# INFORMIX®-Universal Server

# Backup and Restore Guide

Version 9.1
March 1997
Part No. 000-4804

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025

Documentation Team: Jill Berman, Barbara Daniell, Kami Shahi, Eileen Wollam

# Table of Contents

# Introduction

Read this introduction for an overview of the information provided in this manual and for an understanding of the documentation conventions used.

## About This Manual

The *INFORMIX-Universal Server Backup and Restore Guide* explains the concepts and methods that you need to understand to back up and restore INFORMIX-Universal Server data using the **onbar** utility. This manual is both a reference manual and a user guide.

### Organization of This Manual

This manual includes the following chapters:

- This Introduction provides an overview of the manual and describes the documentation conventions used.
- Chapter 1, "The ON-Bar Backup and Restore System," describes the components of the ON-Bar backup and restore system and explains the concepts involved in backing up and restoring Universal Server data.
- Chapter 2, "Using ON-Bar," describes how to use ON-Bar commands.
- Chapter 3, "Configuring ON-Bar," describes the process of configuring ON-Bar.

- Chapter 4, "Catalog Tables," describes the tables in the **sysutils** database that ON-Bar uses to track data objects and the components of the backup and restore system.
- Appendix A describes the ON-Bar message file and presents the ON-Bar messages.

## Types of Users

This manual is written for the administrator who must back up and restore Universal Server data. This manual assumes that you are familiar with Universal Server and the UNIX operating system.

If you have limited experience with relational databases, SQL, or your operating system, refer to *Getting Started with INFORMIX-Universal Server* for a list of introductory texts.

## Software Dependencies

This manual assumes that you are using INFORMIX-Universal Server, Version 9.1, as your database server.

ON-Bar requires that a shared-library implementation of the X/Open Backup Services Application Programmer's Interface (XBSA) that conforms to the X/Open specification be available.

In this manual, all instances of Universal Server refer to INFORMIX-Universal Server.

## Assumptions About Your Locale

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

This manual assumes that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale(s). For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *Guide to GLS Functionality*.

### Demonstration Database

The DB-Access utility, which is provided with your Informix database server products, includes a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. Sample command files are also included.

Many examples in Informix manuals are based on the **stores7** demonstration database. The **stores7** database is described in detail and its contents are listed in Appendix A of the *Informix Guide to SQL: Reference*.

The script that you use to install the demonstration database is called **dbaccessdemo7** and is located in the **$INFORMIXDIR/bin** directory. For a complete explanation of how to create and populate the demonstration database on your database server, refer to the *DB-Access User Manual*.

## Major Features

The Introduction to each Version 9.1 product manual contains a list of major features for that product. The Introduction to each manual in the Version 9.1 *Informix Guide to SQL* series contains a list of new SQL features.

Major features for Version 9.1 Informix products also appear in release notes.

## Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other Informix manuals.

The following conventions are covered:

- Typographical conventions
- Icon conventions
- Command-line conventions

## Typographical Conventions

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

| Convention | Meaning |
|---|---|
| KEYWORD | All keywords appear in uppercase letters in a serif font. |
| *italics* | Within text, new terms and emphasized words appear in italics. Within syntax diagrams, values that you are to specify appear in italics. |
| **boldface** | Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, filenames, table names, column names, icons, menu items, command names, and other similar terms appear in boldface. |
| `monospace` | Information that the product displays and information that you enter appear in a monospace typeface. |
| KEYSTROKE | Keys that you are to press appear in uppercase letters in a sans serif font. |
| ◆ | This symbol indicates the end of feature-, product-, platform-, or compliance-specific information. |

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

## Icon Conventions

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

### Comment Icons

Comment icons identify warnings, important notes, or tips. This information is always displayed in italics.

| Icon | Description |
|---|---|
| ⚠ | The *warning* icon identifies vital instructions, cautions, or critical information. |
| ⇨ | The *important* icon identifies significant information about the feature or operation that is being described. |
| 💡 | The *tip* icon identifies additional details or shortcuts for the functionality that is being described. |

## Command-Line Conventions

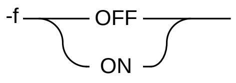
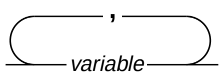
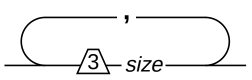
This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

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

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

| **Element** | **Description** |
|---|---|
| command | This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters. |
| *variable* | A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value. |
| -flag | A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen. |
| .ext | A filename extension, such as **.sql** or **.cob**, might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products. |
| ( . , ; + * - / ) | Punctuation and mathematical notations are literal symbols that you must enter exactly as shown. |
| ' ' | Single quotes are literal symbols that you must enter as shown. |
| $\boxed{\text{Privileges p. 5-17}}$ $\boxed{\text{Privileges}}$ | A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page. |
| — ALL — | A shaded option is the default action. |
| —►——► | Syntax within a pair of arrows indicates a subdiagram. |
| —\| | The vertical line terminates the command. |

(1 of 2)

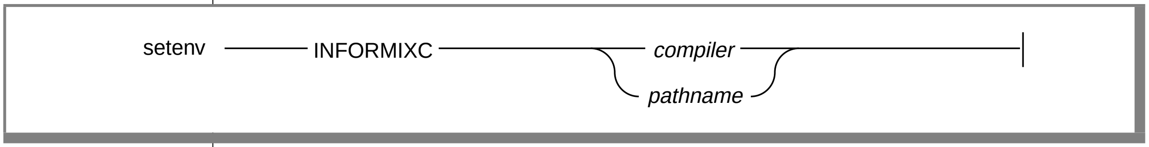
| Element | Description |
|---|---|
|  | A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.) |
|  | A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items. |
|  | A gate ( 3 ) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. Here you can specify *size* no more than three times within this statement segment. |

(2 of 2)

## How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

**Figure 1**
*Example of a Command-Line Diagram*



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

Figure 1 diagrams the following steps:

1. Type the word `setenv`.
2. Type the word `INFORMIXC`.
3. Supply either a compiler name or pathname.

   After you choose *compiler* or *pathname*, you come to the terminator. Your command is complete.
4. Press RETURN to execute the command.

## Additional Documentation

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

- On-line manuals
- Printed manuals
- Error message files
- Documentation notes, release notes, and machine notes

### On-Line Manuals

A CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see either the installation guide for your product or the installation insert that accompanies the documentation CD.

The documentation set that is provided on the CD describes Universal Server, its implementation of SQL, and its associated application-programming interfaces. For an overview of the manuals in the Universal Server documentation set, see *Getting Started with INFORMIX-Universal Server*.

### Printed Manuals

The Universal Server documentation set describes Universal Server, its implementation of SQL, and its associated application-programming interfaces. For an overview of the manuals in the Universal Server documentation set, see *Getting Started with INFORMIX-Universal Server*.

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com.

Please provide the following information:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

## Error Message Files

Informix software products provide ASCII files that contain all the Informix error messages and their corrective actions. To read the error messages in the ASCII file, Informix provides scripts that let you display error messages on the screen (**finderr**) or print formatted error messages (**rofferr**). For a detailed description of these scripts, see the Introduction to the *Informix Error Messages* manual.

## Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following on-line files, located in the **$INFORMIXDIR/release/en_us/0333** directory, supplement the information in this manual.

| On-Line File | Purpose |
|---|---|
| **ONBARDOC_9.1** | The documentation-notes file describes features that are not covered in this manual or that have been modified since publication. |
| **SERVERS_9.1** | The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds. |
| **IUNIVERSAL_9.1** | The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. Machine notes are named for the product described. |

Please examine these files because they contain vital information about application and performance issues.

## Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. 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, on INFORMIX-Universal Server. In addition, many features of Universal Server comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

## Informix Welcomes Your Comments

Please tell us what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about corrections or clarifications that you would find useful. Include the following information:

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

Write to us at the following address:

> Informix Software, Inc.
> SCT Technical Publications Department
> 4100 Bohannon Drive
> Menlo Park, CA 94025

If you prefer to send email, our address is:

> doc@informix.com

Or send a facsimile to the Informix Technical Publications Department at:

> 415-926-6571

We appreciate your feedback.

