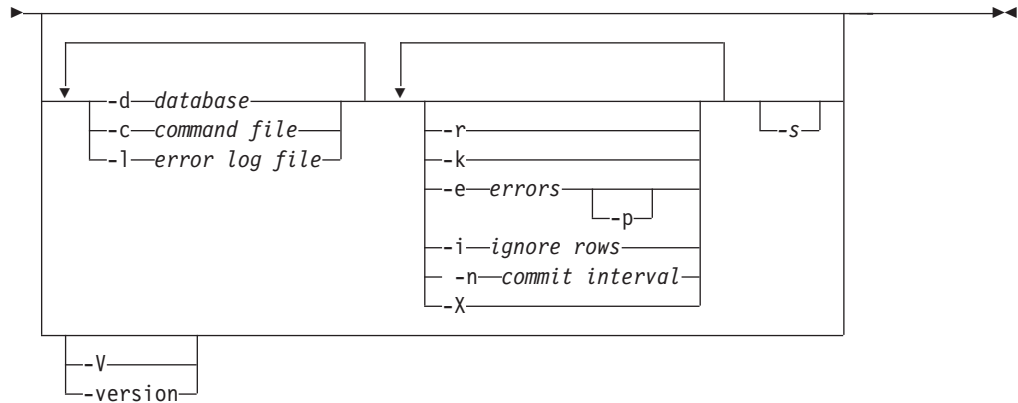
Related reference:

“Data-migration tools” on page 2-1

Syntax of the dbload command

The **dbload** command loads data into databases or tables.

▶—dbload—▶



Element	Purpose	Key Considerations
-c <i>command file</i>	Specifies the file name or path name of a dbload command file	References: For information about building the command file, see “Command file for the dbload utility” on page 10-5.
-d <i>database</i>	Specifies the name of the database to receive the data	Additional Information: To use more than the simple name of the database, see Database Name.
-e <i>errors</i>	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 during a load operation” on page 10-4.
-i <i>ignore rows</i>	Specifies the number of rows to ignore in the input file	References: For more information, see “Rows to ignore during a load operation” on page 10-4.
-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 during a load operation” on page 10-3. 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 <i>error log file</i>	Specifies the file name or path name of an error log file	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 <i>commit interval</i>	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 during a load operation” on page 10-4.

Element	Purpose	Key Considerations
-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 during a load operation.” 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.
-V	Displays the software version number and the serial number	None.
-version	Extends the -V option to display additional information about the build operating system, build number, and build date	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.

dbload command example

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

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

Table locking during a load operation

The **dbload -k** option overrides the default table lock mode during the load operation. The **-k** option instructs **dbload** to lock the tables in exclusive mode rather than shared mode.

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 must 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 need 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.

Rows to ignore during a load operation

The **dbload -i** option specifies the number of new-line characters in the input file that **dbload** ignores before **dbload** begins to process data.

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.

The **-i** option is also useful if header information in the input file precedes the data records.

Bad-row limit during a load operation

The **dbload -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.

Termination of the dbload utility

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

Name and object guidelines for the dbload utility

You must follow guidelines for specifying network names and handling simple large objects, indexes, and delimited identifiers when you use the **dbload** utility.

Table 10-1. Name and object guidelines for the dbload utility

Objects	Guideline
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.
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.

Table 10-1. Name and object guidelines for the dbload utility (continued)

Objects	Guideline
Delimited identifiers	<p>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.</p> <p>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.</p>

Command file for the dbload utility

Before you use the **dbload** utility, 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 must 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.

Delimiter form of the FILE and INSERT statements

The FILE and INSERT statements that define information for the **dbload** utility can appear in a 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**.

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO stock;
FILE customer.unl DELIMITER '|' 10;
INSERT INTO customer;
FILE manufact.unl DELIMITER '|' 3;
INSERT INTO manufact;
```

To see the **.unl** input data files, refer to the directory **\$INFORMIXDIR/demo/prod_name** (UNIX or Linux) or **%INFORMIXDIR%\demo\prod_name** (Windows).

Syntax for the delimiter form

The syntax for the delimiter form specifies the field delimiter, the input file, and the number of fields in each row of data.

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	If the delimiter specified by <i>c</i> 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 <i>c</i> is specified as a square bracket (I) , 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 10-8.

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.

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:

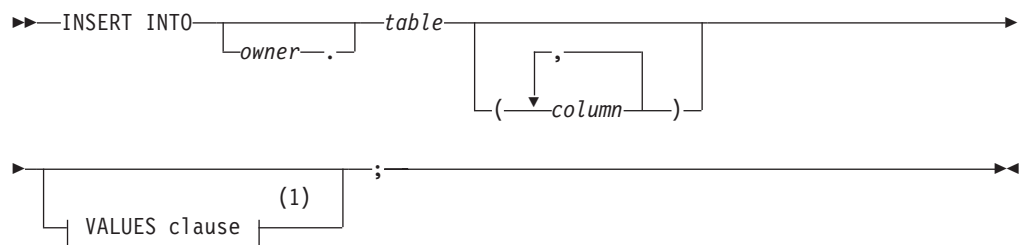
```
FILE "testtbl2.un1" DELIMITER '|' 1;
    INSERT INTO testtbl
        (testuser, testtime, testfield)
    VALUES
        ('kae', CURRENT, f01);
```

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 VALUES clause.

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 run **dbload** with this command file must have the Insert privilege on the named table.

How to write a dbload command file in delimiter form

Command files must contain required elements, including delimiters.

The FILE statement in the following example describes the **stock.unl** data rows as composed of six fields, each separated by a vertical bar (|) as the delimiter.

```
FILE stock.unl DELIMITER '|' 6;  
INSERT INTO stock;
```

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|HRO|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 correspond to the 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 value in the **col4** position (assuming that the column permits null values). 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

Character-position form of the FILE and INSERT statements

The FILE and INSERT statements that define information for the **dbload** utility can appear in a character-position form.

The examples in this topic 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 10-1 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 10-1. 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);
```

```

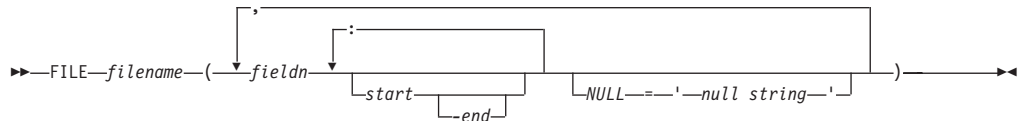
INSERT INTO cust_address (col1, col3, col4)
VALUES (city, state, zip);
INSERT INTO cust_sort
VALUES (area_cd, zip);

```

Syntax for the character-position form

The syntax for the character-position form specifies information that includes the character position within a data row that starts a range of character positions and the character position that ends a range of character positions.

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	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 must substitute a null value	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.

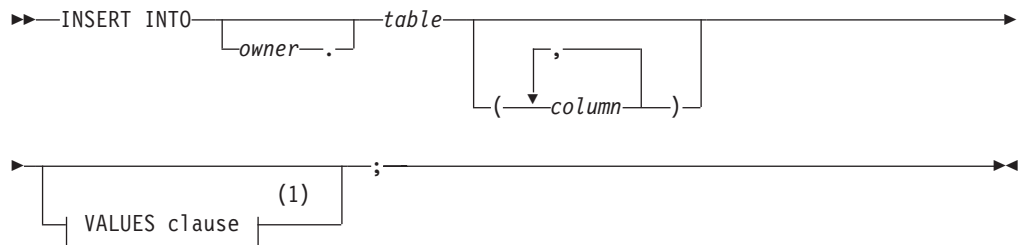
Inserted data types correspond to the explicit or default column list. If the data-field width is different from its corresponding character column, inserted values are padded with blanks if the column is wider, or inserted values are truncated if the field is wider.

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 VALUES clause.

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 runs **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

Command files must define data fields and use character positions to define the length of each field.

The FILE statement in the following example defines six data fields from the **cust_loc_data** table data rows.

```
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 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 the following figure.

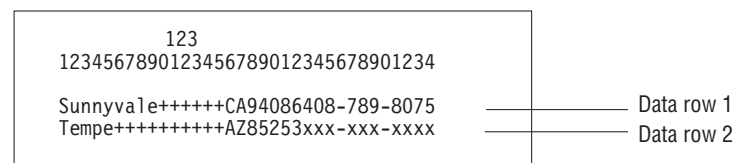


Figure 10-2. A Sample Data File

The FILE statement defines the following data fields, which are derived from the data rows in the sample data file.

Column	Values from Data Row 1	Values from Data Row 2
city	Sunnyvale+++++	Tempe+++++++
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 the sample data file and the FILE statement 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

You can create **dbload** command files that load columns containing complex data types into tables.

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

Important: 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.

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 must be able to load it.

You can use **dbload** with named row types, unnamed row types, sets, and lists.

Using the dbload utility with named row types

The procedure for using the **dbload** utility with named row types is somewhat different than the procedure for using **dbload** with other complex data types, because named row types are actually user-defined data types.

Suppose you have a table named **person** that contains one column with a named row type. Also suppose that 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 (or for any user-defined data 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|1820 Elm Ave #100|Fremont|CA|94502|01/13/1983|
```

2. 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
```

3. Use DB-Access to re-create the **person** table in the new database.

For detailed steps, see “Use dbschema output as DB-Access input” on page 11-15.

4. 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');
```

5. Run 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 the dbload utility with unnamed row types

You can use the **dbload** utility with unnamed row types, which are created with the ROW constructor and define the type of a column or field.

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('Maria','E','Martinez')|ROW('2387 West Wilton  
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('Stephen','M','Wu')",  
        "row('1200 Grand Ave.','Richmond','OR','97200')");
```

Using the dbload utility with collection data types

You can use the **dbload** utility with collection data types such as SET, LIST, and MULTISSET.

SET data type example

The SET data type is an unordered collection type that stores unique elements. The number of elements in a SET data type can vary, but no nulls are allowed.

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

The LIST data type is a collection type that stores ordered, non-unique elements; that is, it allows duplicate element 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.

Chapter 11. The **dbschema** utility

The **dbschema** utility displays the SQL statements (the *schema*) that are necessary to replicate database objects.

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

- To display the distributions that the UPDATE STATISTICS statement creates.
- To display the schema for the Information Schema views
- To display the schema for creating objects such as databases, tables, sequences, synonyms, storage spaces, chunks, logs, roles, and privileges
- To display the distribution information that is stored for one or more tables in the database
- To display information about user-defined data types and row types

After you obtain the schema of a database, you can redirect the **dbschema** output to a file that you can use with DB-Access.

The **dbschema** utility is supported on all updatable secondary servers.

The **dbschema** utility is also supported on read-only secondary servers. However, the **dbschema** utility displays a warning message when running on these servers.

Attention: 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.

Related concepts:

“Choosing a tool for moving data before migrating between operating systems” on page 8-1

Related reference:

“Data-migration tools” on page 2-1

Object modes and violation detection in **dbschema** output

The output from the **dbschema** utility shows object modes and supports violation detection.

The **dbschema** 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 object mode of objects that are in the disabled state. These objects can be constraints, triggers, or indexes.
- The object mode of objects that are in the filtering state. These objects can be constraints or unique indexes.
- 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*.

Guidelines for using the dbschema utility

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

Global Language Support: You must disable SELECT triggers and correctly set GLS environment variables before using the **dbschema** utility.

When the GLS environment variables are set correctly, as the *IBM Informix GLS User's Guide* describes, the **dbschema** utility can handle foreign characters.

Syntax of the dbschema command

The **dbschema** command displays the SQL statements (the *schema*) that are necessary to replicate a specified database object. The command also shows the distributions that the UPDATE STATISTICS statement creates.

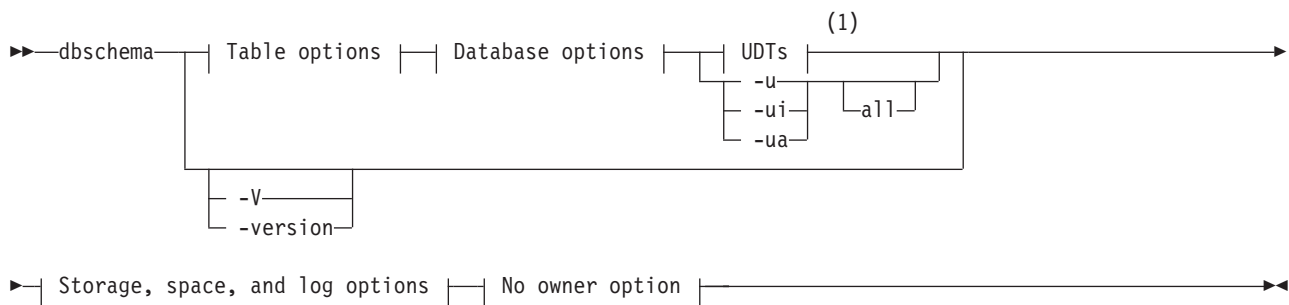
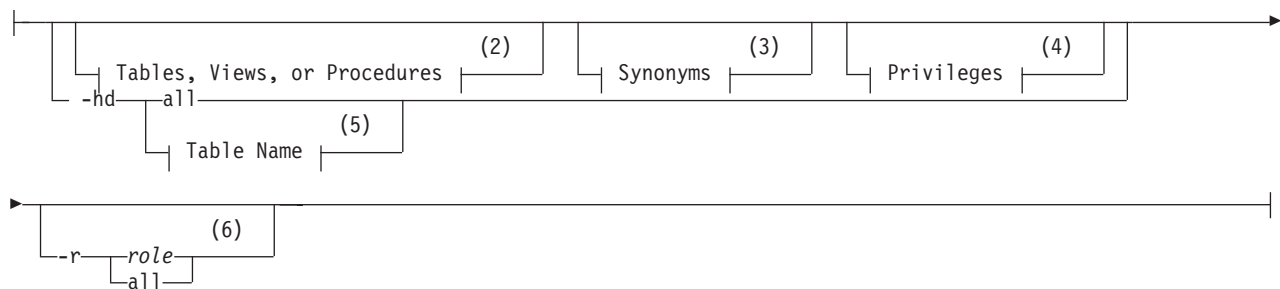
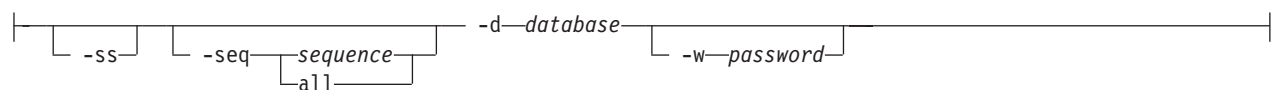


Table options:



Database options:



Storage space and log options:



No owner option:



Notes:

- 1 See “User-defined and complex data types” on page 11-6
- 2 See “Table, view, or procedure creation” on page 11-8.
- 3 See “Synonym creation” on page 11-7.
- 4 See “Privileges” on page 11-12.
- 5 See Identifier.
- 6 See “Role creation” on page 11-11
- 7 See “Storage space, chunk, and log creation” on page 11-9

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.
-c <i>file_name</i>	Generates commands to reproduce storage spaces, chunks, physical logs, and logical logs.	If you use the -c element without the -ns element, the database server generates SQL administration API commands. If you use the -c element and also use the -ns element, the database server generates onspaces or onparams commands.
-d <i>database</i>	Specifies the database to which the schema applies. The <i>database</i> can be on a remote database server.	Additional Information: To use more than the simple name of the database, see Database Name.
<i>filename</i>	Specifies the name of the file that contains the dbschema output	If you omit a file name, dbschema sends the output to the screen. If you specify a file name, dbschema creates a file named <i>filename</i> to contain the dbschema output.
-hd	Displays the distribution as data values	If you specify the ALL keyword for the table name, the distributions for all the tables in the database are displayed.

Element	Purpose	Additional Information
-it	Sets the isolation type for dbschema while dbschema queries catalog tables. Isolation types are: DR = Dirty Read CR = Committed Read CS = Cursor Stability CRU = Committed Read with RETAIN UPDATE LOCKS CSU = Cursor Stability with RETAIN UPDATE LOCKS DRU = Dirty Read with RETAIN UPDATE LOCKS LC = Committed Read, Last Committed RR = Repeatable Read	This option does not display any additional information.
-l	Sets the lock mode to wait <i>number of</i> seconds for dbschema while dbschema queries catalog tables.	This option does not display any additional information.
-ns	Generates onspaces or onparams utility commands to reproduce storage spaces, chunks, physical logs, and logical logs.	The -c element must precede the -ns element in your command.
-nw	Generates the SQL for creating an object without the specification of an owner.	The -nw element is also a dbexport command option.
-q	Suppresses the database version from the header.	This optional element precedes other elements.
-r	Generates information about the creation of roles.	For details, see “Role creation” on page 11-11.
-seq <i>sequence</i>	Generates the DDL statement to define the specified <i>sequence</i> object	None.
-ss	Generates server-specific information	This option is ignored if no table schema is generated.
-si	Excludes the generation of index storage clauses for non-fragmented tables	This option is available only with the -ss option.
-sl <i>length</i>	Specifies the maximum length (in bytes) of unformatted CREATE TABLE and ALTER TABLE statements.	Note
-u	Prints the definitions of functions, casts, and user-defined data types	Specify -u all to include all the tables in the list of distributions.
-ua	Prints the definitions of user-defined data types, including all functions and types defined over a data type.	None.
-ui	Prints the definitions of user-defined data types, including type inheritance	None.
-V	Displays the software version number and the serial number	None.
-version	Extends the -V option to display additional information about the build version, host, operating system, build number and date, and the GLS version.	None.
-w <i>password</i>	Specifies the database password, if you have one.	

You must be the DBA or have the Connect or Resource privilege for the database before you can run **dbschema** on it.

Example

The following command generates the schema with all the tables or sequence objects in the customer database, but without the specification of an owner:

```
dbschema -d customer all -nw
```

Database schema creation

You can create the schema for an entire database or for a portion of the database.

Use the **dbschema** utility options 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.
- 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.

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

The **dbschema -d** option creates and displays the schema for databases on a UNIX or Linux network.

You can specify a database on any accessible non-SE Informix database server.

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

You can edit **dbschema** output to change the owner of a new 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 path name to a file in which compile-time warnings are stored. This path name 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*.

dbschema server-specific information

The **dbschema -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 **-si option**, which is available only with the **-ss** option, excludes the generation of index storage clauses for non-fragmented tables.

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

The **dbschema -u** option displays the definitions of any user-defined and complex data types that the database contains. The suboption **i** adds the type inheritance to the information that the **dbschema -u** option displays.

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 ran 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** that ran 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

The **dbschema -seq** *sequence* command generates information about sequence creation.

The following syntax diagram fragment shows sequence creation.



Element	Purpose	Key Considerations
-seq <i>sequence</i>	Displays the CREATE SEQUENCE statement defining <i>sequence</i>	None.
-seq all	Displays all CREATE SEQUENCE statements for the database	None.

Running **dbschema** with option **-seq** *sequitur* might produce this output:

```
CREATE SEQUENCE sequitur INCREMENT 10 START 100 NOCACHE CYCLE
```

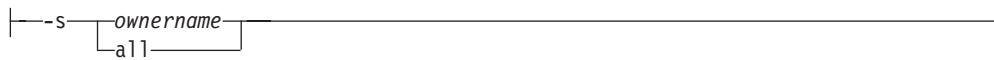
For more information about the CREATE SEQUENCE statement, see the *IBM Informix Guide to SQL: Syntax*.

Synonym creation

The **dbschema -s** command generates information about synonym creation.

The following syntax diagram fragment shows the creation of synonyms.

Synonyms:



Element	Purpose	Key Considerations
-s <i>ownername</i>	Displays the CREATE SYNONYM statements owned by <i>ownername</i>	None.
-s all	Displays all CREATE SYNONYM statements for the database, table, or view specified	None.

Output from **dbschema** that ran with the specified option **-s alice** might appear as the following example shows:

```
CREATE SYNONYM 'alice'.cust FOR 'alice'.customer
```

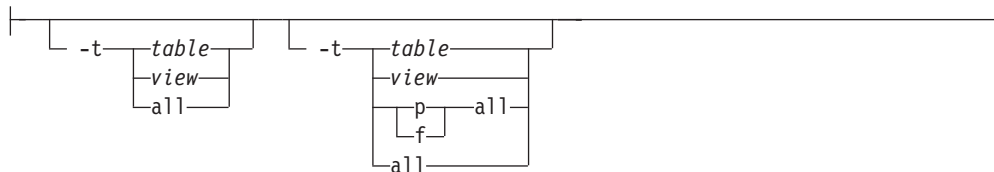
For more information about the CREATE SYNONYM statement, see the *IBM Informix Guide to SQL: Syntax*.

Table, view, or procedure creation

Several **dbschema** options generate information that shows the creation of tables, views, and procedures.

The following syntax diagram shows the creation of tables, views, and procedures.

Tables, Views, or Procedures::



Element	Purpose	Key Considerations
-f all	Limits the SQL statement output to those statements that replicate all functions and procedures	None.
-f function	Limits the SQL statement output to only those statements that replicate the specified function	None.
-f procedure	Limits the SQL statement output to only those statements that replicate the specified procedure	None.
-ff all	Limits the SQL statement output to those statements that replicate all functions	None.
-fp all	Limits the SQL statement output to those statements that replicate all procedures	None.
-t table	Limits the SQL statement output to only those statements that replicate the specified table	None.
-t view	Limits the SQL statement output to only those statements that replicate the specified view	None.
-t all	Includes in the SQL statement output all statements that replicate all tables and views	None.

For more information about the CREATE PROCEDURE and CREATE FUNCTION statements, see the *IBM Informix Guide to SQL: Syntax*.

Table information

The **dbschema -ss** command retrieves information about fragmented tables, the lock mode, and extent sizes.

The following **dbschema** output shows the expressions specified for fragmented table.

```
{ 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 about partitions in partition-fragmented tables.

```
DBSCHEMA Schema Utility                                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;
```

Storage space, chunk, and log creation

The **dbschema -c** command generates SQL administration API commands for reproducing storage spaces, chunks, logical logs, and physical logs. If you use the **dbschema -c -ns** command, the database server generates **onspaces** or **onparams** utility commands for reproducing storage spaces, chunks, physical logs, and logical logs.

For example:

- Run the following command to generate a file named `dbschema1.out` that contains the commands for reproducing the storage spaces, chunks, physical logs, and logical logs in SQL Admin API format:
`dbschema -c dbschema1.out`
- Run the following command to generate a file named `dbschema2.out` that contains the commands for reproducing the storage spaces, chunks, physical logs, and logical logs in **onspaces** and **onparams** utility format:
`dbschema -c -ns dbschema2.out`

Optionally, specify **-q** before you specify **-c** or **-c -ns** to suppress the database version when you run the command. For example, specify:

```
dbschema -q -c -ns dbschema3.out
```

Sample output for the creation of storage spaces, chunks, and logs

The output of the **dbschema -c** or **dbschema -c -ns** commands contain all of the SQL administration API or **onspaces** and **onparams** utility commands that you can use to reproduce storage spaces, chunks, and logs.

Example of output in SQL administration API format

```
# Dbspace 1 -- Chunk 1
EXECUTE FUNCTION TASK ('create dbspace', 'rootdbs',
  '/export/home/informix/data/rootdbs1150fc4', '200000',
  '0', '2', '500', '100')

# Dbspace 2 -- Chunk 2
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs1',
  '/export/home/informix/data/datadbs1150fc4', '5000000',
  '0', '2', '100', '100')

# Dbspace 3 -- Chunk 3
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs2',
  '/export/home/informix/data/datadbs2150fc4', '5000000',
  '0', '2', '100', '100')

# Dbspace 4 -- Chunk 4
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs3',
  '/export/home/informix/data/datadbs3_1150fc4', '80000',
  '16', '8', '400', '400')
EXECUTE FUNCTION TASK ('start mirror', 'datadbs3',
  '/export/home/informix/data/datadbs3_1150fc4', '80000',
  '16', '/export/home/informix/data/mdatadbs3_1150fc4', '16')

# Dbspace 5 -- Chunk 5
EXECUTE FUNCTION TASK ('create tempdbspace', 'tempdbs',
  '/export/home/informix/data/tempdbs_1150fc4', '1000',
  '0', '2', '100', '100')

# Dbspace 6 -- Chunk 6
EXECUTE FUNCTION TASK ('create sbspace', 'sbspace',
  '/export/home/informix/data/sbspace_1150fc4',
  '1000', '0')

# Dbspace 6 -- Chunk 7
EXECUTE FUNCTION TASK ('add chunk', 'sbspace',
  '/export/home/informix/data/sbspace_1_1150fc4',
  '1000', '0')

# Dbspace 7 -- Chunk 8
EXECUTE FUNCTION TASK ('create blobdbspace', 'blobdbs',
  '/export/home/informix/data/blobdbs_1150fc4',
  '1000', '0', '4')

# External Space 1
EXECUTE FUNCTION TASK ('create extspace', 'extspace',
  '/export/home/informix/data/extspac_1150fc4')

# Physical Log
EXECUTE FUNCTION TASK ('alter plog', 'rootdbs', '60000')

# Logical Log 1
EXECUTE FUNCTION TASK ('add log', 'rootdbs', '10000')
```

Example of output in onspaces and onparams utility format

```
# Dbspace 1 -- Chunk 1
onspaces -c -d rootdbs -k 2 -p
/export/home/informix/data/rootdbs1150fc4
-o 0 -s 200000 -en 500 -ef 100

# Dbspace 2 -- Chunk 2
onspaces -c -d datadbs1 -k 2 -p
/export/home/informix/data/datadbs1150fc4
-o 0 -s 5000000 -en 100 -ef 100

# Dbspace 3 -- Chunk 3
onspaces -c -d datadbs2 -k 2 -p
/export/home/informix/data/datadbs2150fc4
-o 0 -s 5000000 -en 100 -ef 100

Dbspace 4 -- Chunk 4
onspaces -c -d datadbs3 -k 8
-p /export/home/informix/data/datadbs3_1150fc4
-o 16 -s 80000 -en 400 -ef 400
-m /export/home/informix/data/mdatadbs3_1150fc4 16

# Dbspace 5 -- Chunk 5
onspaces -c -d tempdbs -k 2 -t -p
/export/home/informix/data/tempdbs_1150fc4 -o 0 -s 1000

# Dbspace 6 -- Chunk 6
onspaces -c -S sbspace -p
/export/home/informix/data/sbspace_1150fc4
-o 0 -s 1000 -Ms 500

# Dbspace 7 -- Chunk 7
onspaces -c -b blobdbs -g 4 -p
/export/home/informix/data/blobdbs_1150fc4 -o 0 -s 1000

# External Space 1
onspaces -c -x extspace -l
/export/home/informix/data/extspac_1150fc4

# Logical Log 1
onparams -a -d rootdbs -s 10000
```

Role creation

The **dbschema -r** command generates information on the creation of roles.

The following syntax diagram shows the creation of roles.

Roles:



Element	Purpose	Key Considerations
-r role	Displays the CREATE ROLE and GRANT statements that are needed to replicate and grant the specified role.	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.
-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
Software Serial Number RDS#N0000000
create role calen;
```

```
grant calen to cathl with grant option;
grant calen to judith ;
grant calen to sallyc ;
```

Related concepts:

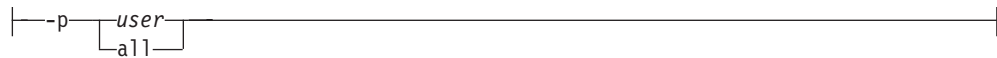
 [User roles \(Security Guide\)](#)

Privileges

The **dbschema -p** command generates information on privileges.

The following syntax diagram fragment shows privileges information.

Privileges:



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	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

You can generate **dbschema** information about the grantor of a GRANT statement.

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

You can generate **dbschema** information about the privileges that were granted for a particular 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.

For example, issue the following **dbschema** command and to display privileges that were granted for the **calen** role.

```
sharky% dbschema -p calen -d stores_demo
```

An example of information the **dbschema** utility displays is:

```
grant alter on table1 to 'calen'
```

Distribution information for tables in dbschema output

The **dbschema -hd** command with the name of the table retrieves the distribution information that is stored for a table in a database. If you specify the ALL keyword for the table name, the distributions for all the tables in the database are displayed.

During the **dbimport** operation, distribution information is created automatically for leading indexes on non-opaque columns. Run the UPDATE STATISTICS statement in MEDIUM or HIGH mode to create distribution information about tables that have the following types of indexes:

- Virtual Index Interface (VII) or function indexes
- Indexes on columns of user-defined data types
- Indexes on columns of built-in opaque data types (such as BOOLEAN or LVARCHAR)

Output from the **dbschema** utility shows distribution information if you used the SAMPLING SIZE keywords when UPDATE STATISTICS in MEDIUM or HIGH mode ran on the table.

For information about using the UPDATE STATISTICS statement, see 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 of dbschema output showing distribution information

The **dbschema** output can show the data distributions that have been created for the specified table and the date when the UPDATE STATISTICS statement that generated the distributions ran.

The follow example of **dbschema** output shows distribution information.

Distribution for cath1.invoices.invoice_num

High Mode, 10.000000 Resolution

--- DISTRIBUTION ---

```
(
1: ( 16, 7, 5)
2: ( 16, 6, 11)
3: ( 16, 8, 17)
4: ( 16, 8, 25)
5: ( 16, 8, 38)
6: ( 16, 7, 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)
}
```

Description of the distribution information in the example

The first part of the sample **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 **dbschema** output also shows the date when the UPDATE STATISTICS statement that generated the distributions ran. You can use this date to tell how outdated your distributions are.

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 runs in HIGH mode with a resolution of 10, the last line appears as the following example shows:

High Mode, 10.000000 Resolution

Distribution information in dbschema output

The distribution information in **dbschema** output 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 in dbschema output

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.

Use dbschema output as DB-Access input

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

Inserting a table into a dbschema output file

You can insert CREATE TABLE statements into the **dbschema** output file and use this output as DB-Access input.

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 12. The LOAD and UNLOAD statements

You can use the SQL LOAD and UNLOAD statements to move data. The LOAD statement is moderately fast and easy to use, but it only accepts specified data formats. You usually use the LOAD statement 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.

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.

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**.

If the database contains label-based access control (LBAC) objects, you can load or unload only those rows in which your security label dominates the column-security label or the row-security label. If entire table is to be loaded or unloaded, you must have the necessary LBAC credentials for writing/reading all of the labeled rows and columns. For more information about LBAC objects, see the *IBM Informix Security Guide* and the *IBM Informix Guide to SQL: Syntax*.

Related concepts:

“Choosing a tool for moving data before migrating between operating systems” on page 8-1

Related reference:

“Data-migration tools” on page 2-1

Syntax of the UNLOAD statement

The UNLOAD statement in DB-Access unloads selected rows from a table into a text file.