Chapter

# 1 The ON-Bar Backup and Restore System

This chapter explains the concepts of the ON-Bar backup and restore system for Universal Server. It answers the following questions:

- What is ON-Bar?
- What is a backup?
- What is a logical-log backup?
- What is a restore?

## What Is ON-Bar?

ON-Bar is a *backup and restore system* for Universal Server. ON-Bar enables you to create a copy of your Universal Server data and later restore the data if it becomes lost or corrupted. The causes of lost or corrupted data can range from a program error to a disk crash to a disaster that damages the facility in which your computer resides.

As Figure 1-1 illustrates, the ON-Bar backup and restore system involves the following components:

- The **onbar** utility
- A storage manager provided by a third-party vendor
- The XBSA interface
- The ON-Bar catalog tables
- The ON-Bar emergency boot file
- The ON-Bar message file

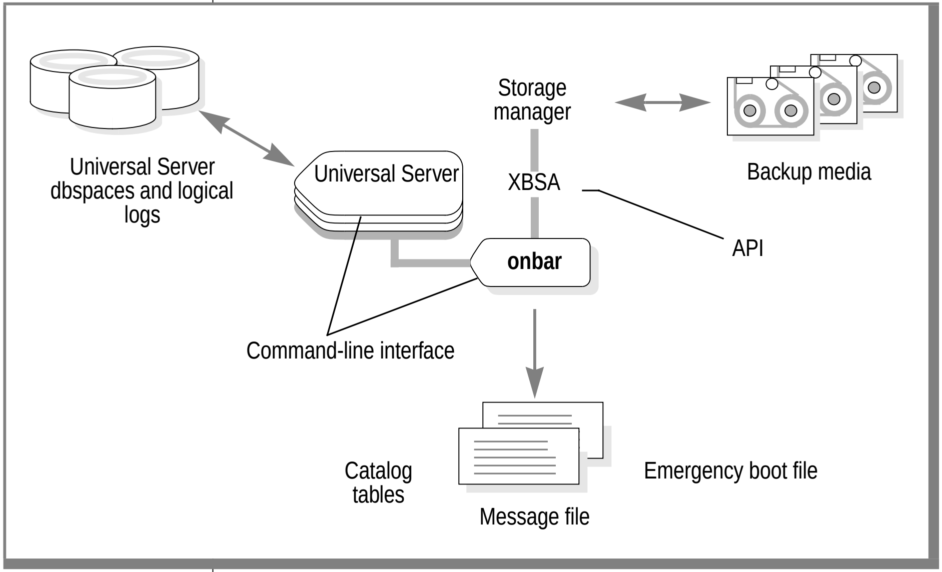


***Figure 1-1***
*ON-Bar Components*

## The onbar Utility

The **onbar** utility executes backup and restore requests; it can receive a request from the following sources:

- The command line
- A storage manager

The **onbar** utility communicates with both Universal Server and a storage manager. For a backup, **onbar** requests data from dbspaces, blobspaces, and logical-log files from Universal Server and passes them to the storage manager. For a restore, the process is reversed. The **onbar** utility requests data about dbspaces, blobspaces, and logical-log files from the storage manager and passes them to Universal Server.

## The Storage Manager

In the ON-Bar backup and restore system, the *storage manager* is an application that manages the storage devices and media used with backup and restore requests. A storage manager can manage a variety of storage devices and media ranging from simple tape and disk devices to stackers, robots, and jukeboxes. A storage manager might also perform the following functions:

- Schedule backups
- Support networked and distributed backups and restores
- Compress and decompress data
- Encrypt and decrypt data

## The XBSA Interface

ON-Bar and the storage manager communicate with each other through the X/Open Backup Services Application Programmer's Interface (XBSA). By using an open system interface to the storage manager, ON-Bar is able to work with a variety of storage managers that also use XBSA.

ON-Bar uses XBSA to exchange the following types of information with a storage manager:

- **Control data.** ON-Bar exchanges control data with a storage manager to verify that ON-Bar and XBSA are compatible, to ensure that objects are restored to the proper instance of Universal Server and in the proper order, and to track the history of backup objects.
- **Backup or restore data.** During backup and restore, ON-Bar and the storage manager use XBSA to exchange data from specified *dbobjects*, such as dbspaces, blobspaces, or logical-log files.

## The Catalog Tables

ON-Bar uses the following catalog tables in the **sysutils** database to track the compatibility of component versions, as well as to back up dbobjects and track instances of dbobjects through multiple backups. A list of ON-Bar tables in the **sysutils** database follows:

- The **bar_server** table tracks instances of Universal Server.
- The **bar_object** table tracks backup objects.
- The **bar_action** table tracks all backup and restore attempts against each backup object.
- The **bar_instance** table describes each object that is backed up during a successful backup attempt.
- The **bar_version** table lists compatible versions of ON-Bar, storage managers, and XBSA.

For a description of the content of these tables, see Chapter 4, "Catalog Tables."

## The Emergency Boot File

The ON-Bar *emergency boot file* contains enough information to perform a restore. The dbspaces in the following list are critical dbspaces:

- The root dbspace
- The dbspace that contains the physical log
- Any dbspace that contains a logical-log file

The purpose of the emergency boot file is to replace the **sysutils** tables during a cold restore so that ON-Bar can request the correct backup object from the storage manager.

ON-Bar must be able to restore objects from a storage manager even when the tables in the **sysutils** database are not available. During a cold restore, for example, the Universal Server is not available to access **sysutils**, so ON-Bar obtains the information it needs for the cold restore from the emergency boot file.

For information about the location of the emergency boot file, see "The Catalog Tables and the Emergency Boot File" on page 4-8.

## The Message File

As ON-Bar executes, it periodically writes to the ON-Bar *message file*. When ON-Bar encounters an error or a condition that warrants a warning, it writes a message to the message file. The message file lets you determine whether a backup or restore operation succeeded. It also documents which database objects were included in a backup or restore operation and approximately how long the operation took. The message log is located in either **$INFORMIXDIR/tmp/BAR_ACT.LOG** or in the location that the BAR_ACT_LOG configuration parameter specifies. For more information about the message file and a list of ON-Bar informational, warning, and error messages, see Appendix A.

## What Is a Backup?

A Universal Server *backup* copies all database objects that are managed by Universal Server to storage media at a *secondary storage* location. A database object, or dbobject, is a dbspace, blobspace, or logical-log file. A secondary storage location is a secure location that is separate from the computer where your data resides. Store the backup media *off-line* after you create a backup so that it cannot be overwritten or damaged.

Universal Server backs up all dbobjects with the following exceptions:

- The *dbspace pages* that are allocated to Universal Server but that are not yet allocated to a tblspace extent are not backed up.
- None of the configuration files, such as the **ONCONFIG** and **sqlhosts** files, are backed up.
- Mirror chunks are not backed up if the corresponding primary chunks are accessible.
- Blobs in blobspaces that are stored on optical platters (managed by INFORMIX-OnLine/Optical) are not backed up.
- Temporary dbspaces are not backed up.

If, during a backup, ON-Bar encounters a dbspace or blobspace that is down, ON-Bar skips it and writes a message to the message log. Dbspaces and blobspaces that ON-Bar skips are not backed up. Such dbspaces and blobspaces can be restored from older backups assuming that they have been backed up at least once.

## What Else Needs to Be Backed Up?

In addition to Universal Server dbobjects, back up the following files:

- The **ONCONFIG** file
- The **sqlhosts** file
- The **oncfg_$INFORMIXSERVER.$INFORMIXSERVERNUM**
- The ON-Bar emergency boot file

ON-Bar does not back up these files. How often you need to back them up depends on how frequently changes are made to them. In the case of the emergency boot file, back it up at least daily and always after you back up any of the critical dbspaces.