```
►► UNLOAD TO 'filename' [ DELIMITER 'delimiter' ] SELECT Statement (1) ►►
```

Notes:

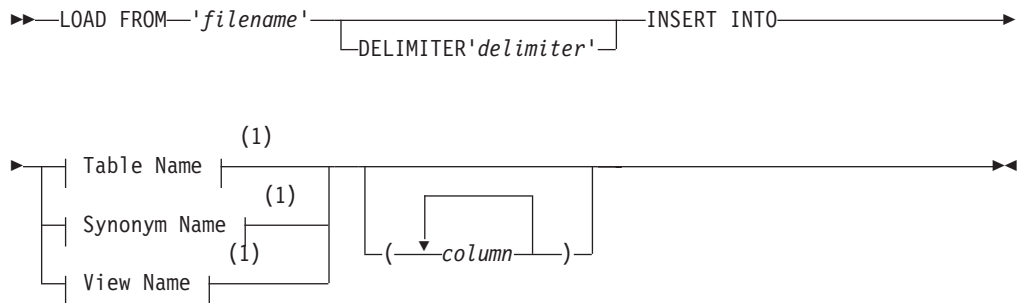
1 See SELECT statement.

Element	Purpose	Key Considerations
<i>delimiter</i>	Character to use as delimiter	Requirements: See “Syntax for the delimiter form” on page 10-6
<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 UNLOAD statement.

Syntax of the LOAD statement

The LOAD statement in DB-Access appends rows to an existing table of a database.



Notes:

- 1 See Identifier.

Element	Purpose	Key Considerations
<i>column</i>	The name of a column to receive data from <i>filename</i>	Must be a column in the specified table or view.
<i>delimiter</i>	Character to use as delimiter	See "Syntax for the delimiter form" on page 10-6
<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 LOAD statement.

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.

Load and unload operations for nondefault locales and GL_DATETIME or DBTIME environment variables

In nondefault locales, operations that load or unload DATETIME or INTERVAL values can be sensitive to the settings of the GL_DATETIME, DBTIME, and USE_DTENV environment variables.

If the database uses a nondefault locale and the GL_DATETIME or DBTIME environment variable has a nondefault setting, you must set the USE_DTENV environment variable to the value of 1 before you can process localized DATETIME or INTERVAL values correctly

- with the LOAD or UNLOAD statements of DB-Access,

- or with the **dbimport** or **dbexport** utilities,
- or in DML operations on objects that the CREATE EXTERNAL TABLE statement defined.

Chapter 13. The **onunload** and **onload** utilities

The **onunload** and **onload** utilities provide the fastest way to move data between computers that use the same database server on the same platform.

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.

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 **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 and do not let you modify the database schema or move from one operating system or database server version to another.

Related concepts:

“Choosing a tool for moving data before migrating between operating systems” on page 8-1

Related reference:

“Data-migration tools” on page 2-1

Guidelines for when to use the **onunload** and **onload** utilities

You can use **onunload** and **onload** only when certain conditions are met.

You can use only **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 only if your answer to each question is yes	My answer
Is the target database server on the same hardware platform?	
Do you want to move to another database server of the same version?	
Do you want to keep the existing database schema without modifying it?	
Do you want to move an entire database or an entire table?	
Are the page images compatible?	
Are the numeric representations the same?	

When you cannot use the **onunload** and **onload** utilities

Because the data is written in page-sized units, 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 2 KB on some UNIX systems and 4 KB on Windows.

Additionally, you cannot use **onunload** and **onload**:

- To move data between GLS and non-GLS databases.
- To move compressed data from one database to another.

You must uncompress data in compressed tables and fragments before you use the **onload** and **onunload** utilities.

- To move external tables or databases that contain external tables.
You must drop all the external tables before you use the **onunload** utility.
- To move tables and databases that contain extended or smart-large-object data types

Requirements for using the **onload** and **onunload** utilities

The **onload** and **onunload** utilities have limitations. You can use these utilities only to move data between database servers of the same version on the same operating system. You cannot modify the database schema, logging must be turned off, and the utilities can be difficult to use.

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

- The original database and the target database must be from the same version of the database server. You cannot use the **onload** and **onunload** utilities to move data from one version to another version.
- You cannot use **onload** and **onunload** to move data between different types of database servers.
- The **onload** command must have the same scope as the corresponding **onunload** command that unloaded the same table or tables that **onload** references. You cannot, for example, use **onunload** to unload an entire database, and then use **onload** to load only a subset of the tables from that database.
- Do not use **onload** and **onunload** to move data if the database contains extended or smart-large-object data types. (Use the HPL instead to move the data.)
- Because the tape that **onload** reads contains binary data that is stored in disk-page-sized units, 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 same page size, the same representation of numeric data, the same byte alignment for structures and unions.
- You cannot use **onload** and **onunload** to move data between non-GLS and GLS locales.
- You cannot use **onload** and **onunload** on servers in high-availability clusters.
- You cannot use **onload** and **onunload** if you compressed tables or fragments.

You can use **onunload** and **onload** to move data between databases if the NLS and GLS locales are identical. For example, if both the NLS and GLS tables were created with the same French locale, **onload** and **onunload** can move data. However, 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** reports errors.

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 as last instead of first, or different float-type representations), the contents of the data page could be misinterpreted.

How the onunload and onload utilities work

The **onunload** utility, which unloads data from a database, writes a database or table into a file on tape or disk. The **onload** utility loads data that was created with the **onunload** command into the database server.

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.

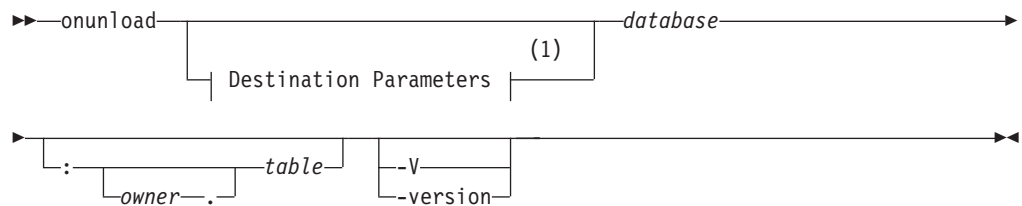
Important: You cannot use the **onload** and **onunload** utilities to move data from one version of a database server to another or between different types of database servers. In addition, the **onload** command must have the same scope as the corresponding **onunload** command that unloaded the same table or tables that **onload** references. You cannot, for example, use **onunload** to unload an entire database, and then use **onload** to load only a subset of the tables from that database.

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

The **onunload** command unloads data from a database and writes a database or table into a file on tape or disk.



Notes:

- 1 See “onunload destination parameters” on page 13-4

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 Identifier.
<i>owner.</i>	Specifies the owner of the table	Additional Information: The owner name must not include invalid characters. References: For path name syntax, see your operating-system documentation.

Element	Purpose	Key Considerations
<i>table</i>	Specifies the name of the table	Requirement: The table must exist. References: Syntax must conform to the Identifier segment; see Identifier.

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 Reference*.)

The **-V** option displays the software version number and the serial number. The **-version** option extends the **-V** option to display additional information about the build operating system, build number, and build date.

onunload destination parameters

The **onunload** utility supports tape or file destination options.

The following syntax diagram fragment shows **onunload** destination parameters

Destination parameters:



Notes:

- 1 Only one occurrence of each option allowed. More than one option can occur in a single invocation.

Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies in kilobytes the block size of the tape device	Requirement: The <i>blocksize</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	Requirement: The <i>tapesize</i> must be an integer. To write to the end of the tape, specify a tape size of 0. If you do not specify 0, then the maximum <i>tapesize</i> is 2 097 151 KB. Additional Information: This option overrides the default value in TAPESIZE or LTAPESIZE.
-t <i>source</i>	Specifies the path name 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. The path name must be a valid path name.

Constraints that affect onunload

When you use the **onunload** utility, you must be aware of constraints that affect how you load the data on the **onunload** tape.

The following constraints apply to **onunload**:

- You must load the data on the **onunload** tape into a database or table that your database server manages.
- You cannot use **onunload** and **onload** if the databases contain 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.

Privileges for 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**.

Tables that are unloaded with a database

If you unload a database, all of the tables in the database, including the system catalog tables, are unloaded.

All triggers, SPL routines, defaults, constraints, and synonyms for all of the tables in the database are also unloaded.

Data that is unloaded with a table

If you unload a table, **onunload** unloads the table data and information from the **systables**, **systables**, **syscolumns**, **sysindexes**, and **sysblobs** system catalog tables.

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.

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.

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*.

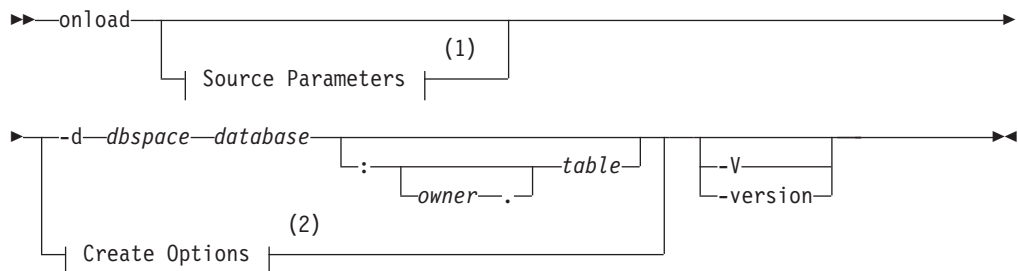
During the load, you can move simple large objects that are stored in a blobspace to another blobspace.

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 Administrator's Guide*.)

If you do not specify creation options, **onload** stores the database or table in the root dbspace.

Syntax of the onload command

The **onload** command loads data that was created with the **onunload** command into the database server.



Notes:

- 1 See "onload source parameters" on page 13-7
- 2 See "onload create options" on page 13-8

Element	Purpose	Key Considerations
-d <i>dbspace</i>	Loads a database or table into the specified dbspace	The tape being loaded must contain the specified database or table.
<i>database</i>	Specifies the name of the database	The database name cannot include a database server name, such as <i>database@dbservername</i> . References: Syntax must conform to the Identifier segment; see Identifier.

Element	Purpose	Key Considerations
<i>owner.</i>	Specifies the owner of the table	The owner name must not include invalid characters. References: For path name syntax, refer to your operating-system documentation.
<i>table</i>	Specifies the name of the table	The table must exist. References: Syntax must conform to the Identifier segment; see Identifier.

The **-V** option displays the software version number and the serial number. The **-version** option extends the **-V** option to display additional information about the build operating system, build number, and build date.

onload source parameters

The **onload** command includes options for specifying information about the tape or file source.

The following syntax diagram fragment shows **onload** source parameters.

Source parameters:



Notes:

- 1 Only one occurrence of each option allowed. More than one option can occur in a single invocation.

Element	Purpose	Key Considerations
-b <i>blocksize</i>	Specifies in kilobytes the block size of the tape device	Requirements: Unsigned integer. Must specify the block size of the tape device. Additional Information: This option overrides the default value in TAPEBLK or LTAPEBLK.
-l	Directs onload to read the values for tape device, block size, and tape size from the configuration parameters LTAPEDEV, LTAPEBLK, and LTAPESIZE, respectively	Additional Information: If you specify -l , and then -b , -s , or -t , the value that you specify overrides the value in the configuration file.
-s <i>tapesize</i>	Specifies in kilobytes the amount of data that the database server can store on the tape	Requirements: Unsigned integer. To write to the end of the tape, specify a tape size of 0. If you do not specify 0, then the maximum <i>tapesize</i> is 2 097 151 KB. Additional Information: This option overrides the default value in TAPESIZE or LTAPESIZE.

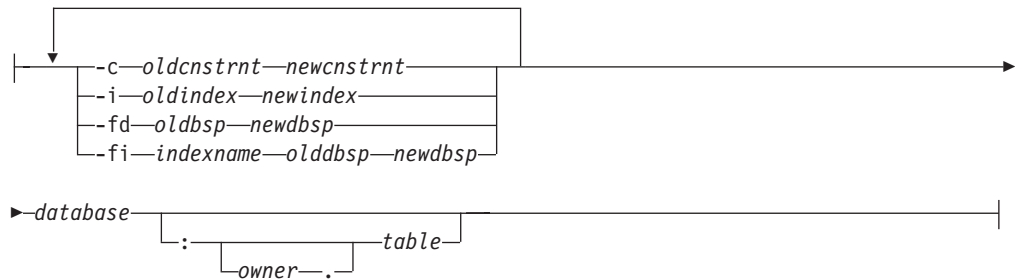
Element	Purpose	Key Considerations
-t source	Specifies the path name of the file on disk or of the tape device where the input tape is mounted	<p>Must be a legitimate path name.</p> <p>Additional Information: This option overrides the tape device that TAPEDEV or LTAPEDEV specifies.</p> <p>References: For path name syntax, see your operating-system documentation.</p>

onload create options

The **onload** command includes information that is used to recreate the database.

The following syntax diagram fragment shows **onload** create options.

Create options:



Element	Purpose	Key Considerations
-c oldcnstrnt newcnstrnt	Directs onload to rename the specified constraint.	None.
-i oldindex newindex	Directs onload to rename the table index when it stores the index on disk.	<p>Additional Information: Use the -i option to rename indexes during the load to avoid conflict with existing index names.</p> <p>References: Syntax must conform to the Identifier segment; see Identifier.</p>
-fd olddbsp newdbsp	Moves a data fragment from one dbspace to another.	<p>The new dbspace must exist and must not already contain another data fragment for the table.</p> <p>Additional Information: This option is used with parallel data query (PDQ) and table fragmentation.</p>
-fi indexname olddbsp newdbsp	Moves index fragments from one dbspace to another.	<p>The new dbspace must exist and must not already contain another index fragment for the table.</p> <p>Additional Information: This option is used with PDQ and table fragmentation.</p>
<i>database</i>	Specifies the name of the database	<p>Requirement: The database name cannot include a database server name, such as <i>database@dbservername</i>.</p> <p>References: Syntax must conform to the Identifier segment; see Identifier.</p>

Element	Purpose	Key Considerations
<i>owner.</i>	Specifies the owner of the table	Requirement: The owner name must not include invalid characters. References: For path name syntax, refer to your operating-system documentation.
<i>table</i>	Specifies the name of the table	Requirement: The table must not exist. References: Syntax must conform to the Identifier segment; see Identifier.

If you do not specify any create options for non-fragmented tables, the **onload** utility stores the database or table in the root dbspace.

For fragmented tables, **onunload** preserves the fragmentation expression for later use by **onload**. Thus an imported table is fragmented in the same way as the original table.

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 certain constraints.

The **onload** utility has the following constraints:

- The **onload** utility only creates 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. Although 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 runs **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.
- For fragmented tables, the dbspace assignment is preserved, unless you override it using the **-fn** option.
- For non-fragmented tables, the **onload** utility attempts to store the table in root dbspace if a target dbspace is not specified with the **-d** option. If storing the table in root dbspace or in the dbspace specified with the **-d** option is not possible due to difference in page sizes, the **onload** utility tries to use a dbspace that has the same dbspace number as the dbspace number of the originally unloaded table. If this dbspace still has a different page size, the load operation will fail.

Logging during loading

When you use the **onload** utility to create tables from an **onunload** input tape, **onload** can load information only into a database without logging. Thus, before you load a table into an existing, logged database, you must 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.

The **onload** utility performs all its loading within a transaction. This feature allows the changes to be rolled back if an error occurs.

Movement of simple large objects to a blob space

If you load a table that contains simple large objects stored in a blob space, the **onload** utility asks you if you want to move them to another blob space.

If you respond yes, **onload** displays the blob space name where the simple large objects were stored when the tape was created. It then asks you to enter the name of the blob space where you want the simple large objects stored.

If you enter a valid blob space name, **onload** moves all simple-large-object columns in the table to the new blob space. Otherwise, **onload** prompts you again for a valid blob space name.

Ownership and privileges

When you load a new database, the user who runs the **onload** utility 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 11, "The dbschema utility," on page 11-1 describes, before you run **onunload**.

Exclusive locking during a load operation

During a load operation, the **onload** utility 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.

Moving a database between computers with the **onunload** and **onload** utilities

You can use the **onunload** and **onload** utilities to move a complete database from one computer to another.

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 2 KB on certain UNIX systems and 4 KB on Windows. 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 the TAPEDEV or LTAPEDEV configuration parameter, or specify another tape device. Make sure that the tape device that you specify is available for **onunload**. However, if you set the TAPEDEV configuration parameter to STDIO,, the **onunload** utility will not be able to unload data.
3. Run the **oncheck** utility to make sure that your database is consistent.
For information about **oncheck**, see your *IBM Informix Administrator's Reference*.
4. Run the **onunload** utility to unload the data from the database.
For details on the syntax of the **onunload** command, see "Syntax of the onunload command" on page 13-3.
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.
For details on the syntax of the **onload** command, see "Syntax of the onload command" on page 13-6.
7. Set the 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. Create a level-0 backup of the new database.

Moving a table between computers with the **onunload** and **onload** utilities

You can use the **onunload** and **onload** utilities to move a table from one computer to another.

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 2 KB on certain UNIX systems and 4 KB on Windows.)

2. Decide where to store the unloaded data.
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.
For details on the syntax of the **onunload** command, see “Syntax of the onunload command” on page 13-3.
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.
For details on the syntax of the **onload** command, see “Syntax of the onload command” on page 13-6.
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 that are not preserved, and synonyms for the table, run the **dbschema** utility or recreate these objects manually.
Constraints such as primary keys or default values are preserved, even for a single table. Foreign keys, access privileges, SPL routines and synonyms are not preserved.

Moving a table between dbspaces with the onunload and onload utilities

You can use the **onunload** and **onload** utilities to move a table from one dspace to another dspace on the same computer.

To move a table from one dspace to another dspace on the same computer:

1. Run the **onunload** utility to unload the table.
For details on the syntax of the **onunload** command, see “Syntax of the onunload command” on page 13-3.
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 dspace name in the **onload** command.
For details on the syntax of the **onload** command, see “Syntax of the onload command” on page 13-6.
4. If the data loads successfully, delete the old table in the old dspace 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.

Chapter 14. The onmode utility reversion option

You use the **-b** option of the **onmode** utility to revert to the older database server from which you converted.

What the onmode -b command does

When you convert a database server, several modifications make the format of the databases incompatible with the older version. The **onmode -b** command modifies data so that the earlier version of the database server can access it.

When you convert a database server, several modifications make the format of the databases incompatible with the older version. The **onmode -b** command modifies data so that the earlier version of the database server can access it. In some cases the format of the databases is compatible between versions and the **onmode -b** command is not needed. Type **onmode -b** to see the usage message for options that are available for your database server.

The utility does not revert changes made to the layout of the data that do not affect compatibility.

You must revert the databases before users can access the data with the earlier database server version.

For information about other **onmode** options, see *The onmode utility in your IBM Informix Administrator's Reference*.

Related concepts:

“Run the reversion utility” on page 7-14

Preparation for using the onmode -b command

Before you use the **onmode -b** command, 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 **onmode -b** command includes an implicit **-yuk** command.

Make sure that the **INFORMIXSERVER** environment variable is set to the correct database server.

UNIX/Linux Only

You must be user **root** or user **informix** to run **onmode**.

Windows Only

You must be a member of the **Informix-Admin** group to run **onmode**.

Related concepts:

“Run the reversion utility” on page 7-14

Syntax of the onmode -b command

The **onmode -b** command restores your databases to the version of Informix from which you converted. You cannot use this command to revert to any other version of the server.

When you convert a database server, several modifications make the format of the databases incompatible with the older version. The **onmode -b** command modifies data so that the earlier version of the database server can access it. In some cases the format of the databases is compatible between versions and the **onmode -b** command is not needed.

Tip: To see a list of all of the versions to which you can revert, run this command:

```
onmode -b
```

When you see the options that are available, choose the option that is closest to the version that you want.

Syntax

```
▶▶ onmode -b version_number ▶▶
```

Element	Purpose
<i>-b version_number</i>	Reverts the database to the specified version.

Related concepts:

“Run the reversion utility” on page 7-14

Chapter 15. The onrestorept utility

If an upgrade to a new version of the server fails, you can use the **onrestorept** utility to restore a server instance back to its original state just before the start of the upgrade. You can run the **onrestorept** utility only if the **CONVERSION_GUARD** configuration parameter was set to 1 or 2 and the upgrade to the new version of the server fails.

However, if you set the **CONVERSION_GUARD** configuration parameter to 1 or 2 and conversion guard operations fail (for example, because the server has insufficient space to store restore point data), and the upgrade to the new version fails, you cannot use the **onrestorept** utility to restore your data.

Before you begin an upgrade:

- Set the **CONVERSION_GUARD** configuration parameter to 1 to enable a restore point as part of the upgrade process, and stop the upgrade if an error related to capturing restore point data occurs.
- Set the value to 2 (the default value) if you want the server to continue the upgrade even if an error related to capturing restore point data occurs, for example, because the server has insufficient space to store the restore point data.

When the **CONVERSION_GUARD** configuration parameter is enabled, the **onrestorept** utility uses the data captured during the upgrade and stored in the directory specified in the **RESTORE_POINT_DIR** configuration parameter. If an upgrade completes successfully, the server automatically deletes restore point files.

The directory specified in the **RESTORE_POINT_DIR** configuration parameter must be empty when an upgrade begins. If the directory contains any restore point files from a previous upgrade, you must remove the files before a new upgrade begins a new restore point.

Important: Informix must be offline when you run the **onrestorept** utility. Do not start the server until the **onrestorept** utility has finished running. Executing the **onrestorept** utility when the server is not offline or starting the server before the **onrestorept** utility has finished running can damage the database, requiring a restore of the database from a backup copy.

You can use the **onrestorept** utility only after a failed upgrade. If you need to revert to the prior version of the server after successfully upgrading to the new version, you must revert using the **onmode -b** command. You should still back up your files before beginning the upgrade in case you need to use the **onmode -b** command to revert to the prior version of the server after a successful migration.

Syntax of the onrestorept command

The **onrestorept** command undoes changes made during a failed upgrade, restoring files to the state they were in when you shut down the server.

```
▶▶ onrestorept [ -v ] [ -c ] [ -y ] ▶▶
```

Element	Purpose	Key Considerations
-V	Display the version of the current server and the software serial number.	
-c	After a failed upgrade, delete the files in the directory specified in the RESTORE_POINT_DIR configuration parameter.	<p>Before you begin another upgrade, you must delete these restore point files. Do this before you make another migration attempt, but not before you run the onrestorept utility to recover the files (if possible).</p> <p>If the upgrade was successful, restore point files are automatically deleted and there is no need to run onrestorept -c.</p>
-y	Automatically display the response y (yes) after every prompt that appears in the information that is displayed while the onrestorept command runs.	If you do not specify -y , you must respond to every prompt. Valid responses are y, Y, n, or N. For example, if you do not specify -y , you can decide whether to proceed with the upgrade whenever the prompt OK to proceed (Y/N) appears.

Examples

The following command restores Informix files after a failed upgrade:

```
onrestorept
```

The following command removes restore point files after a failed upgrade:

```
onrestorept -c
```

Part 5. New and changed features in Informix servers

Each version of the Informix database server contains new features, and new and changed environment variables, configuration parameters, SQL reserved words, system catalogs, and system databases. Some of these changes might affect your applications.

Descriptions of the new, changed, and discontinued features for each version of Informix are available on the web:

- Informix 11.70 http://pic.dhe.ibm.com/infocenter/idshelp/v117/topic/com.ibm.po.doc/new_features.htm
- Informix 11.50 http://pic.dhe.ibm.com/infocenter/idshelp/v115/topic/com.ibm.po.doc/new_features.htm
- Informix 11.10 http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp?topic=/com.ibm.po.doc/new_features.htm.
- Informix 10.00 and earlier <http://publib.boulder.ibm.com/infocenter/idshelp/v10/topic/com.ibm.gsg.doc/gsg51.htm>.

Specific changes that were made from release to release are listed in the following topics.

Chapter 16. Environment variable changes by version

Each version of Informix contains new environment variables that might affect your installation. You might also need to adjust the values of existing environment variables.

For more information about environment variables, see the *IBM Informix Guide to SQL: Reference* and your *IBM Informix Administrator's Guide*.

Table 16-1 lists the new or changed environment variables in various versions of Informix.

Table 16-1. New environment variables

Version	Environment Variable	Description
11.70.xC8	DBTIME	In 11.70.xC7 and earlier release versions, the %F directive inserted by default the ASCII 46 character (.) between the SECOND and FRACTION field values. Starting in 12.10.xC2, however, the %F directive does not include a separator by default.
11.70.xC8	GL_DATETIME	In 11.70.xC7 and earlier release versions, the %F directive inserted by default the ASCII 46 character (.) between the SECOND and FRACTION field values. Starting in 11.70.xC8, however, the %F directive does not include a separator by default.
11.70.xC8	IFX_BAR_NO_BSA_PROVIDER	To be set only at the request of IBM Software Support. Forces ON-Bar to use the sm_versions file as the source of information about the XBSA library for the storage manager.
11.70.xC8	IFX_BAR_NO_LONG_BUFFERS	To be set only at the request of IBM Software Support. Prevents the size of transfer buffers from exceeding 64 KB when the BAR_XFER_BUF_SIZE configuration parameter is set to a long transfer buffer size value.
11.70.xC8	IFX_BAR_USE_DEDUP	Optimizes the deduplication capabilities of storage managers.
11.70.xC8	IFX_TSM_OBJINFO_OFF	Disables support for restoring backup objects that are replicated, imported, or exported between TSM servers.
11.70.xC1	IFX_UNLOAD_EILSEQ_MODE	Enables DB-Access, the dbexport utility, and the High Performance Loader (HPL) to use character data that is invalid for the locale specified in the environment.

Table 16-1. New environment variables (continued)

Version	Environment Variable	Description
11.70.xC1	LOGINTIMEOUT (supported by the client JDBC driver)	Immediately establishes a connection to the Informix database server if the server is running. If the server is not running, this environment variable specifies how long, in milliseconds, the server port is polled to establish a connection.
11.70.xC1	TRUSTED_CONTEXT (supported by the client JDBC driver)	Specifies if a trusted connection can be established between the database server and the client.
11.50.xC4	IFX_LARGE_PAGES (AIX and Solaris)	Enables the use of large pages for non-message shared memory segments that are resident in physical memory. When large pages have been configured by operating system commands and the RESIDENT configuration parameter is set appropriately, this feature can offer significant performance benefits for large memory configurations.
11.50.xC4	IFX_NO_SECURITY_CHECK	Turns off the utilities that check the security of \$INFORMIXDIR when the database server is started. Use this environment variable only when necessary to fix a security flaw in your Informix installation.
11.50.xC4	ONINIT_STDOUT (Windows)	Captures the output of the oninit command on Windows systems.
11.50.xC3	IFX_LOB_XFERSIZE	Provides error checking when transmitting large CLOB or BLOB data types from clients to the database server.
11.50.xC3	CDR_DISABLE_SPOOL	Prevents the generation of ATS and RIS files.
11.50.xC3	CDR_ATSRISNAME_DELIM	Sets the delimiter for the timestamp portion of the ATS and RIS file names.
11.50.xC3	IFX_NOT_STRICT_THOUS_SEP	Removes enforcement of the restriction that three digits must exist after the thousand separator.
11.10	IFX_AUTO_REPREPARE	Controls whether the database server automatically recompiles prepared objects and reoptimizes SPL routines that reference tables whose schemas change Enabling the IFX_AUTO_REPREPARE session environment variable can avoid many -710 errors, and can reduce the number of manual reprepare and reoptimize operations after the schema of a table is modified
11.10	IFX_NODBPROC	An environment variable that enables or prevents the execution of a sysdbopen() or sysdbclose() procedure

Table 16-1. New environment variables (continued)

Version	Environment Variable	Description
10.00.xC4	BAR_SORT_DBS	A variable (used only in Version 10.00.xC4 and later Version 10.00 fix packs) for backup and restore operations when the scope is not the whole system.
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	Supports time-limited license
10.0	IFX_ONPLOAD_AUTO_UPGRADE	Automatically upgrades the onpload database the first time you start the HPL utility with 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 high-availability cluster 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	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.
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 Informix 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.

Related tasks:

“Completing required post-migration tasks” on page 6-7

Chapter 17. Configuration parameter changes by version

Each version of Informix contains new configuration parameters that might affect your installation.

Each version of IBM Informix includes a new `onconfig.std` template file with new default values or other changes for some configuration parameters. In addition, some configuration parameters, such as those used with earlier versions of the database server, might be removed from the database server.

After you upgrade, use the new `onconfig.std` file. You can customize it as necessary to match the configuration of the prior version of your database. Do not use the old `onconfig.std` file with the new version of the server.

If you want to revert to a prior version of the server, you must either replace the Informix Version 11.70 ONCONFIG configuration file with the ONCONFIG file that you used before you converted, or you must remove configuration parameters that the earlier database server does not support.

The following table lists the new configuration parameters in various versions of Informix. All parameters are in the ONCONFIG file, unless otherwise noted. For more information about the configuration parameters, see the *IBM Informix Administrator's Reference* and the *IBM Informix Administrator's Guide*.

Tip:

If a configuration parameter is added to more than one version, the configuration parameter is listed more than once in the following table.

Table 17-1. New configuration parameters

Version	New Configuration Parameter	Description
11.70.xC8	INFORMIXCONRETRY	Specifies the number of connection attempts that can be made to the database server after the initial connection attempt fails. With the INFORMIXCONTIME configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.
11.70.xC8	INFORMIXCONTIME	Specifies the duration, in seconds, that the CONNECT statement attempts to establish a connection to the database server. With the INFORMIXRETRY configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.
11.70.xC8	TLS_VERSION	Specifies the version of the Transport Layer Security (TLS) connection.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.70.xC6	CLUSTER_TXN_SCOPE	If set on the servers in a high-availability cluster, controls whether a transaction commit can be returned to a client application before the transaction has been applied in another server session or on another cluster node
11.70.xC6	HA_FOC_ORDER	If set on a primary server in a high-availability cluster, defines a single failover rule that will be used by all Connection Managers connecting to that primary server.
11.70.xC6	IFX_XA_UNIQUExID_IN_DATABASE	Enables the transaction manager to use same XID to represent global transactions on different databases in the same database server instance.
11.70.xC4	GSKIT_VERSION	Specifies the major version of the IBM Global Security Kit (GSKit) that the database server uses for encryption and SSL communication.
11.70.xC4	SDS_LOGCHECK	Prevents an SD secondary server from taking over the role of the primary server if network communication between the primary and secondary servers is unavailable.
11.70.xC4	S6_USE_REMOTE_SERVER_CFG	Specifies the file used to authenticate secure server connections in a non-trusted network environment. S6_USE_REMOTE_SERVER_CFG is used with the REMOTE_SERVER_CFG configuration parameter.
11.70.xC4	USTLOW_SAMPLE	Enables sampling during the gathering of statistics for UPDATE STATISTICS LOW operations.
11.70.xC3	AUTO_READAHEAD	Changes the automatic read-ahead mode or disables automatic read ahead.
11.70.xC3	LOW_MEMORY_MGR	Enables automatic low memory management, which you can use to change the default behavior of the server when it reaches its memory limit.
11.70.xC3	LOW_MEMORY_RESERVE	Reserves a specific amount of memory for use when critical activities (such as rollback activities) are needed and the server has limited free memory.
11.70.xC3	PN_STAGEBLOB_THRESHOLD	Reserves space for BYTE and TEXT data in round-robin fragments.
11.70.xC2	REMOTE_SERVER_CFG	Specifies the name of a file that lists the remote hosts that are trusted by the computer on which the database server resides.
11.70.xC2	REMOTE_USERS_CFG	Specifies the name of a file that lists names of trusted users that exist on remote hosts.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.70.xC1	AUTO_STAT_MODE	Enables or disables the recalculation of distribution statistics.
11.70.xC1	BATCHEDREAD_INDEX	Enables the optimizer to automatically fetch a set of keys from an index buffer.
11.70.xC1	BAR_CKPTSEC_TIMEOUT	Sets the amount of time, in seconds, that an RS secondary server should wait for a checkpoint to arrive from the primary server while performing an external backup.
11.70.xC1	CDR_LOG_LAG_ACTION	Specifies how Enterprise Replication responds to a potential log wrap situation.
11.70.xC1	CDR_LOG_STAGING_MAXSIZE	Specifies the maximum amount of space that Enterprise Replication uses to stage compressed log files in the directory specified by the LOG_STAGING_DIR configuration parameter.
11.70.xC1	DEFAULTESCCHAR	Specifies the default escape character.
11.70.xC1	ENABLE_SNAPSHOT_COPY	Enables or disables the ability to clone a server using the ifxclone utility.
11.70.xC1	FAILOVER_TX_TIMEOUT	In high-availability cluster environments, enables transaction survival and indicates the maximum number of seconds after failover that the server should wait before rolling back transactions.
11.70.xC1	FULL_DISK_INIT	Specifies whether or not the disk initialization command (oninit -i) can run on your instance when a page zero exists at the root path location.
11.70.xC1	MQCHLLIB	Specifies the path to the directory containing the IBM WebSphere MQ client channel definition table.
11.70.xC1	MQCHLTAB	Specifies the name of the client channel definition table.
11.70.xC1	MQSERVER	Defines a channel, specifies the location of the IBM WebSphere MQ server, and specifies the communication method to be used.
11.70.xC1	NS_CACHE	Defines the maximum retention time for an individual entry in the host name/IP address cache, the service cache, the user cache, and the group cache.
11.70.xC1	NUMFDSERVERS	For network connections on UNIX, specifies the maximum number of poll threads to handle network connections moving between virtual processors (VPs).
11.70.xC1	PRELOAD_DLL_FILE	Specifies the path name for a shared library file that is preloaded when the database server is started.
11.70.xC1	SEQ_CACHE_SIZE	Specifies the maximum number of sequence objects that can have preallocated values in the sequence cache.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.70.xC1	SMX_COMPRESS	Specifies the level of compression that the database server uses before sending data from the source database server to the target database server.
11.70.xC1	SP_AUTOEXPAND	Enable or disables the automatic creation or extension of chunks in a storage space
11.70.xC1	SP_THRESHOLD	Defines the minimum amount of free kilobytes that can exist in a storage space
11.70.xC1	SP_WAITTIME	Specifies the maximum number of seconds that a thread will wait for a storage pool to expand before returning an "out of space" error
11.70.xC1	STATCHANGE	Specifies a percentage of changed rows that triggers the recalculation of distribution statistics.
11.70.xC1	USERMAPPING	Specifies whether or not the database server accepts connections from mapped users.
11.50.xC8	NET_IO_TIMEOUT_ALARM	Controls whether to be notified if network write operations are blocked for 30 minutes or more.
11.50.xC8	RSS_FLOW_CONTROL	Specifies the boundaries within which flow control is enabled in a high-availability cluster that contains at least one RS secondary server.
11.50.xC6	BATCHEDREAD_TABLE	Enables or disables light scans on compressed tables, tables with rows that are larger than a page, and tables with any type of data, including VARCHAR, LVARCHAR, and NVARCHAR data.
11.50.xC6	CONVERSION_GUARD	Enables changes made during an upgrade to a new version of the server and the upgrade fails.
11.50.xC6	RESTORE_POINT_DIR	Specifies the path of the directory where all restore-point files are located if you are undoing changes made during an upgrade that failed. This directory must be empty, and the server offline, before the upgrade begins. The server stores restore point files in a subdirectory of the specified directory, with the server number as the subdirectory name.
11.50.xC6	SB_CHECK_FOR_TEMP	Prevents the copying of a temporary smart large object into a permanent table.
11.50.xC5	DELAY_APPLY	Used to configure RS secondary servers to wait for a specified period of time before applying logs.
11.50.xC5	LOG_STAGING_DIR	Specifies the location of log files received from the primary server when configuring delayed application of log files on RS secondary servers.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.50.xC5	STOP_APPLY	Used to stop an RS secondary server from applying log files received from the primary server.
11.50.xC5	SQL_LOGICAL_CHAR	When enabled, causes size specifications in the declarations of character data types to be interpreted in units of logical characters, rather than as bytes.
11.50.xC4	CHECKALLDOMAINSFORUSER	Configures how Informix searches for user names in a networked Windows environment. Not in the onconfig.std file.
11.50.xC2	LIMITNUMSESSIONS	Defines the maximum number of sessions that you want connected to Informix. If you specify a maximum number, you can also specify whether you want Informix to print messages to the online.log file when the number of sessions approaches the maximum number.
11.50	FAILOVER_CALLBACK	Specifies the full path name of a script that the database server executes when the server transitions from a secondary server to a primary or standard server.
11.50	HA_ALIAS	When a secondary server connects to a primary server, specifies the name of a network alias to use if a failover occurs.
11.50	MSG_DATE	When enabled, adds a date to the front of each message in the online log.
11.50	SHMNOACCESS	Specifies a virtual memory address range to not use to attach shared memory.
11.50	SSL_KEYSTORE_FILE	On clients, specifies the fully qualified file name of the keystore that stores the certificates of all servers to which the client connects. Not in the onconfig.std file.
11.50	SSL_KEYSTORE_LABEL	On Informix, specifies the label of the server digital certificate used in the keystore database that stores Secure Sockets Layer (SSL) keys and digital certificates.
11.50	SSL_KEYSTORE_STH	On clients, specifies the fully qualified file name of the stash file that contains the encrypted keystore password. Not in the onconfig.std file.
11.50	STORAGE_FULL_ALARM	Configures the frequency and severity of messages and alarms when storage spaces become full.
11.50	UPDATABLE_SECONDARY	Enables client applications to perform update, insert, and delete operations on a high-availability secondary server.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.10	ADMIN_MODE_USERS	Enables user informix or a DBSA to give one or more specific users the ability to connect to the database server in administration mode through the onmode -j command, the oninit -U command, or the ADMIN_MODE_USERS configuration parameter.
11.10	ADMIN_USER_MODE_WITH_DBSA	Specifies whether user informix and the DBSA group users can connect to the database server while it is in administration mode.
11.10	AUTO_AIOVPS	Enables or disables the ability of the database server to automatically increase the number of AIO VPs and flusher threads when the server detects that AIO VPs are not keeping up with the I/O workload.
11.10	AUTO_CKPTS	Enables or disables automatic checkpoints when the database server starts.
11.10	AUTO_LRU_TUNING	Enables or disables automatic LRU tuning when the database server starts.
11.10	AUTO_REPREPARE	Controls whether Informix automatically re-optimizes SPL routines and re-prepares prepared objects after the schema of a table referenced by the SPL routine or by the prepared object was significantly changed.
11.10	BACKUP_FILTER	Specifies the path name of a backup filter program and any options that ON-Bar uses.
11.10	BAR_PERFORMANCE	Controls the level of information in the ON-Bar Activity log.
11.10	DIRECT_IO	Controls the use of direct I/O for cooked files used for database space chunks.
11.10	DRDA_COMMBUFFSIZE	Sets the buffer size of the DRDA communications buffer.
11.10	ENCRYPT_HDR	Enables or disables high-availability server encryption.
11.10	ENCRYPT_SMX	Sets the level of encryption for high-availability secondary server configurations.
11.10	EXPLAIN_STAT	Enables or disables the inclusion of a Query Statistics section in the explain.out file that the SET EXPLAIN statement of SQL or the onmode -Y session_id command can display.
11.10	LOG_INDEX_BUILDS	Enables or disables index page logging Index page logging is required when using RS secondary servers.
11.10	MAX_FILL_DATA_PAGES	Enables the database server to insert more rows per page into tables with variable-length rows.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
11.10	PLCY_HASHSIZE	Specifies the number of hash buckets in the cache that holds information about label-based access control (LBAC) credentials for users
11.10	PLCY_POOLSIZE	Specifies the maximum number of entries in each hash bucket of the security policy information cache.
11.10	RESTORE_FILTER	Specifies the path name of a restore filter program and any options that ON-Bar uses.
11.10	RTO_SERVER_RESTART	Sets the amount of time, in seconds, that Informix has to recover from a problem after you restart the server and bring it into online or quiescent mode.
11.10	SDS_ENABLE,	Enables the shared-disk (SD) secondary server function.
11.10	SDS_PAGING	Specifies the location of two files that act as buffer-paging files.
11.10	SDS_TEMPDBS	Specifies information that the SD secondary server uses to dynamically create temporary dbspaces when the SD secondary server starts.
11.10	SDS_TIMEOUT	Specifies the amount of time in seconds that the primary server waits for the SD secondary server to send a log-position acknowledgment.
11.10	SHMVIRT_ALLOCSEG	Specifies a threshold at which Informix allocates server memory, and specifies the alarm level activated if the server cannot allocate the new memory segment.
11.10	SQLTRACE	Controls the default behavior, such as the number of SQL statements to trace and the tracing mode, of the Query Drill-Down feature.
11.10	USELASTCOMMITTED	Specifies whether the database server uses the last committed version of the data when a lock occurs.
11.10	USRC_HASHSIZE	Specifies the number of hash buckets in the cache that holds information about LBAC credentials for users.
11.10	USRC_POOLSIZE	Specifies the maximum number of entries in each hash bucket of the cache that holds information about LBAC credentials for users.
11.10	TEMPTAB_NOLOG	Disables logging on temporary tables.
10.00.xC6	VP_MEMORY_CACHE_KB	Enables a private memory cache that is associated with a CPU virtual processor and contains blocks of free memory.
10.00.xC5	BAR_IXBAR_PATH	Specifies the path and name of the ixbar , the ON-Bar boot file.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
10.00.xC5	FASTPOLL	Enables fast polling of your network, if your operating-system platform supports fast polling.
10.00.xC5	IFX_FOLDVIEW	Enables views to be folded into a parent query.
10.00.xC4	DB_LIBRARY_PATH	Specifies a comma-separated list of valid directory prefix locations from which the database server can load external modules.
10.00.xC4	SECURITY_LOCALCONNECTION	Lets you verify security on local connections by verifying that the ID of the local user who is running a program is the same ID of the user who is trying to access the database.
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.
10.0	DRIDXAUTO	Determines how a secondary database server reacts to a high-availability data-replication failure.
10.0	DS_NONPDQ_QUERY_MEM	Increases 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	IFX_EXTEND_ROLE	Enables a database server administrator (DBSA) to prevent unauthorized users from registering DataBlade modules or external user-defined routines (UDRs).
10.0	LISTEN_TIMEOUT	Sets the incomplete connection timeout period.
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 tablespace tblspace in kilobytes.

Table 17-1. New configuration parameters (continued)

Version	New Configuration Parameter	Description
10.0	TBLTBLNEXT	Specifies the next extent size of tablespace tblspace in kilobytes.
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_CIPHERS	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.

Related concepts:

“Customizing configuration files” on page 6-4

Related tasks:

“Completing required post-migration tasks” on page 6-7

Configuration parameter changes in Version 11.70

Informix Version 11.70 contains some configuration parameter changes, including new default values. In addition, some configuration parameters are now deprecated and thus, no longer used.

Configuration parameters added to the onconfig.std file

The following table lists the configuration parameters that were added to the onconfig.std file in Version 11.70.xC8. For a list of configuration parameters that were added in other 11.70 fix packs, go to *Configuration parameter changes by version*.

Table 17-2. Configuration parameters added to the onconfig.std file

Configuration Parameter	Description
INFORMIXCONRETRY	Specifies the number of connection attempts that can be made to the database server after the initial connection attempt fails. With the INFORMIXCONTIME configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.
INFORMIXCONTIME	Specifies the duration, in seconds, that the CONNECT statement attempts to establish a connection to the database server. With the INFORMIXRETRY configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.
TLS_VERSION	Specifies the version of the Transport Layer Security (TLS) connection.

Configuration parameters that have new default values

The following table lists the configuration parameters that have new default values in the Version 11.70 onconfig.std file.

Table 17-3. Configuration parameters with new default values in the onconfig.std file

Configuration Parameter	Previous Value	New Value
BATCHEDREAD_TABLE	0	1

Configuration parameters that have changed values

The following table lists the configuration parameters that have changed values in Version 11.70.

Table 17-4. Configuration parameters that have changed values

Version	Configuration Parameter	Previous Values	New Values
11.70.xC8	VP_MEMORY_CACHE_KB	The total size of all private memory caches.	The total size of all private memory caches, optionally followed by a comma and the mode (Dynamic or Static) of the caches.
11.70.xC6	ALRM_ALL_EVENTS	1 and 2	0 (on) and 1 (off)

Configuration parameters that are deprecated