## What Is a Whole System Backup?

A *whole-system* backup is a backup of all Universal Server dbobjects from one **onbar** utility command. ON-Bar cannot back up dbobjects concurrently during a whole-system backup so they are backed up serially. A *selected-dbspace backup* is a backup of one or more specified dbobjects, the number of which is less than or equal to the total number of Universal Server dbobjects. For an explanation of how ON-Bar backs up and restores dbobjects concurrently, see "Parallel Backups and Restores" on page 1-22.

For information on how to perform both whole-system and selected-dbspace backups, see Chapter 3, "Configuring ON-Bar."

## What Are Backup Levels?

When Universal Server manages a large amount of data, it does not always make sense to back up all the data all the time. For example, if some of your information changes daily, but some data remains stable, it is inefficient to back up the unchanged data every time you back up the database server.

To provide a more flexible backup environment, Universal Server supports three *backup levels*:

- Level 0 backs up all used pages.
- Level 1 backs up all pages that have changed since the last level-0 backup.
- Level 2 backs up all pages that have changed since the last level-1 backup.

The following sections explain these three backup levels.

### *Level-0 Backups*

A level-0 backup is a baseline backup. It contains a copy of all pages that contain data for the specified dbspaces. You need all these pages to restore the database to the state that it was in when you made the backup. If computer media is completely destroyed and needs to be replaced with new media, you need a level-0 backup of all dbspaces to completely restore data on the replacement computer.

### *Level-1 Backups*

A level-1 backup contains a copy of every page that has changed since the last level-0 backup. The data that is copied to the backup reflects the state of the data at the time the level-1 backup began. A level-1 backup usually takes less time than a level-0 backup because only data that changed is copied to the storage manager.

A level-0 backup can be very time consuming because ON-Bar must write all the disk pages to backup media. Level-1 and level-2 backups take less time to create because ON-Bar only writes the changes that occurred since the last level-0 or level-1 backup. However, if you create level-0 backups infrequently, the level-1 backup could be quite large. For example, if you completed the last level-0 backup a day ago, there might not be many changes since then, and the level-1 backup might be quite small. However, if the last level-0 backup was a month ago, and many changes have occurred since then, the level-1 backup will be considerably larger.

### Level-2 Backups

A level-2 backup contains a copy of every page that has changed since the last level-1 backup. All data that is copied to the backup reflects the state of the data at the time the level-2 backup began.

A level-2 backup after a level-1 backup usually takes less time than another level-1 backup because only the changes that have been made since the last level-1 backup, instead of the last level-0 backup, are backed up.

## What Is a Logical-Log Backup?

The logical log stores a record of Universal Server activity. To illustrate, if you perform a level-0 backup on Monday at 10:00 P.M., and on Wednesday morning at 11:00 A.M. you suffer a mishap that destroys your databases, you *could* lose all transactions that occurred between 10:00 P.M. Monday and 11:00 A.M. Wednesday. However, if you specified transaction logging for your databases, any transactions that occurred after 10:00 P.M. Monday are recorded in the logical log. Figure 1-2 on page 1-11 illustrates the function of the logical log.

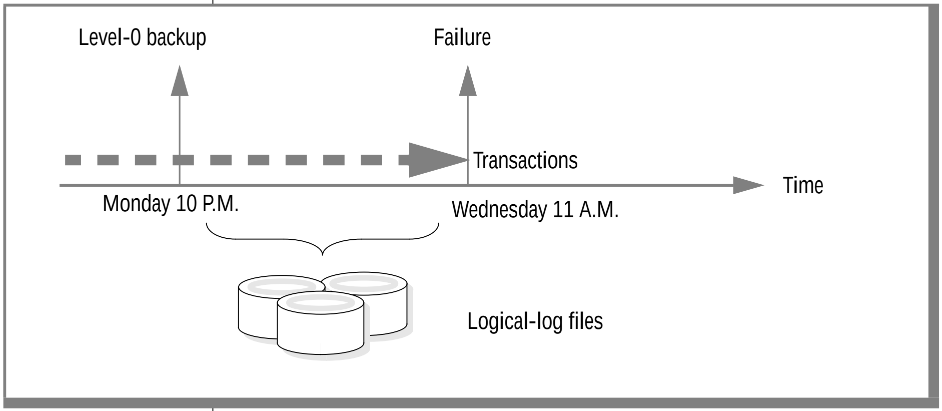


***Figure 1-2***
*The Function of the Logical Log*

A *logical-log backup* copies a logical-log file to a second location on storage media. Figure 1-3 illustrates a logical-log backup.

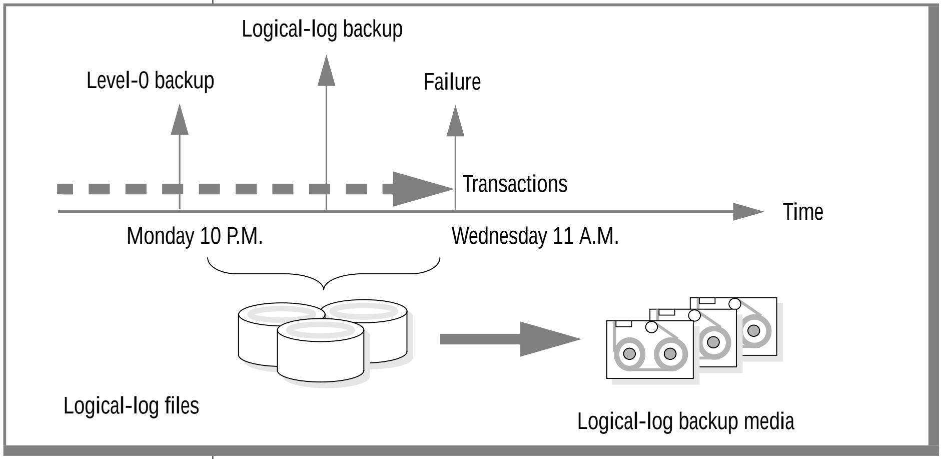


***Figure 1-3***
*Logical-Log Backup*

## Why You Need to Back Up the Logical-Log Files

When you specify logging for your databases, Universal Server records transactions that occur between backups of your dbspace data in the *logical log*, which consists of a finite number of *logical-log files* on disk. Universal Server continually writes new log records while retaining the log records it has already written in case those transactions must be restored. To retain the records in the logical log, yet allow Universal Server to continue writing new log records in a finite amount of space, you must free full log files by copying them to a safe place on disk or tape.

## If You Do Not Use Logging

Be aware that even if you do not specify logging for any of your databases, you can still have log backups. These backups are required relatively infrequently because they contain only administrative information such as checkpoint records and additions and deletions of chunks. By backing up these logical-log files, you are able to perform warm restores even though you are not using logging for any of your databases.

## When Should Logical-Log Files Be Backed Up?

Informix recommends that you attempt to back up each logical-log file as soon as it fills. You can tell if a log file is ready to be backed up by checking the flags field of **onstat -l**. When the flags field displays the following status, the log file is ready to be backed up:

```
U------
```

You can interpret this status to mean that the log file is Used, not Current and not yet Backed up. For more information on monitoring the status of logical-log files, see the *INFORMIX-Universal Server Administrator's Guide*.

Universal Server reuses logical-log files to minimize the amount of disk space that it needs for logging transactions. This means that after ON-Bar backs up a logical-log file because it is full, ON-Bar frees the logical-log file to make room for additional transactions. For a complete description of the logical log, see the *INFORMIX-Universal Server Administrator's Guide*.

If you do not want to monitor the logical-log files, you can use automatic logical-log backups.

## On-Demand and Automatic Backups

When you want to back up all the logical-log files that are full and ready to be backed up, you can start an *on-demand* backup. An on-demand backup backs up all the full logical-log files and then stops at the current logical-log file.

You can also start an *automatic* backup in which Universal Server backs up each logical-log file as it becomes full. If you perform automatic logical-log file backups you will never lose more than a partial logical-log file. Even in the worst-case situation when a chunk that contains logical-log files fails, you still lose no more than a partial logical-log file.

When you opt for automatic backups, the backup device must be dedicated to the backup process.

For information on how to set up automatic backups, see "Specifying Automatic Backup of the Logical-Log Files" on page 3-7.

## Saving Logical-Log Backup Data

The nature of logical-log data makes backing up logical-log files different from backing up other dbobject data. If one of your dbspace backup volumes becomes inaccessible, you can always restore from an older backup, if you have one. If a volume that contains logical-log files fails, however, you cannot roll forward the transactions from those log files or any subsequent log files.

Keep copies of your logical-log files until you are sure that you do not need them to complement a restore from a backup.

## Logical-Log Files and Blobspaces

Remember the following two points if you use blobspace data in a database that uses transaction logging:

- To ensure timely reuse of blobpages, back up the logical-log files. When users delete blobs in blobspaces, the blobpages are not freed for reuse until the log file that contains the delete records is freed. To free the log file, you must back it up.
- If a blobspace that needs to be backed up is unavailable during a logical-log backup, it is impossible to recover the blobs it contains. (However, blobpages from deleted blobs do become free when the blobspace is available again, even though the blobs within are not backed up.)

In addition, regardless of whether the database uses transaction logging, when you create a blobspace or add a chunk to a blobspace, the blobspace or new chunk is not available for use until the log file that records the event is not the current log file. For information on switching log files, see the *INFORMIX-Universal Server Administrator's Guide*.

## Salvaging Logical-Log Files

Under some circumstances, the physical media containing the logical logs must be replaced before beginning a cold restore. For more information on a cold restore, see "A Cold Restore" on page 1-17. If these media are changed before the logical-log files are salvaged, data is certainly lost. Perform a manual salvage before restoring the media and then perform the cold restore. For an example of how to perform a manual salvage, see "Salvaging Logical-Log Files" on page 2-12.

If you do not need to replace media, you do not need to perform the manual-salvage operation. Cold restore always attempts to salvage logical-log files before restoring the root dbspace. If the salvage fails, the restore continues. Transactions that cannot be salvaged are lost.

## What Is a Restore?

An Universal Server *restore* operation re-creates Universal Server data that has become inaccessible due to hardware or software failure. For example, any one of the following three conditions could require you to restore your Universal Server data:

- You need to replace a disk that contains the data.
- A logic error in a program has corrupted a database.
- You need to move your data to a new computer.

To restore your Universal Server data up to the time of the failure, you must have a dbspace backup of your Universal Server data and the logical-log files that contain all transactions since the dbspace backup was created.

A Universal Server restore takes data from backups to re-create Universal Server dbobjects in two phases. The first phase is the physical restore, which relies on a backup of the dbobjects. The second phase is the logical restore, which relies on the logical-log file backup.

A restore consists of the following operations:

- One or more physical-restore operations
- A logical-log salvage
- A logical restore

Figure 1-4 depicts a restore.

## Physical Restore

During a physical restore, ON-Bar replaces a lost or corrupted dbobject by copying a backup version of it from secondary-storage media. Figure 1-4 illustrates a physical restore.

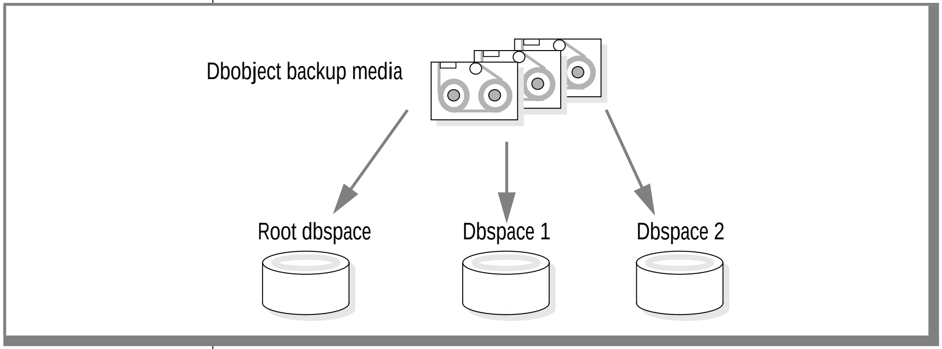


***Figure 1-4***
*Physical Restore*

If Universal Server goes off-line because of a disk failure or corrupted data, it means that a critical dbspace was damaged. You need to perform a cold restore to restore the critical dbspaces. For a description of critical dbspaces, see "The Emergency Boot File" on page 1-6.

If you need to restore any critical dbspace, you must restore all critical dbspaces, starting with a cold restore. For a description of a cold restore, see "A Cold Restore" on page 1-17.

If a disk failure or the corruption of data does not cause Universal Server to go to off-line mode, the damaged dbobjects are not critical and you can restore them individually. For example, if you suffer a disk crash, you can restore to a new disk only those dbspaces with chunks that resided on the failed disk.

## Imported Restore

Sometimes you might want to transfer all of the data from one instance of Universal Server to another. ON-Bar allows you to restore objects to a Universal Server instance that is different from the one from which the data was backed up. However, ON-Bar requires that you use a whole-system backup and a whole-system restore to accomplish the transfer. You must also use compatible versions of XBSA for both operations.

## Logical Restore

During a logical restore, ON-Bar reapplies any transactions that were applied to a dbspace or blobspace since the last backup. By reapplying transactions from the logical log, ON-Bar makes the dbspace as up-to-date as possible.

**Warning:** *If you do not use transaction logging for your databases, ON-Bar can only restore a dbspace up to the time it was most recently backed up. Changes made to data since the last dbspace backup will not be restored on unlogged databases.*

To avoid overwriting the current logical log during a restore, ON-Bar writes to temporary space the logical-log files that it will replay. Therefore, a restore requires enough temporary space to hold the logical log (one set of logical-log files) or the number of logical-log files being replayed, whichever is smaller. For information on how Universal Server looks for temporary space, see the discussion of DBSPACETEMP in the *INFORMIX-Universal Server Administrator's Guide*.

**Warning:** *Make sure that you have enough temporary space for the logical-log portion of the restore; the maximum amount of temporary space that Universal Server needs is the size of the logical log (the size of all the logical-log files).*

## Choosing Between Cold, Warm, or Mixed Restore

When you restore Universal Server data, you must decide whether you will do it while Universal Server is in off-line mode or in on-line mode. This decision is not completely arbitrary, however. It depends in part on the data that you are restoring. The following sections explain the factors that determine which Universal Server mode you should use when you perform a restore.

### *A Cold Restore*

A *cold restore* is a restore that you perform while Universal Server is in off-line mode. It consists of a logical-log file salvage, one or more physical restores, and a logical restore. You must perform a cold restore to restore critical dbspaces. You can restore as many noncritical dbobjects as you deem necessary.

As Figure 1-5 shows, you can restore all the dbobjects that are managed by Universal Server with one physical restore and one logical restore.