The following configuration parameters were deprecated and are not required in IBM Informix Version 11.70:

- AFF_NPROCS
- AFF_SPROC
- BUFFERS
- FAST_RESTART_CKPT_FUZZYLOG
- FAST_RESTART_PHYSLOG

- JVPHOME
- JVPJAVAHOME
- JVPJAVALIB
- JVPJAVAVM
- LRU_MAX_DIRTY
- LRU_MIN_DIRTY
- LRUS
- NOAGE
- NUMAIOVPS
- NUMCPUVPS
- PHYSDBS
- RA_THRESHOLD

Although the previous parameters are still supported, it is recommended that you do not use them. Remove them from the ONCONFIG file before you use the VPCLASS parameter.

The following configuration parameters are no longer supported:

- JDKVERSION
- LBU_PRESERVE
- LOGSMAX

Configuration parameter changes in Version 11.50

The **onconfig.std** file was reorganized for Informix Version 11.50. In addition, some configuration parameters were added to the **onconfig.std** file, some configuration parameters have new default values, and some configuration parameters were deprecated (and thus, no longer used).

In the Version 11.50 **onconfig.std** file, comments and the parameters are listed separately and are grouped by functional areas. Some configuration parameters that specify sizes now have higher values. Some configuration parameters that specify file locations now have more secure default locations under the **\$INFORMIXDIR** directory.

Configuration parameters added to the onconfig.std file

The following table lists the configuration parameters that were added to the **onconfig.std** file in Version 11.50.

Table 17-5. Configuration parameters added to the onconfig.std file

Configuration Parameter	Value
ADMIN_USER_MODE_WITH_DBSA	none
BTSCANNER	num=1,priority=low,threshold=5000, rangesize=1,alice=6,compression=default
BACKUP_FILTER	none
BAR_DEBUG	0
CDR_SUPPRESS_ATSRISWARN	none
DD_HASHMAX	10
DD_HASHSIZE	31

Table 17-5. Configuration parameters added to the onconfig.std file (continued)

Configuration Parameter	Value
DEF_TABLE_LOCKMODE	page
DS_HASHSIZE	31
DS_POOLSIZE	127
ENCRYPT_CDR	none
ENCRYPT_CIPHERS	none
ENCRYPT_HDR	none
ENCRYPT_MAC	none
ENCRYPT_MACFILE	none
ENCRYPT_SMX	none
ENCRYPT_SWITCH	none
EXT_DIRECTIVES	0
FAILOVER_CALLBACK	none
FASTPOLL	1
HA_ALIAS	none
LOG_INDEX_BUILDS	none
MAX_INCOMPLETE_CONNECTIONS	1024
PC_HASHSIZE	31
PC_POOLSIZE	127
PLCY_HASHSIZE	127
PLCY_POOLSIZE	31
PLOG_OVERFLOW_PATH	UNIX: \$INFORMIXDIR/tmp Windows: none
UPDATABLE_SECONDARY	0
RESTORE_FILTER	none
SBSPACETEMP	none
SDS_ENABLE	none
SDS_PAGING	none
SDS_TEMPDBS	none
SDS_TIMEOUT	20
SECURITY_LOCALCONNECTION	none
SQLTRACE	Commented out: # SQLTRACE level=low,ntraces=1000,size=2,mode=global
SSL_KEYSTORE_LABEL	none
STMT_CACHE	0
STMT_CACHE_HITS	0
STMT_CACHE_NOLIMIT	0
STMT_CACHE_NUMPOOL	1
STMT_CACHE_SIZE	512
UNSECURE_ONSTAT	none
USRC_HASHSIZE	31

Table 17-5. Configuration parameters added to the onconfig.std file (continued)

Configuration Parameter	Value
USRC_POOLSIZE	127
VPCLASS	cpu,num=1,noage Commented out: # VPCLASS aio,num=1 Commented out: #VPCLASS jvp,num=1

Configuration parameters that have new default values

The following table lists the configuration parameters that have new default values in the **onconfig.std** file.

Table 17-6. Configuration Parameters with New Default Values in the onconfig.std File

Configuration parameter	Previous value	New value
ADMIN_MODE_USERS	1	None
ALARMPROGRAM	UNIX: /usr/informix/etc/ alarmprogram.sh Windows: None	UNIX: \$INFORMIXDIR/etc/ alarmprogram.sh Windows: \$INFORMIXDIR\etc\ alarmprogram.bat
BAR_ACT_LOG	/usr/informix/bar_act.log	UNIX: \$INFORMIXDIR/tmp/ bar_act.log Windows: \$INFORMIXDIR\tmp\ bar_act.log
BAR_BSALIB_PATH	UNIX: \$INFORMIXDIR/lib/ libsad001.so Window: libbsa.dll	None
BAR_DEBUG_LOG	UNIX: /usr/informix/ bar_dbug.log Windows: bar_dbug.log	UNIX: \$INFORMIXDIR/tmp/ bar_dbug.log Windows: \$INFORMIXDIR\tmp\ bar_dbug.log

Table 17-6. Configuration Parameters with New Default Values in the onconfig.std File (continued)

Configuration parameter	Previous value	New value
BUFFERPOOL	<p>Operating systems with 2K page size:</p> <ul style="list-style-type: none"> • default,buffers=5000,lrus=8, lru_min_dirty=50, lru_max_dirty=60 • size=2k,buffers=5000,lrus=8, lru_min_dirty=50, lru_max_dirty=60 <p>Operating systems with 4K page size:</p> <ul style="list-style-type: none"> • default,buffers=1000,lrus=8, lru_min_dirty=50, lru_max_dirty=60 • size=4k,buffers=1000,lrus=8, lru_min_dirty=50, lru_max_dirty=60 	<p>Operating systems with 2K page size:</p> <ul style="list-style-type: none"> • default,buffers=10000,lrus=8, lru_min_dirty=50.00, lru_max_dirty=60.50 • size=2k,buffers=50000,lrus=8, lru_min_dirty=50, lru_max_dirty=60 <p>Operating systems with 4K page size:</p> <ul style="list-style-type: none"> • default,buffers=10000,lrus=8, lru_min_dirty=50.00, lru_max_dirty=60.50 • size=4k,buffers=10000,lrus=8, lru_min_dirty=50, lru_max_dirty=60
CLEANERS	1	8
CONSOLE	<p>UNIX: /dev/console</p> <p>Windows: console.log</p>	<p>UNIX: \$INFORMIXDIR/tmp/online.con</p> <p>Windows: online.con</p>
DB_LIBRARY_PATH	commented out: # DB_LIBRARY_PATH \$INFORMIXDIR/extend	commented out: # DB_LIBRARY_PATH
DRLOSTFOUND	<p>UNIX: /usr/etc/dr.lostfound</p> <p>Windows: \tmp</p>	<p>UNIX: \$INFORMIXDIR/etc/dr.lostfound</p> <p>Windows: \$INFORMIXDIR\tmp</p>
DUMPDIR	<p>UNIX: /usr/informix/tmp</p> <p>Windows: INFORMIXDIR\tmp</p>	<p>UNIX: \$INFORMIXDIR/tmp</p> <p>Windows: \$INFORMIXDIR\tmp</p>
EXPLAIN_STAT	0	1
LISTEN_TIMEOUT	10	60
LOCKS	2000	20000
LOGBUFF	32	64
LOGSIZE	2000	10000
LTAPEDEV	<p>UNIX: /dev/tapedev</p> <p>Windows: \\.\TAPE1</p>	<p>UNIX: /dev/tapedev (same as previous value)</p> <p>Windows: NUL</p>
MIRRORPATH	None	<p>UNIX: \$INFORMIXDIR/tmp/demo_on.root_mirror</p> <p>Windows: none</p>

Table 17-6. Configuration Parameters with New Default Values in the `onconfig.std` File (continued)

Configuration parameter	Previous value	New value
MSGPATH	UNIX: /usr/informix/ online.log Windows: online.log	UNIX: \$INFORMIXDIR/tmp/ online.log Windows: online.log
NETTYPE	UNIX: none Windows: onsocp,drsocp,1,NET	UNIX: ipcshm,1,50,CPU Windows: none
PHYSBUFF	32	128
PHYSFILE	2000	50000
RA_PAGES	None	64
RA_THRESHOLD	None	16
ROOTPATH	UNIX: /dev/online_root Windows: None	UNIX: \$INFORMIXDIR/tmp/ demo_on.rootdbs Windows: None
ROOTSIZE	30000	200000
SHMVIRT_ALLOCSEG	0	0,3
SHMVIRTSIZE	8192	32656
SYSALARMPROGRAM	UNIX: /usr/informix/etc/ evidence.sh Windows: INFORMIXIDR\etc\ evidence.bat	UNIX: \$INFORMIXDIR/etc/ evidence.sh Windows: Commented out: # SYSALARMPROGRAM \$INFORMIXDIR\etc\ evidence.bat
TAPEBLK	32	UNIX: 32 Windows: 16
TAPESIZE	10240	0

Configuration parameters that were changed

The following table configuration parameters were removed in Version 11.50:

DIRECT_IO

Has new option for concurrent I/O on AIX operating systems.

DUMPSHMEM

Has new options for controlling how much memory is written to a dump file.

JVPJAVAHOME

`/usr/informix` in the directory name of the configuration parameter is replaced with `$INFORMIXDIR`.

The value is now: `$INFORMIXDIR/extend/krakatoa/jre`

JVPPROFILE

`/usr/informix` in the directory name of the configuration parameter is replaced with `$INFORMIXDIR`.

The value is now: \$INFORMIXDIR/extend/krakatoa/.jvpprops

JVPLOGFILE

`/usr/informix` in the directory name of the configuration parameter is replaced with `$INFORMIXDIR`.

The value is now: \$INFORMIXDIR/jvp.log

JVPCLASSPATH

`/usr/informix` in the directory name of the configuration parameter is replaced with `$INFORMIXDIR`.

The value is now: \$INFORMIXDIR/extend/krakatoa/
krakatoa.jar:\$INFORMIXDIR/extend/krakatoa/jdbc.jar

Configuration parameters that were removed

The following configuration parameters were removed from Version 11.50:

- AFF_SPROC (AFF_NPROCS was previously removed.)
- JDKVERSION
- NOAGE
- NUMCPUVPS
- PHYSDBS

Configuration parameter changes in Versions 10.00 through 11.10

Some configuration parameters were changed and some were removed in Version 10.00 and in Version 11.10 of the Informix server.

Table 17-7. Configuration parameters that were changed or removed in version 11.10, 10.0, 9.40, and 9.30

Version	Changed or Removed Configuration Parameter	Description of Change
11.10	FAST_RESTART_CKPT_FUZZYLOG	Removed. The RTO_SERVER_RESTART configuration parameter eliminates fuzzy checkpoints, using interval checkpoints instead.
11.10	FAST_RESTART_PHYSLOG	Removed.
11.10	NOFUZZYCKPT	Removed. The RTO_SERVER_RESTART configuration parameter eliminates fuzzy checkpoints, using interval checkpoints instead.
10.00.xC6	SINGLE_USER_MODE_WITH_DBSA (renamed to ADMIN_USER_MODE_WITH_DBSA in Version 11.10)	Renamed. In Version 11.10, the name of this configuration parameter changed to ADMIN_USER_MODE_WITH_DBSA.
10.0	BUFFERS	Removed. Information now specified with the BUFFERPOOL configuration parameter.
10.0	LRUS	Removed. Information now specified with the BUFFERPOOL configuration parameter.

Table 17-7. Configuration parameters that were changed or removed in version 11.10, 10.0, 9.40, and 9.30 (continued)

Version	Changed or Removed Configuration Parameter	Description of Change
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	New default value.
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.)
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	New default value.
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	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	JVPJAVAHOME	New default location for the JRE.
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	NOAGE	Superseded by the VPCLASS configuration parameter.

Table 17-7. Configuration parameters that were changed or removed in version 11.10, 10.0, 9.40, and 9.30 (continued)

Version	Changed or Removed Configuration Parameter	Description of Change
9.30	NUMAIOVPS	Removed; superseded by the VPCLASS configuration parameter.
9.30	NUMCPUVPS	Superseded by the VPCLASS configuration parameter.

Chapter 18. SQL keyword changes by version

Each version of Informix supports new SQL keywords that might affect the migration of your applications, if an application declares the same string as the identifier of a database object.

Although you can use almost any word as an SQL identifier, syntax ambiguities might produce errors or unexpected results in some contexts if you declare an SQL keyword as an identifier. If your applications use keywords as identifiers, you might consider renaming those identifier. Alternatively, you can enable the DELIMIDENT environment variable, enclose string literal values between single (') quotation marks, and enclose SQL identifiers values between double (") quotation marks.

The following table shows a list of new keywords of SQL in Informix. For a list of all keywords, see Complete list of keywords of SQL in the *IBM Informix Guide to SQL: Syntax*. For details about a keyword, search for the keyword in the *IBM Informix Guide to SQL: Syntax*.

Table 18-1. New keywords of SQL

Informix Version	Keywords
Version 11.70.xC6	CLUSTER_TXN_SCOPE
Version 11.70.xC5	IFX_BATCHEDREAD_INDEX
Version 11.70.xC4	USTLOW_SAMPLE
Version 11.70.xC3	AUTO_READAHEAD
Version 11.70.xC2	AAO ACCOUNT BARGROUP DBSA DBSSO INSENSITIVE NLSCASE SENSITIVE

Table 18-1. New keywords of SQL (continued)

Informix Version	Keywords
Version 11.70.xC1	ADDRESS ATTRIBUTES AUTHENTICATION AUTHID AUTO_STAT_MODE AVOID_FACT AVOID_MULTI_INDEX AVOID_STAR_JOIN BASED BOUND_IMPL_PDQ BUCKETS CONTEXT DEBUG_ENV ENABLE ERKEY FACT FORCED HASH HOME IMPLICIT_PDQ INDEX_ALL MULTI_INDEX NON_DIM NUMTODSINTERVAL NUMTOYMINTERVAL PROPERTIES STAR_JOIN STATCHANGE STATLEVEL STORE TO_DSINTERVAL TO_YMINTERVAL TRANSITION TRUSTED UID UPON USE
Version 11.50.xC6	BLOBDIR CLOBDIR DATAFILES DELIMITED DELUXE DISK EXPRESS FIXED FORMAT FORCE_DDL_EXEC IFX_BATCHEDREAD_TABLE INFORMIX MAXERRORS NUMROWS RECORDEND REJECTFILE RETAINUPDATELOCKS SAMEAS

Table 18-1. New keywords of SQL (continued)

Informix Version	Keywords
Version 11.50.xC5	CONNECT_BY_ISCYCLE CONNECT_BY_ISLEAF CONNECT_BY_ROOT MERGE MATCHED SIBLINGS SYS_CONNECT_BY_PATH
Version 11.50.xC2	HDR
Version 11.50	BIGINT BIGSERIAL EXTDIRECTIVES PREPARE VERCOLS
Version 11.10	ADMIN AVOID_INDEX_SJ FINAL IDSSECURITYLABEL INDEX_SJ INSERTING REFERENCES SAMPLING SELECTING STATEMENT SYSDBCLOSE SYSDBOPEN TASK UPDATING USELASTCOMMITTED WITH In addition, the DBSECADM role is reserved for LBAC administrative work. Version 11.10 contains a new database, the sysadmin database. If your source database server contains a database named sysadmin , you must rename it.

Table 18-1. New keywords of SQL (continued)

Informix Version	Keywords
Version 10.0	ACTIVE CURRENT_ROLE DEFAULT_ROLE DIRECTIVES ENCRYPTION HINT IGNORE INACTIVE INITCAP INLINE INOUT LIMIT LOAD ONLINE OPTCOMPIND PARTITION PASSWORD REUSE SAVE SKIP STORAGE TEMPLATE TEST TRUNCATE TYPEID TYPENAME TYPEOF UNLOAD XADATASOURCE XID
Version 9.40	COLLATION CROSS FULL INSTEAD RESTART RIGHT
Version 9.30	AVOID_EXECUTE AVOID_SUBQF USE_SUBQF
Version 9.21	AVOID_HASH AVOID_INDEX AVOID_NL RAW STANDARD USE_HASH USE_NL
Version 7.31	INNER JOIN LEFT LOCKS RETAIN

Related concepts:

 [Keywords of SQL for IBM Informix \(SQL Syntax\)](#)

Chapter 19. System catalog and system database changes by version

Each version of Informix contains system catalog table changes and **sysmaster** database changes.

Changes for version 11.70

Version 11.70 contains new **sysadmin**, **sysuser**, and system catalog tables. Version 11.70 also contains some system catalog table changes.

The system catalog now contains the following table:

- **sysfragdist**

The **sysadmin** database contains the following new table:

- **storagepool**

The **sysuser** database contains the following new tables:

- **sysintauthusers**
- **sys surrogategroups**
- **sys surrogates**
- **sys cxattributes**
- **sys cxusers**
- **sys trustedcontext**
- **sys usermap**

The **sysprocedures** table contains a new column, **procflags**.

The **sysfragdist**, **sysdistrib**, **sysfragments**, and **sysindices** tables contain new columns with information about new features, such as changes to data distribution statistics and fragmentation strategies.

The **sysaudit** table also contains six new columns: **succ6**, **succ7**, **succ8**, **fail6**, **fail7**, and **fail8**

The **sysams** system catalog table has a new column, **am_expr_pushdown**.

The **am_param** column in the **sysstabamdata** table is now an LVARCHAR(8192) data type.

Changes for version 11.50

Version 11.50 contains two new **sysmaster** database tables and new SMI tables.

The version 11.50.xC6 **sysmaster** database contains the following new tables:

- SYSEXTERNAL
- SYSEXTCOLS
- SYSEXTDFILES

The version 11.50.xC4 **sysmaster** database contains the new **syscompdicts_full** table and the new **syscompdicts** view.

The version 11.50.xC3 **sysmaster** database contains the new **syscdc** table for the Change Data Capture API.

The version 11.50 **sysmaster** database contains the new **sysesappinfo** table.

The following new SMI tables contain information about Enterprise Replication that you can use to monitor status and diagnose problems:

- The **syscdr_state** table contains information about whether Enterprise Replication, data capture, data apply, and the network between servers is active.
- The **syscdr_dds** table contains information about the status of log capture and the proximity or status of transaction blocking (DDRBLOCK) or transaction spooling.
- The **syscdr_nif** table contains information about network connections and the flow of data between Enterprise Replication servers.
- The **syscdr_rcv** table contains information about transactions being applied on target servers and acknowledgments being sent from target servers.
- The **syscdr_atmdir** table contains information about the contents of the ATS directory.
- The **syscdr_risdir** table contains information about the contents of the RIS directory.
- The **syscdr_ats** table contains the first ten lines of content of each ATS file.
- The **syscdr_ris** table contains the first ten lines of content of each RIS file.
- The **syscdr_rqstamp** table contains information about which transaction is being added into each queue.
- The **syscdr_rqmhandle** table contains information about which transaction is being processed in each queue.

Changes for version 11.10

Version 11.10 contains new **sysmaster** database tables and the new **sysadmin** database.

The version 11.10 **sysmaster** database contains these new tables:

syscheckpoint

sysckptinfo

Schema changes were made to the **systables**, **sysindices**, and **sysfragments** tables for Version 11.10. For information about the current schema, see information on **systables**, **sysindices**, and **sysfragments** in the *IBM Informix Guide to SQL: Reference*.

Version 11.10 also includes a new database, **sysadmin**, which contains tables that store task properties. This database is dropped when you revert to earlier versions of the database server. If your source database server contains a **sysadmin** database, you must rename it.

Changes for version 10.00

Version 10.00 contains new **sysmaster** database tables and new SMI tables.

The version 10.00 **sysmaster** database contains these new tables:

sysdirectives sysbufpool, a system-monitoring interface (SMI)

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 about INOUT parameters.
- The **syspaghdr** table has a **pg_pagesize** column.
- The **sysptnhdr** table has a **pagesize** 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 **pagesize** columns.
- The views **syschunks** and **sysdbspaces** tables have a **pagesize** columns.
- The views **systabinfo** table has a **ti_pagesize** column.
- The views **systabpaghdrs** and **sysphyspaghdrs** tables have **pg_pagesize** columns.

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

Changes for version 9.40

Version 9.40 contains a new **sysmaster** database table, a new **collation** column, library pathname changes, and chunk format changes.

The following new system catalog table was added:

- **syssequences**

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

- **sysconstraints sysindices**
- **sysprocplan**
- **systrigbody**

Version 9.40 library pathname changes

Informix 9.40 and later do not install the following libraries in the **/usr/lib** directory on UNIX or Linux platforms:

- 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.

- High-Performance Loader Custom-Code Shared Library

If you use customized files with the High-Performance Loader, set the `HPL_DYNAMIC_LIB_PATH` configuration parameter in the `plconfig` file to the location of the custom-code shared library.

Chunk Format Change in Informix 9.4

Starting with version 9.40, Informix uses the following chunk features that cannot be reverted:

- Chunks greater than 2 GB in size
- Chunk offsets greater than 2 GB in size
- More than 2047 chunks

These features are enabled by default in version 9.40 and later, unless you just converted from a version of Informix that is prior to version 9.4; in that case, chunks that are greater than 2 GB are not enabled.

To stage a migration to support large chunks and large offsets:

1. To support creating new chunks and offsets greater than 2 GB, run **onmode -BC 1**.
2. Perform a complete system level-0 backup.
3. Test your data in the **onmode -BC 1** mode.
4. If you need to revert, drop any storage spaces that use the new chunk features.
5. When you are satisfied that your data has converted correctly and you no longer need to revert, run **onmode -BC 2** to put the server in large-chunk-only mode.

Reversion is not supported after running **onmode -BC 2**. All page writes are in the new format, including those to chunks that were created prior to version 9.40. After support for large chunks is enabled, it cannot be disabled.

6. Perform a complete system level-0 backup.

Changes for version 9.30

Some **sysmaster** database tables were removed from version 9.30.

The following tables were deleted from the **sysmaster** database in Version 9.30:

- **arc_ae_view** **arc_db_file_view**
- **arc_dbspace** **arc_dbspace_set**
- **arc_directory_view**
- **arc_file_view**
- **arc_file_copy_view**
- **arc_pendreq_view**
- **arc_phys_dev**
- **arc_rep_table**
- **arc_replicate**
- **arc_req_vset_view**
- **arc_request_view**
- **arc_save_set_view**
- **arc_server**
- **arc_version**
- **arc_vol_lock_view**

- `arc_volume_view`
- `arc_vset`
- `arc_vset_user_view`
- `arc_vset_view`

Column-width changes in `sysmaster` tables in Version 9.20 and later versions

Version 9.20 and later versions of Informix 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`
- `sysams`
- `sysattrtypes`
- `sysblobs`
- `syscasts`
- `syscolattribs`
- `syscolumns`
- `sysconstraints`
- `sysdomains`
- `sysfragauth`
- `sysfragments`
- `sysindexes`
- `sysindices`
- `sysobjstate`
- `sysopclasses`
- `sysopclstr`
- `sysprocedures`
- `sysroutinelangs`
- `syssynonyms`
- `syssyntable`
- `systabamdata`
- `systables`
- `systracemsgs`
- `systriggers`
- `sysxdtypes`

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

- `arc_dbspace`
- `arc_dbspace_set`

- arc_phys_dev
- arc_rep_table
- arc_replicate
- arc_server
- arc_version
- arc_vset
- arc_vset_view
- flags_text syscfgtab
- sysconfig
- syscrtadt
- sysdatabases
- sysdblocale
- sysdbspaces
- sysdbspartn
- sysdbstab
- sysdic
- sysdiccache
- sysdistcaEche
- sysdsc
- sysextents
- sysextspaces
- syslocks
- sysopendb
- sysprc
- sysproccache
- sysptprof
- syssdblock
- syssqlcurall
- syssqlcurses
- syssqlstat
- systabnames
- systrans
- systxptab

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
- sysams
- syscasts
- syscolauth
- sysconstraints
- sysdomains
- sysfragauth
- sysindexes
- sysindices

- syslangauth
- sysobjstate
- sysopclasses
- sysopclstr
- sysprocauth
- sysprocedures
- sysroleauth
- syssynonyms
- syssyntable
- systabauth
- systables
- systriggers
- sysusers
- sysxtdtypeauth
- sysxtdtypes

The following **sysmaster** database tables changed:

- sysaudit
- sysdatabases
- sysdbspaces
- sysdbspartn
- sysdbstab
- sysdic
- sysdiccache
- sysdistcache
- sysdsc
- sysextspaces
- sysprc
- sysproccache
- sysrstcb
- sysscb1st
- sysessions
- systabnames
- sysuserthreads

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

- sysadinfo
- syschktab
- syschunks
- syscrtadt
- sysdrch
- sysdri
- sysmchktab

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**
- **sysdris**

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

- **sysdistcache**
- **sysdsc**
- **sysprc**
- **sysproccache**

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

- **sysseprof**

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 in sysmaster tables in Version 9.20 and later versions

The data type of some **sysmaster** database tables was changed in Informix Version 9.20.

The “Column-width changes in sysmaster tables in Version 9.20 and later versions” on page 19-5 topic lists columns that have changed from the CHAR data type to the VARCHAR data type. In addition, one or more columns changed from the SMALLINT data type to the integer data type in the following **sysmaster** database tables:

- **sysdbspaces sysdbstab**
- **sysdic sysrstcb**
- **sysdblock**

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

- **sysdistrib**

Changes in treatment of null values in sysmaster tables in Version 9.30

Starting with Version 9.30, nulls are allowed in some **sysmaster** database tables, in which nulls were previously not allowed.

Nulls are allowed for some columns in the following **sysmaster** database tables:

- **arc_ae_view**
- **arc_db_file_view**
- **arc_directory_view**

- `arc_file_copy_view`
- `arc_file_view`
- `arc_pendreq_view`
- `arc_req_vset_view`
- `arc_request_view`
- `arc_save_set_view`
- `arc_vol_lock_view`
- `arc_volume_view`
- `arc_vset_user_view`
- `arc_vset_view`

Other sysmaster database table and column changes in Version 9.30

Informix Version 9.30 contains a new **sysmaster** database table. In addition, some tables were removed and new columns were added to a few tables.

The following tables have been added to the **sysmaster** database:

- `logmessage`
- `syscdrack_buf`
- `syscdrack_txn`
- `syscdrctrl_buf`
- `syscdrctrl_txn`
- `syscdrprog`
- `syscdrq`
- `syscdrrecv_buf`
- `syscdrrecv_txn`
- `syscdrtx`

The `arc_change_log` table has been deleted from the **sysmaster** database:

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

- `sysams`
- `sysdistrib`
- `sysprocedures`
- `sysroutinelangs`

Several columns have been added to the **sysdbstab** system catalog table.

Remote queries on system catalog tables between Version 7.31 and later versions

Certain system catalog tables use data types that are not supported in Informix Version 7.31. Remote queries that issue a `SELECT *` statement on these system catalog tables from Version 7.31 to later versions will fail.

For example, the following queries that originate on Version 7.31 fail if you try to run them on a later version of the server:

```
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 between Version 7.31 and later versions

In Version 7.31, **sysindexes** is a table. In later Informix versions, **sysindexes** is a view. Additionally, changes in functionality in later versions might require changes to your applications, database schema, or administration scripts.

The ALTER TABLE statement fails for **sysindexes** because this statement is not valid for altering a view.

Changes in functionality when upgrading from Informix 7.31

If you are upgrading from Informix Version 7.3x, the following changes in functionality might require changes to your applications, database schema, or administration scripts:

- Some system catalog tables can no longer be queried remotely
Starting with Informix 9.x, some system catalog tables contain columns of user-defined built-in opaque data types, which cannot be queried remotely. Remote queries that issue a SELECT * statement on these system catalog tables will fail. Queries that reference columns of user-defined types will also fail. To query system catalog tables remotely, you must specify non-UDT columns in the SELECT statement.
For example:

```
SELECT idxname, idxtype FROM remdb@remserver:sysindices WHERE tabid=100;
```


For Informix 7.x, **sysindexes** is a table, but starting with Informix 9.x, **sysindexes** is a view on the **sysindices** table. Remote queries on **sysindexes** will always fail due to the way views are materialized. Run remote queries on specific non-opaque columns of the **sysindices** table instead of on **sysindexes**.
This caveat for remote queries on opaque data types is also applicable for queries spanning multiple databases on the same server instance.
- Column limit for NVARCHAR, VARCHAR, BYTE, and TEXT data types
If you are migrating from Informix 7.3x, the number of columns of the VARCHAR or NVARCHAR data type per table has been reduced from 231 to 195, within a row size of 32762 bytes and based on a page size of 2K. The same 195-column restriction also applies to BYTE and TEXT columns.
On platforms where the page size is 4K (Windows and AIX), the limit for the number of columns limit is approximately 450 columns.
Before you upgrade to the current version of Informix, ensure that your tables comply with this restriction.
- Stored procedure parameter limit
Informix has a limit of 341 parameters for each stored procedure.
The same limit on the number of parameters also applies to UDRs written in the Java language. UDRs written in the C language can have no more than 102 parameters.
- Detached indexes
By default, all new indexes that the CREATE INDEX statement creates in Informix Version 9 and later are detached and stored in separate tblspaces from the data. Indexes created in version 7.31, remain attached until you rebuild them.

You can attach only B-tree indexes that are non-fragmented and that are on non-fragmented tables (the version 7.31 behavior). All other indexes, including extensibility related indexes such as R-trees and UDT indexes, must be detached. You cannot revert detached indexes to version 7.31. To enable reversion to version 7.31, retain the version 7.31 attached index behavior by setting the environment variable **DEFAULT_ATTACH** in the application environment.

Chapter 20. Server library name changes by version

Each version of the database server contains some new server library names.

The following database server library names have new names. These library files have a .so or .dll extension.

Library	Name for 9.14, 9.21, 9.30, or 9.40 Server	Name for 10.0 Server	Name for 11.10, 11.50, or 11.70 Server
Optical	iosm09a	iosm10a	iosm11a
pload	ipldd09a	ipldd10a	ipldd11a
Simple password CSM	ispws09a	ispws10a	ispws11a
Encryption CSM	iencs09a	iencs10a	iencs11a

Part 6. Appendixes

Appendix. Accessibility

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

Accessibility features for IBM Informix products

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

Accessibility features

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

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

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

IBM and accessibility

For more information about the IBM commitment to accessibility, see the *IBM Accessibility Center* at <http://www.ibm.com/able>.

Dotted decimal syntax diagrams

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

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

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

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

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

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

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 refers 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 can write HOST STATE, but you cannot 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 repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

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

IBM may not offer the products, services, or features discussed in this document in other 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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 document 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 is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Trademarks

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

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

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

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Index

A

- Abbreviated years 9-1
- Accessibility A-1
 - dotted decimal format of syntax diagrams A-1
 - keyboard A-1
 - shortcut keys A-1
 - syntax diagrams, reading in a screen reader A-1
- ALARMPROGRAM configuration parameter 6-4, 6-6
- ANSI joins 7-2

B

- Backups
 - after upgrading 6-10
 - before reverting 7-11
 - before upgrading to a new version 3-8
 - logical logs 6-6
 - ON-Bar utility 3-8, 7-16
 - ontape utility 3-8, 7-16
 - source database 3-8
- Binary files, loading 13-1, 13-2
- BladeManager
 - installing and registering DataBlade modules 6-11, 7-16
 - removing new extensions before reversion 7-13
- Blobspaces
 - moving, with onunload and onload 13-10, 13-11

C

- Character-position form of FILE and INSERT statements 10-9
- Checking available space
 - before migration 3-2
- Checking database integrity 6-10
- Chunks
 - reverting reserve pages 7-9
- Clusters
 - migrating to new release 5-1, 5-3
 - restoring from a backup archive 5-12
 - restoring from the HDR secondary server 5-12
 - reverting 5-1, 5-10
 - upgrading to a new fix pack 5-2
 - upgrading to a new PID 5-2
- Code-set conversion
 - HPL 2-4
- Column-level encryption 7-2
- Command file
 - dbload 10-5
- Communications Support Module
 - configuring after migration 6-5
 - removing if reverting 7-15
 - saving before reverting 7-10
- compliance with standards xiv
- concsm.cfg file
 - creating entries after migration 6-5
 - removing if reverting 7-15
 - saving before reverting 7-10
- Configuration file
 - customizing after migration 6-4
 - replacing after reversion 7-14
 - saving before migration 3-5

- Configuration file (*continued*)
 - saving before reverting 7-10
- Configuration parameters
 - ALARMPROGRAM 6-4, 6-6
 - changed in the onconfig.std file 17-9, 17-11
 - changed in Version 10.0 17-16
 - changed in Version 11.10 17-16
 - changed in Version 11.50 17-11
 - changed in version 11.70 17-9
 - changed in Version 9.30 17-16
 - changed in Version 9.40 17-16
 - changes by version 17-1
 - CONVERSION_GUARD 6-7, 15-1
 - deprecated in Version 11.50 17-11
 - deprecated in version 11.70 17-9
 - HPL_DYNAMIC_LIB_PATH 6-4
 - removed in Version 10.0 17-16
 - removed in Version 11.10 17-16
 - removed in Version 11.50 17-11
 - removed in version 11.70 17-9
 - removed in Version 9.30 17-16
 - removed in Version 9.40 17-16
 - RESTORE_POINT_DIR 6-7, 15-1
 - ROOTOFFSET 6-4
 - ROOTPATH 6-4
 - ROOTSIZE 6-4
 - STOP_APPLY 9-1
 - UPDATABLE_SECONDARY 9-1
 - USELASTCOMMITTED 9-1
- conpload.sh script 6-6
- conploadlegacy.sh script 6-6
- CONVERSION_GUARD configuration parameter 6-7, 15-1

D

- Data integrity 6-10
- Data migration
 - constraints 2-1
 - issues to consider 2-1
 - overview 2-1
 - prerequisites 2-1
 - tools 2-1, 2-4, 2-5
- Data types
 - user defined
 - converting from version 9.21 4-3
- Database server
 - initializing after upgrading 6-6
 - new version
 - performance tuning 6-11
 - reverting 7-1
 - reverting from current version 7-13
 - starting after upgrading 6-6
- Database servers
 - migrating 6-1, 6-7
 - platforms and versions 1-5
 - preparing for migration 3-1
 - upgrading 6-1, 6-7
- Databases
 - ownership, set by onload 13-10
 - verifying integrity 6-10