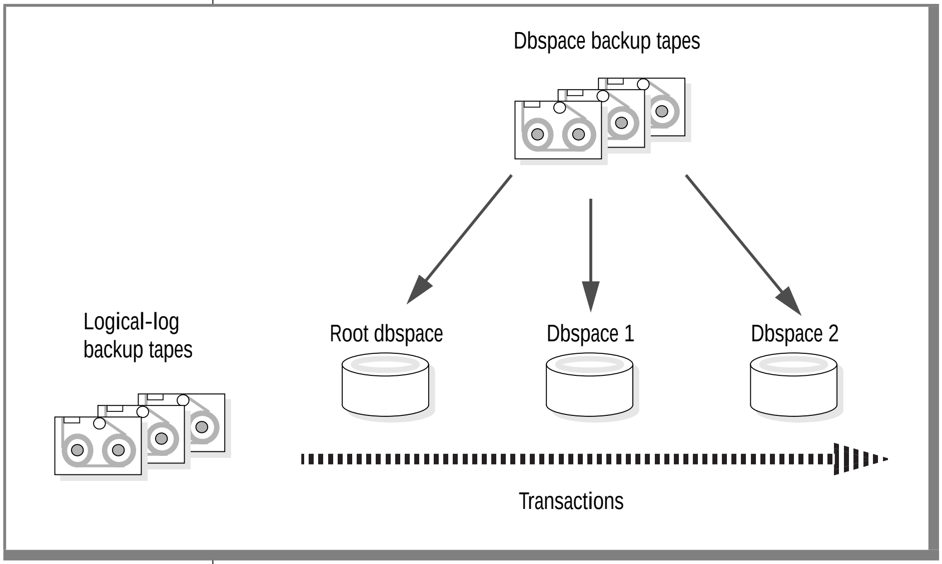


***Figure 1-5***
*Cold Restore*

Universal Server is off-line when you begin a cold restore, but it goes into recovery mode after the reserved pages are restored. From that point on, it stays in recovery mode until either the logical restore is complete (after which it is in quiescent mode) or you use the **onmode** utility to place it in another mode.

You can perform a cold restore on only critical dbobjects and then restore the remaining dbobjects after you bring Universal Server into on-line mode. By doing this, you can decrease the amount of time the database server is inaccessible to users. For a description of this type of restore, see "A Mixed Restore" on page 1-20.

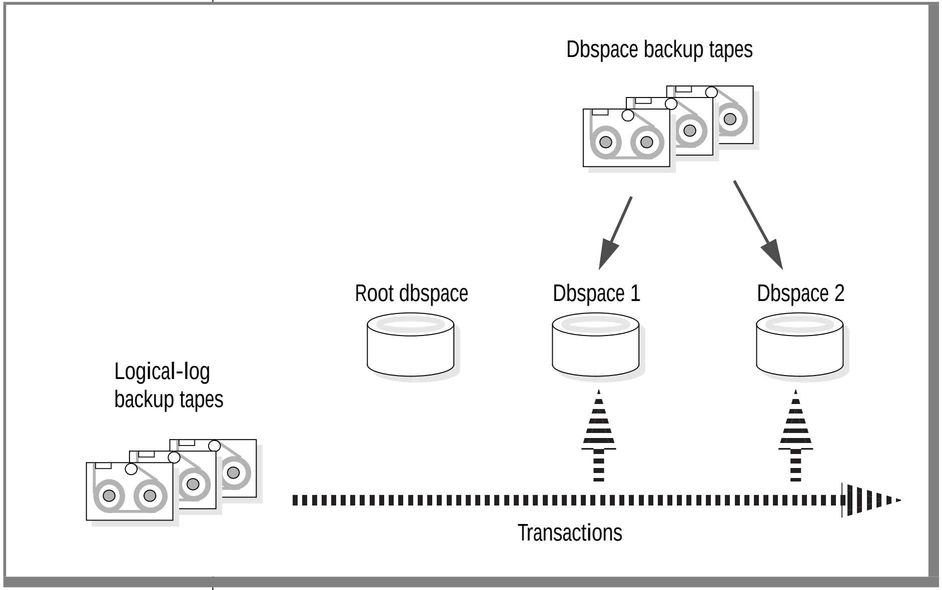
The logical log restore that takes place during a cold restore uses the same disk space to sort logical logs that is devoted to the logical-log files during normal Universal Server processing. Therefore, when the logical-log files are restored, they will overwrite any logical-log files that are left over from normal Universal Server processing. For information on how to avoid overwritten left over logical-log files, see "Salvaging Logical-Log Files" on page 1-14.

After a cold restore, the next backup must be a level-0 backup.

## A Warm Restore

*A warm restore* restores noncritical dbobjects while Universal Server is in on-line or quiescent mode. It consists of one or more physical-restore operations (if you are restoring multiple dbobjects concurrently), a logical-log backup, and a logical restore. Figure 1-6 depicts a warm restore.

**Figure 1-6**
*Warm Restore*



During a warm restore, ON-Bar applies backed-up logical-log files to the restored dbspaces. However, since the database server is in on-line mode, users are generating transactions that are being recorded in the logical-log files. To avoid overwriting the current logical log, ON-Bar writes to temporary space the logical-log files that will be replayed. Therefore, a warm restore requires enough temporary space to hold the logical log (one set of logical-log files) or the number of log files being replayed, whichever is smaller. For information on how Universal Server looks for temporary space, see the discussion of DBSPACETEMP in the *INFORMIX-Universal Server Administrator's Guide*.

## A Mixed Restore

A *mixed restore* is a cold restore followed by a warm restore. A mixed restore restores some dbobjects during a cold restore (while Universal Server is in off-line mode) and some dbobjects during a warm restore (while Universal Server is in on-line mode). If you are doing a restore but you need to provide access to a particular table or set of tables as soon as possible, you might want to do a mixed restore. In this case, you perform a cold restore to restore the critical dbspaces and the dbspaces that contain the important tables. Then you apply the logical-log files to the newly restored dbspaces. Figure 1-7 illustrates the cold portion of a mixed restore.

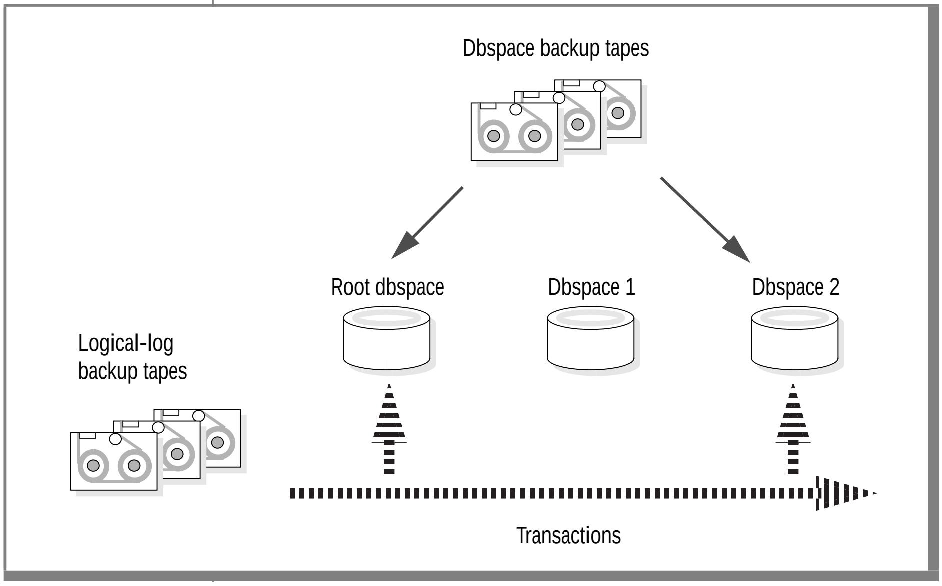


**Figure 1-7**
*Cold Portion of a Mixed Restore*

Following the cold restore, place Universal Server in on-line mode and perform a warm restore to restore the remaining dbobjects. Apply the logical-log files to these dbobjects as well. Figure 1-8 illustrates the warm portion of a mixed restore.

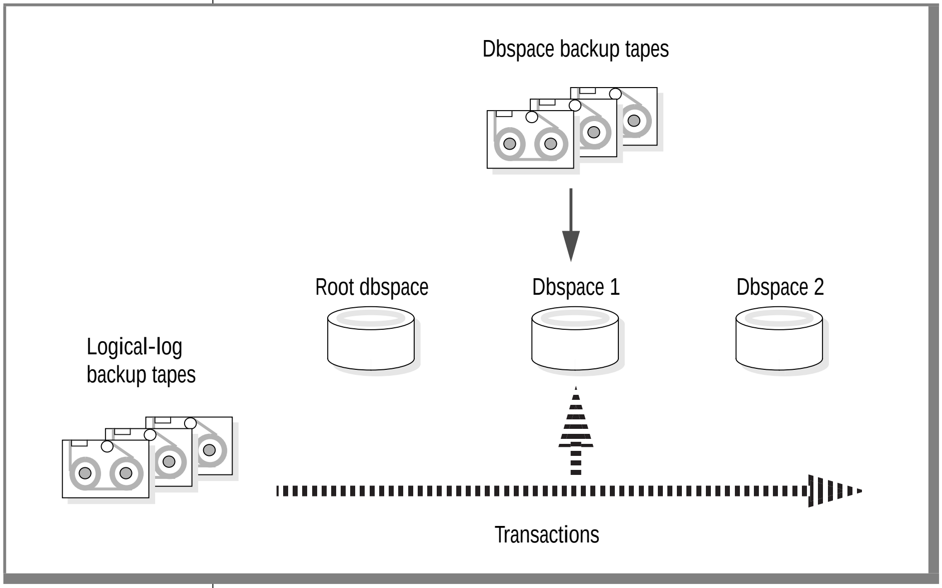


***Figure 1-8***
*Warm Portion of a Mixed Restore*

The dbspaces that are not restored during the cold restore are not available until after they are restored during a warm restore, even though they might not have been damaged by the failure of a critical dbspace.

## Restoring to a Point in Time

You can restore Universal Server data to a specific time instead of restoring data right up to the last complete transaction. Start a point-in-time restore with a physical restore followed by a logical restore of all transactions up to a specific time.

When you restore Universal Server data to a specific time, you restore *all* data up to that time. You cannot restore only a particular dbspace to the specific time. For information on how to perform a restore to a specific time, see Chapter 3, "Configuring ON-Bar."

**Warning:** *When ON-Bar restores data to a specified time, it can only restore non-logging databases to the point of the dbspace backup because there are no logical-log records to restore.*

## Parallel Backups and Restores

Except for whole-system backups and restores, ON-Bar backs up and restores dbspaces concurrently to achieve better performance than it could if it backed up or restored dbspaces and blobspaces serially.

When ON-Bar receives a request, it determines how many objects are involved. If the request involves more than one object, ON-Bar creates a new ON-Bar process for each object up to the limit specified by the BAR_MAX_BACKUP configuration parameter. Each new instance of ON-Bar creates a new XBSA session. For information about setting a limit on the number of ON-Bar processes that can run in parallel, see "BAR_MAX_BACKUP" on page 3-5.

The original ON-Bar process is called the *parent* process, and each additional ON-Bar process that it creates is called a *child* process. The parent process monitors each child process. When all the child processes have completed their work, the parent determines whether an error occurred and returns a status to the process that requested the backup or restore.

Chapter

# Using ON-Bar

2

The first part of this chapter explains what preliminary tasks you need to complete to perform a successful backup. The rest of this chapter explains how you can back up and restore dbspaces and logical-log files. If you need to configure ON-Bar because you have just installed it, skip ahead to Chapter 3, "Configuring ON-Bar." When you finish the configuration, come back to this chapter for information regarding backups and restores.

You must be user **informix** or **root** to execute the commands that are described in this chapter.

## Preliminary Tasks

This section explains the steps that you must take to perform backups. This section does not cover logical-log backups. Logical-log backups are explained in "Backing Up Logical-Log Files" on page 2-8.

Before you can create a backup with ON-Bar, you must install and configure your storage manager as described in your storage-manager manual. Informix recommends that you also consider completing the following tasks before you create your backup:

- Be sure that you have sufficient logical-log space to create a backup.
- Print or keep a copy of essential database-server configuration information.
- Verify data consistency.
- Synchronize with other administrative tasks.

The following sections address these topics.

Depending on your storage manager, you might also need to label media and have an operator feed media to the storage manager. For more information on tasks that you must complete before you back up your data, consult your storage-manager manual.

## Be Sure That You Have Enough Logical-Log Space

If the total available space in the logical log (all the logical-log files) is less than half of a single log file, Universal Server does not create a backup. In this situation, ON-Bar performs the logical-log backup automatically so you can attempt the backup again.

If only one backup device is available, be sure that all of your logical-log files are backed up before you start your backup of dbobjects. This precaution frees as much space as possible in your logical-log files.

## Copy Your Database Server Configuration Information

Make a copy of the current ONCONFIG file, **sqlhosts** file, **oncfg** file, and emergency boot file when you create a level-0 backup. You need this information to restore Universal Server data from the backup media.

## Verify Consistency Before a Level-0 Backup

To ensure the integrity of your backups, periodically verify that all Universal Server data and overhead information are consistent before you create a level-0 backup. You do not need to check for consistency before every level-0 backup. Informix recommends, however, that you do not discard a backup that has been verified for consistency until the next time you verify the consistency of your database. For information on consistency checking, see the *INFORMIX-Universal Server Administrator's Guide*.

### *Choose a Database Server Mode*

You must create backups while Universal Server is in on-line or quiescent mode. Once you start a backup, do not change the mode until the backup finishes; changing the mode terminates your backup.

If you create a backup while Universal Server is in quiescent mode, it is a quiescent backup. Quiescent backups are useful when you want to eliminate partial transactions in a backup.

Quiescent backups might not be practical if users need continuous access to Universal Server databases.

## Synchronize Administrative Tasks with Backups

The following administrative changes require a level-0 backup as part of the procedure. Consider waiting to make these changes until your next regularly scheduled level-0 backup. You must perform a backup:

- after you change logging.
- when you add a dbspace or blobspace, before you restore.
- after you start mirroring for a dbspace that contains logical-log files.
- after you add a logical-log file, to make the log file available.
- after you drop a logical-log file.
- when you move one or more logical-log files, after you drop the old logical-log file and after you add the new logical-log file.
- when you change the size or location of the physical log and after you reinitialize shared memory.
- when you drop a chunk before you can reuse the dbspace that contains that chunk.

You only need to back up the root dbspace (instead of all spaces) when you make the following administrative changes:

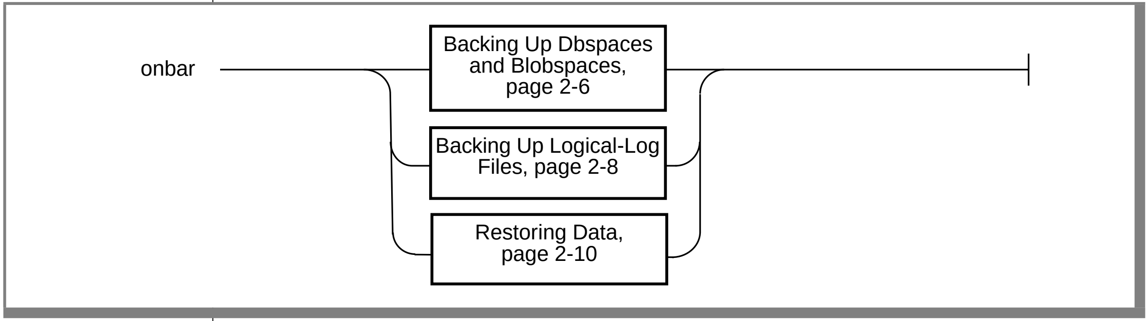
- Add mirroring
- Add a logical-log file
- Change the size or location of the physical log
- Drop a chunk

If you add logging for a database, the logging change takes effect after you back up all dbspaces that contain database data. You do not need to back up all dbspaces at once; Universal Server checks after each backup to see if it can start logging.