- DataBlade modules
 - installing after upgrading 6-5
 - registering 6-11, 7-16
 - DB-Access
 - input from the dbschema utility 11-15, 11-16
 - dbexport
 - SELECT triggers, disabling 9-1
 - dbexport utility 9-1
 - c option 9-3, 9-5
 - d option 9-3
 - nw option 9-3
 - q option 9-3
 - si option 9-3, 9-6
 - ss option 9-3, 9-6
 - V option 9-3
 - version option 9-3
 - X option 9-3
 - destination options 9-6
 - Interrupt key 9-5
 - schema output 9-7
 - syntax 9-3
 - dbimport utility 9-1
 - c option 9-8, 9-9
 - l option 9-13
 - nv option 9-8
 - q option 9-8
 - V option 9-8
 - version option 9-8
 - X option 9-8
 - create options 9-11
 - database logging mode 9-13
 - importing from another computer 2-6
 - input file location options 9-10
 - Interrupt key 9-9
 - locale, changing 9-14
 - renaming a database 9-13
 - syntax 9-8
 - using with GLS 9-8
 - using with NLS 9-14
 - dbload utility
 - c command file option 10-1
 - d database option 10-1
 - e errors option 10-1
 - e option 10-4
 - i ignore rows option 10-1
 - i option 10-4
 - k option 10-1
 - l error log file option 10-1
 - p option 10-1
 - r option 10-1, 10-3
 - s option 10-1
 - V option 10-1
 - version option 10-1
 - X option 10-1
 - compared to LOAD 10-1
 - creating a command file 10-5
 - dbload utility
 - n commit interval option 10-1
 - FILE statement 10-5
 - guidelines for handling objects 10-4
 - ignoring rows 10-4
 - importing from another computer 2-6
 - INSERT statements 10-5
 - compared to SQL INSERT statement 10-10
 - using 10-6
 - Interrupt key 10-4
 - number errors to allow 10-4
 - dbload utility (*continued*)
 - overview 10-1
 - speed, increasing 10-4
 - syntax 10-1
 - table locking 10-3
 - writing a command file
 - in character-position form 10-11
 - in delimiter form 10-8
 - dbschema utility
 - ss option 11-6
 - u option 11-6
 - chunk schema 11-9, 11-10
 - create schema across a network 11-5
 - create schema for a database 11-5
 - distribution information 11-13
 - example of file for DB-Access 11-16
 - guidelines 11-2
 - log schema 11-9, 11-10
 - output example 11-14
 - overview 11-1
 - owner conventions 11-5
 - privileges information 11-12
 - privileges information for a role 11-13
 - re-creating the schema 11-16
 - sequence schema 11-7
 - specifying a table, view, or procedure 11-8
 - storage space schema 11-9, 11-10
 - synonym schema 11-7
 - syntax 11-2
 - syntax for role schema 11-11
 - table information 11-9
 - DBSECADM role 9-1
 - dbspaces
 - moving tables to another dbspace 13-12
 - Delimiter form of FILE and INSERT statements 10-6, 10-8
 - Diagnostic information to gather 3-9
 - Directories
 - installation 6-2
 - Disabilities, visual
 - reading syntax diagrams A-1
 - Disability A-1
 - Distributed queries
 - with ANSI joins 7-2
 - Dotted decimal format of syntax diagrams A-1
 - Dummy updates 7-11
- ## E
- Enterprise Gateway
 - using to import data 2-6, 2-7
 - Enterprise Gateway Manager 2-7
 - Environment variables
 - changes by version 16-1
 - DB_LOCALE 9-14
 - DBCENTURY 9-1
 - DBDATE 9-1
 - DBTEMP 9-14
 - GL_DATETIME 12-2
 - GL_USEGLU 6-4
 - IFX_ONLOAD_AUTO_UPGRADE 6-6
 - INFORMIXSERVER 6-4
 - INFORMIXSQLHOSTS 6-4
 - ONCONFIG 6-4
 - PATH 6-4
 - resetting after reversion 7-15
 - TEMP 9-14
 - TMP 9-14

Environment variables (*continued*)
 USE_DTENV 12-2
Extents, checking 6-10
Extracting schema information 8-1

F

Fast recovery
 initiating 3-7
features 0-3
Features
 reviewing 3-2
FILE statement
 character-position form 10-9
 delimiter form 10-6, 10-8
 syntax for
 character-position form 10-10
 delimiter form 10-6
 with dbload 10-5
FIRST clause 7-2
Fix packs
 naming conventions 1-6

G

GL_DATETIME environment variable 9-1, 12-2
GL_USEGLU environment variable 6-4
Global Language Support (GLS)
 dbimport utility 9-8
 using onload and onunload 13-2
GRANT statement
 role privileges 11-13

H

Hardware
 prerequisites for migrating 1-5
HEX binary data 9-8
High-Performance Loader
 ipload utility 2-4
 loading data 2-4
 onpladm utility 2-4
HPL_DYNAMIC_LIB_PATH configuration parameter 6-4

I

id_column 7-11
IFX_ONPLOAD_AUTO_UPGRADE environment variable 6-6
Importing
 non-Informix data 2-6
in dbschema output 11-12, 11-13
In-place alters
 dummy updates 7-11
 oncheck -pT command 7-11
In-place migration 1-2, 1-3
Index
 checking 6-10
 rebuilding 6-6
industry standards xiv
Informix
 fix packs 1-6
 installing 6-2
 naming conventions
 releases 1-6
INFORMIXDIR directory 6-2

INFORMIXSERVER environment variable 6-4
INFORMIXSQLHOSTS environment variable 6-4
Initializing
 after upgrading 6-6
INSERT statements
 character-position form 10-9
 delimiter form 10-6
 syntax for character-position form 10-10
 with dbload 10-5
Installation directory 6-2
Installing Informix 6-2

J

Java UDRs 7-15

K

Keywords of SQL
 new 18-1

L

Label-based access control (LBAC) 9-1
LATERAL keyword 7-2
Level-0 backup 3-8, 6-10, 7-16
 after moving data 13-11
LIMIT keyword 7-2
LOAD SQL statement
 for locales that support multibyte code sets 12-2
 overview 12-1
 syntax 12-2
LOAD statement
 for non-default GL_DATETIME environment variable
 settings 12-2
 for non-default locales 12-2
Loading
 ASCII files 2-6
 binary data 13-1, 13-2
 data 2-1, 2-4, 2-5, 2-6
Locking
 set by onload 13-10
Logical log
 backup 6-6
 out of space 6-6
 space required for migration 3-2
LTAPEDEV configuration parameter
 onunload/onload 13-11

M

Migrating
 between 32-bit and 64-bit database servers 3-10
 on the same computer 1-3
 overview 1-1
 to a different computer 1-4
Migrating a database
 constraints 2-1
 issues to consider 2-1
 overview 2-1
 planning for 2-1
 prerequisites 2-1
 to a new operating system 8-1
 tools 2-1, 2-4, 2-5
Migrating with Enterprise Replication 4-1, 4-2

- Migration
 - before you begin 3-1
 - checklist 3-9
 - diagnostic information that you need before upgrading 3-9
 - hardware prerequisites 1-5
 - in-place 1-2, 1-3
 - monitoring status with online.log 6-4
 - Non-in-place 1-2, 1-4
 - onload utility 13-3
 - onunload utility 13-3
 - paths 1-6
 - planning 3-1
 - planning for 1-2
 - preparing for 3-1
 - prerequisites 3-1
 - process 1-1, 1-2
 - skills you need 1-1
 - space requirements 3-2
 - time it takes 1-1
 - tools 1-3
 - with Enterprise Replication 4-1
 - with HDR, RS, and SD servers 5-1, 5-2, 5-3
 - with high-availability clusters 5-1, 5-2, 5-3

- Mode
 - checking 3-8
- Monitoring migration status 6-4

- Moving data
 - blobspaces 13-11
 - constraints 2-1
 - overview 2-1
 - using dbexport and dbimport 9-1
 - using dbload 10-1
 - using distributed SQL 2-7
 - using onload and onunload 13-1
 - using onunload and onload 13-11, 13-12
 - when changing operating systems 8-1, 8-2, 8-3

- Multi-node Active Clusters for High Availability (MACH) Clusters
 - migrating to new release 5-1

N

- Native Language Support (NLS)
 - populating with dbimport 9-14
- Non-in-place migration 1-4
- Non-Informix data, importing 2-6
- NOVALIDATE constraint mode 9-8

O

- ON-Bar utility 6-8
 - backing up
 - after upgrading 6-10
 - backing up before upgrading 3-8
- oncheck utility
 - cc database_name option 3-8, 6-10
 - cD database_name option 3-8, 6-10
 - ce option 3-8, 6-10
 - cI database_name option 3-8, 6-10
 - cr option 3-8, 6-10
 - cs sbspace_name option 6-10
 - cS sbspace_name option 6-10
 - rebuilding table indexes 6-6
 - verifying database integrity 3-7, 6-10
- ONCONFIG environment variable 6-4

- ONCONFIG file
 - changes by version 17-1
 - customizing after migration 6-4
- onconfig.std file
 - changes in Version 11.50 17-11
 - changes in version 11.70 17-9
- oninit utility
 - s option 3-7
- online.log 6-4
- onload and onunload utilities 13-11, 13-12
- onload utility
 - constraints 13-9
 - constraints on use 13-2
 - create options 13-8
 - handling large objects in a blobspace 13-10
 - how it works 13-3
 - logging status 13-10
 - moving a database 13-11
 - moving a table 13-11, 13-12
 - moving locales 13-2
 - moving to another dbspace 13-12
 - ownership and privileges 13-10
 - specifying source parameters 13-7
 - syntax 13-6
 - using between computers 13-1
- onmode -b command 14-1
- onmode utility
 - b option 7-14
 - ky option 3-6
 - sy option 3-6
 - reverting from the current version 7-14
 - shutting down 3-6
 - shutting down the server 3-6
- onmode-b command 14-1
 - syntax 14-2
- onpload database
 - upgrading 6-6
- onrestorept utility
 - Clusters
 - restoring primary server to a consistent point 5-11
 - overview 15-1
 - syntax 15-1
 - undoing failed upgrade changes 5-11, 6-7
- onstat utility 3-8
- ontape utility
 - a option 6-6
 - backing up
 - after upgrading 6-10
 - before upgrading 3-8
- onunload utility
 - constraints on use 13-2, 13-5
 - destination parameters 13-4
 - how it works 13-3
 - locking 13-6
 - logging mode 13-6
 - moving a database 13-11
 - moving a table 13-11, 13-12
 - moving locales 13-2
 - moving to another dbspace 13-12
 - ownership and privileges 13-5
 - syntax 13-3
 - unloading tables 13-5
 - using between computers 13-1
 - what is included with a
 - database 13-5
 - table 13-5

- Operating system
 - adjusting tables after changing operating systems 8-2
 - moving data to another one 8-1, 8-2, 8-3
- Operating systems
 - not supported 1-5
 - supported 1-5
- ORDER BY clause 7-2
- Overview of migration process 1-1, 1-2, 1-3

P

- PATH environment variable 6-4
- Performance tuning
 - adjusting queries after upgrading 6-11
 - after upgrading 6-11
- Planning
 - before moving data 2-1
 - data migration 2-1
 - for exporting and importing data 8-1
 - for migration 1-1, 3-1
 - for upgrading your server 3-1
- Platforms, moving data between
 - compatible computers 2-6
- Privileges 11-12, 11-13
 - required for onunload 13-5

Q

- Queries
 - adjusting after upgrading 6-11
- Quiescent mode 3-8

R

- Recompiling Java UDRs 7-15
- Registering DataBlade modules 6-11, 7-16
- Reserve pages, checking 6-10
- restore points 15-1
- RESTORE_POINT_DIR configuration parameter 6-7, 15-1
- Reverse migration 7-1
- Reversion utility 7-14
- Revert to original version 7-1
- Reverting
 - backing up before you start 7-11
 - before using the onmode -b command 14-1
 - chunk reserve pages 7-9
 - determining if it is possible 7-2
 - from the new version 7-1
 - from Version 11.70 7-13
 - limitations 7-2
 - remove features 7-13
 - restrictions for 7-2
 - restrictions for reverting to prior versions 7-2
 - using the onmode -b command 14-1
 - with Enterprise Replication 4-3
 - with HDR, RS, and SD servers 5-1, 5-10
 - with high-availability clusters 5-1, 5-10
- ROOTOFFSET configuration parameter 6-4
- ROOTPATH configuration parameter 6-4
- ROOTSIZE configuration parameter 6-4
- Running the reversion utility 7-14

S

- Schema
 - create across a network 11-5
 - create for a database 11-5
 - display with dbschema 11-1
- Screen reader
 - reading syntax diagrams A-1
- Scripts
 - concdr.bat 4-2
 - concdr.sh 4-2
 - conpload.sh 6-6
 - conploadlegacy.sh 6-6
- SELECT triggers, disabling with dbexport 9-1
- Shortcut keys
 - keyboard A-1
- Simple large objects
 - moving with onload 13-10, 13-11
- SKIP keyword 7-2
- Slow queries
 - adjusting after upgrading 6-11
- sm_versions file 3-6, 6-8
- smi_unld utility 7-10
- Space
 - checking availability before migration 3-2
 - for sysmaster database 3-2
- SQL keywords 18-1
- SQL statements
 - UPDATE STATISTICS
 - data distributions 11-13
- sqlhosts file
 - save a copy when migrating 3-5
- sqlhosts file, UNIX
 - changing name or path 6-4
 - csn option 6-5
- standards xiv
- Starting
 - after upgrading 6-6
 - server after reversion 7-15
- STOP_APPLY configuration parameter 9-1
- Syntax diagrams
 - reading in a screen reader A-1
- sysadmin database 19-2
- sysbufpool table 19-3
- syscdc database 19-1
- sysdirectives table 19-3
- sysindexes
 - changes to 19-10
- sysmaster database
 - and logical logs 6-6
 - changes to 19-1
 - space required for migration 3-2
- sysseappinfo database 19-1
- System catalogs
 - changes to 19-1, 19-9
 - checking tables 6-10

T

- TAPEDEV configuration parameter, with onunload and onload 13-11
- Transactions
 - checking for open ones 3-7
- Truncate keyword 7-2

U

- UNLOAD SQL statement
 - for locales that support multibyte code sets 12-2
 - overview 12-1
 - syntax 12-1
- UNLOAD statement
 - for non-default GL_DATETIME environment variable settings 12-2
 - for non-default locales 12-2
- UPDATABLE_SECONDARY configuration parameter 9-1
- UPDATE statements
 - dummy 7-11
- UPDATE STATISTICS statement 6-9
 - data distributions 11-13
 - using after reversion 7-15
- Upgrading 1-3
 - migration paths 1-6
- Upgrading your server
 - overview of tasks 6-1
 - planning 3-1
 - preparing for 3-1
 - preparing to undo changes 3-4
 - prerequisites 3-1
 - restoring files after a failure 6-7, 15-1
- USE_DTENV environment variable 9-1, 12-2
- USELASTCOMMITTED configuration parameter 9-1
- USELASTCOMMITTED session environment variable 9-1
- User-defined data types
 - converting from version 9.21 4-3
- Utilities
 - dbexport 8-1, 9-1
 - dbexport syntax 9-3
 - dbimport 8-1, 9-1
 - dbimport syntax 9-8
 - dbload 10-1
 - dbschema 11-1
 - onload 8-1, 13-6
 - onload and onunload 13-1, 13-11, 13-12
 - onpladm 2-4
 - onpload 2-4
 - onunload 8-1, 13-3

V

- Verifying data integrity 6-10
- Visual disabilities
 - reading syntax diagrams A-1



Printed in USA

GC27-3546-06



Spine information:

Informix Product Family Informix **Version 11.70**

IBM Informix Migration Guide