## Syntax of the onbar Utility

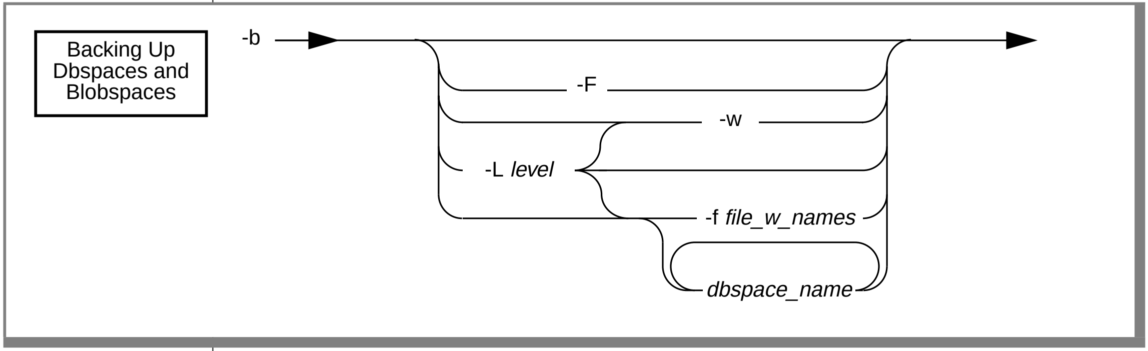
The **onbar** utility provides options that enable you to back up dbobjects and to restore Universal Server data from dbobject back ups.

The following diagram summarizes the **onbar** utility syntax and refers you to the locations of detailed syntax diagrams in this chapter.


onbar
Backing Up Dbspaces and Blobspaces, page 2-6
Backing Up Logical-Log Files, page 2-8
Restoring Data, page 2-10

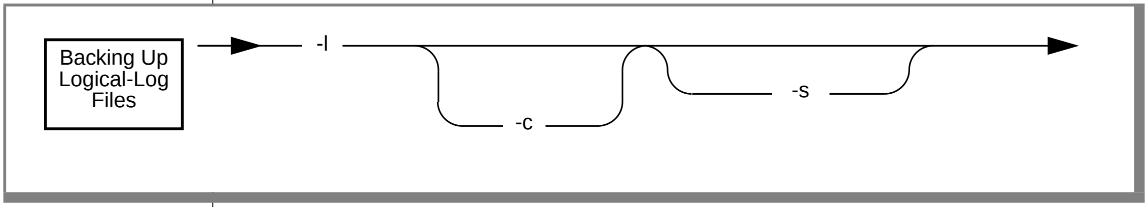
### Backing Up Dbspaces and Blobspaces

Universal Server has to be in on-line or quiescent mode to perform a backup. Only on-line dbspaces and blobspaces are backed up. Temporary dbspaces are skipped. Use the **-d** option of the **onstat** utility to see which dbspaces and blobspaces are on-line.



| Element | Purpose | Key Considerations |
|---|---|---|
| **-b** | Indicates a backup process. | None. |
| **-F** | Performs a simulated backup. | You can execute this option whether or not a storage manager application is running. ON-Bar ignores *dbspace_name* if you specify it. Use simulated backups to facilitate changes in database logging modes; to allow the user to use new logs, chunks, or mirrors without performing a backup; or in special situations when you, the administrator, judge that a backup is not needed. No backup actually occurs, so no restore is possible from a simulated backup. Informix recommends that you use simulated backups sparingly, if at all. For more information on simulated backups, see the *INFORMIX-Universal Server Archive and Backup Guide*. |
| **-w** | Performs a whole system backup | None. |
| **-L** *level* | Indicates the level of backup to perform:<br>■ 0 for a complete backup<br>■ 1 for changes since the last level-0 backup<br>■ 2 for changes since the last level-1 backup<br>The default for *level* is 0. | If you request an incremental backup and the **onbar** utility finds no previous level backup has been performed for a particular dbspace, it does the previous level backup. |
| **-f** *file_w_names* | Backs up the dbspaces or blobspaces that are listed (one per line) in the text file whose pathname is given by *file_w_names*. | Use this option if you have a long list of dbspaces or blobspaces that you do not want to type every time you exercise this option. The filename can be any valid UNIX filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2**), and absolute (**/usr/informix/backup_lists/listfile3**) filenames. |
| *dbspace_name* | Names a space-delimited list of dbspaces or blobspaces that are to be backed up. | If you do not enter a *dbspace_name*, **onbar** backs up all objects that are managed by the database server. |

## Backing Up Logical-Log Files



| Element | Purpose | Key Considerations |
|---|---|---|
| **-l** | Performs a backup of full logical-log files. | None. |
| **-c** | Close and backup the current logical log. | None. |
| **-s** | Salvages any logical logs that are still on disk after an OnLine database server crash. | This option must be used before you replace damaged media. If you are performing a cold restore, **onbar** automatically performs a salvage operation. |

You must back up the logical-log files independently of performing a backup of dbspaces and blobspaces

### *Performing an On-Demand Backup*

To make an on-demand backup of the logical-log files that are full (instead of an automatic backup that takes place every time a logical-log file fills), use the **-l** option as shown in the following example:

```
onbar -l
```

To back up the current logical-log file and switch to the next logical-log file, use the **-c** option, as in the following example:

```
onbar -l -c
```

## *Performing an Automatic Backup*

If you set the ALARMPROGRAM configuration parameter to **$INFORMIXDIR/etc/log_full.sh**, **onbar** performs an automatic backup of your logical logs. Every time Universal Server fills a logical-log file, an event alarm is triggered. The event alarm in turn calls **onbar**, which backs up the full logical-log file.

The configuration parameter ALARMPROGRAM governs whether logical-log backup is on-demand or automatic. If you do not set ALARMPROGRAM, or if you set it to **$INFORMIXDIR/etc/log_full.sh**, **onbar** performs automatic backups.

To disable automatic backups triggered by event alarms, set ALARMPROGRAM to **$INFORMIXDIR/etc/no_log.sh** or any value other than **$INFORMIXDIR/etc/log_full.sh**. Remember, if you turn automatic backups off, it is your responsibility to initiate on-demand back ups of the logical logs as they fill. For information on how to perform on-demand backups, see "Performing an On-Demand Backup" on page 2-8.

***Important:*** *Each time Universal Server backs up a logical-log file, it sends the following message to the ON-Bar and database server message logs:*

```
14:13:05 Logical Log 12 - Backup Started
```

*When Universal Server commits the transaction, it sends the following message to the message log:*

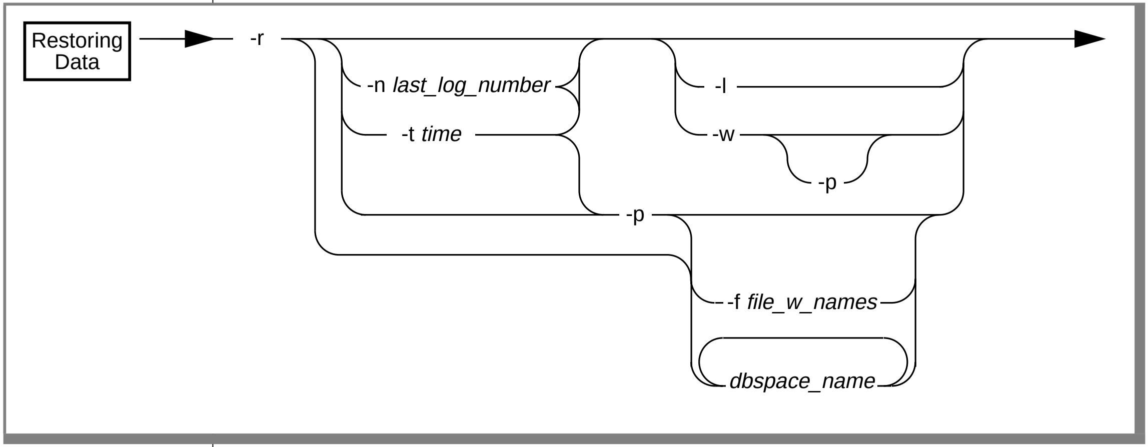
```
14:13:21 Logical Log 12 - Backup Completed
```

*You can use the* ***onstat -l*** *command to verify that the database server has marked the logical-log file as backed up. Once the logical-log file is marked as backed up, it is free for subsequent use when it is needed again. For more information on how to use the* ***onstat*** *utility, see the "INFORMIX-Universal Server Administrator's Guide."*

## Restoring Data

This section explains how to use ON-Bar to perform a physical and logical restore.

Use the **-p** option to perform a physical restore, and use the **-l** option to perform a logical restore. You can perform multiple physical restore operations separately or concurrently (that is, restore multiple dbspaces concurrently) followed by a single logical-restore operation. Keep in mind, however, that a logical restore is required after a physical restore before data is accessible to users. A logical restore can be applied to dbspaces that have been restored over several physical restores.



| Element | Purpose | Key Considerations |
|---|---|---|
| **-r** | Indicates a restore. | None. |
| **-n** *last_log_number* | Indicates the number of the last log to restore. | If any logs exist after this one, ON-Bar does not restore them. |
| **-l** | Indicates a logical restore only. Restores and rolls forward the logical logs. | The logical restore applies only to those dbspaces that have just been physically restored. |
| **-t** *time* | Indicates the time at which ON-Bar is to stop the recovery. | This option can also be used to restore a backup made earlier. |
| **-w** | Performs a whole-system restore. | Searches for the last whole-system backup and restores from that. |

| Element | Purpose | Key Considerations |
|---|---|---|
| **-p** | Indicates a physical restore only. | Must be followed by a logical restore before data is accessible. |
| **-f** *file_w_names* | Restores the dbspaces or blobspaces that are listed (one per line) in the text file whose pathname is given by *file_w_names*. | Use this option if you have a long list of dbspaces or blobspaces that you do not want to type every time you exercise this option. The filename can be any valid UNIX filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2**), and absolute (**/usr/informix/backup_lists/listfile3**) filenames. |
| *dbspace_name* | Names one or more dbspaces or blobspaces to be restored. | If you do not enter a *dbspace_name*, ON-Bar restores all objects for the database server. |

During a cold restore, **onbar** always attempts to salvage any logical logs that are on disk. This salvage attempt takes place even if you use the **-p** option so that the logs are available later during logical restore.

The following sections contain examples of ON-Bar syntax for restoring data.

### Restoring Down Dbspaces and Logical Logs

To restore appropriate logical logs as well as dbspaces and blobspaces that Universal Server marked as down, use the **-r** option as shown in the following example. A list of down dbspaces is not required.

```
onbar -r
```

### Restoring Down Dbspaces and Blobspaces Only

To restore dbspaces and blobspaces that are down without restoring the logical log, use the **-r** and **-p** options as shown in the following example:

```
onbar -r -p
```

### Restoring Logical Logs Only

To restore the appropriate logical logs after restoring dbspaces or blobspaces, use the **-r** and **-l** options as shown in the following example:

```
onbar -r -l
```

## Restoring a Particular Dbspace or Blobspace

To restore a given dbspace or blobspace, for example a blobspace named **my_blobspace** and two dbspaces named **my_dbspace1** and **my_dbspace2**, use the **-r** option as shown in the following example:

```
onbar -r my_blobspace my_dbspace1 my_dbspace2
```

## Salvaging Logical-Log Files

If you experience a media failure and you need to replace the media before you can perform a cold restore, salvage the logical-log files that are still on the media with the following command before you replace the media:

```
onbar -l -s
```

Now replace the physical media and perform a restore with the following command:

```
onbar -r
```

Chapter

# Configuring ON-Bar

3

This chapter describes the ON-Bar configuration tasks. Perform the following tasks before you start ON-Bar:

- Setting ON-Bar Configuration Parameters
- Specifying Automatic Backup of the Logical-Log Files

Of these tasks, configuring the storage manager is the only task that might be required, depending on the storage manager that you have chosen. It is optional to set ON-Bar configuration parameters or specify an event alarm to have **onbar** back up the logical-log files automatically. For more information on automatic and on-demand backups, see "On-Demand and Automatic Backups" on page 1-13.

## Setting ON-Bar Configuration Parameters

You can set the following ON-Bar configuration parameters in the **$INFORMIXDIR/etc/$ONCONFIG** file.

| Parameter | Purpose |
|---|---|
| BAR_ACT_LOG | Specifies the location of the ON-Bar message file. |
| BAR_MAX_BACKUP | Specifies the maximum number of processes per **onbar** command. |
| BAR_NB_XPORT_COUNT | Specifies the number of data buffers for each **onbar** process. |

(1 of 2)

| Parameter | Purpose |
|---|---|
| BAR_RETRY | Specifies how many times **onbar** should retry a backup or restore operation if the first attempt fails. |
| BAR_XFER_BUF_SIZE | Specifies the size of buffers that are used to exchange data with Universal Server. |
| BAR_BSALIB_PATH | Specifies the path of the storage-manager library. |

(2 of 2)

These configuration parameters are optional. They allow you to specify the location of ON-Bar files and set limits that can affect performance. If you do not set a parameter, **onbar** automatically sets it to its default value. For information about setting parameters in the ONCONFIG file, see the *INFORMIX-Universal Server Administrator's Guide*.

## BAR_ACT_LOG

| | |
|---|---|
| *default value* | **/tmp/bar_act.log** |
| *takes effect* | When ON-Bar starts |
| *refer to* | Appendix A |

The BAR_ACT_LOG parameter specifies the full pathname of the ON-Bar message file. The **onbar** utility periodically writes a brief description of what it is doing to the message file. The format of the file resembles the format of the Universal Server message log. You can examine the message file to determine the results of **onbar** actions.

## BAR_MAX_BACKUP

| | |
|---|---|
| *default value* | Unlimited = 0 |
| *units* | **onbar** processes |
| *range of values* | >= 0 to unlimited |
| *takes effect* | When **onbar** starts |

The BAR_MAX_BACKUP parameter specifies the maximum number of parallel processes that are allowed for each **onbar** command. For example, if you set BAR_MAX_BACKUP to 5 and execute two **onbar** commands, the maximum number of processes that **onbar** will run concurrently is ten.

The **onbar** utility ignores the BAR_MAX_BACKUP parameter for a whole-system backup or restore.

Use the BAR_MAX_BACKUP parameter to limit the computer resources that **onbar** uses. If you do not set BAR_MAX_BACKUP or set it to 0, the number of **onbar** processes is limited only by the number of dbobjects, whichever is less.

## BAR_NB_XPORT_COUNT

| | |
|---|---|
| *default value* | 10 |
| *units* | Buffers |
| *range of values* | 3 to unlimited |
| *takes effect* | When **onbar** starts |

The BAR_NB_XPORT_COUNT parameter specifies the number of data buffers that each **onbar** process can use to exchange data with Universal Server. The value of this parameter affects **onbar** performance. For example, if you set BAR_MAX_BACKUP to 5 and BAR_NB_XPORT_COUNT to 5 and subsequently issue 5 **onbar** commands, the resulting 25 child ON-Bar processes will use a total of 125 buffers.

## BAR_RETRY

| | |
|---|---|
| *default value* | BAR_CONT |
| *range of values* | BAR_ABORT, BAR_CONT, or `n` |
| *takes effect* | When **onbar** starts |

The BAR_RETRY parameter specifies how many times **onbar** should retry a backup or restore operation if the first attempt fails. The possible settings for the BAR_RETRY parameter affect **onbar** behavior in the following ways:

- If set to BAR_ABORT, **onbar** aborts the backup or restore attempt when an error occurs, returns an error, and quits.
- If set to BAR_CONT, **onbar** aborts the backup or restore attempt for that particular dbobject, returns an error, and attempts to back up or restore any dbobjects that remain.
- If set to a specific number (`n`), **onbar** attempts to back up or restore this object the specified number of times before it gives up and moves on to the next object.
- If the backup or restore of a logical-log file fails, **onbar** aborts the attempt regardless of the setting.

## BAR_XFER_BUF_SIZE

| | |
|---|---|
| *default value* | `15` when the PAGESIZE is `4` k<br>`31` when the PAGESIZE is `2` k |
| *units* | PAGESIZE |
| *range of values* | > `0` (absolute range is `1` to `1,000,000` pages) |
| *takes effect* | When **onbar** starts |

The BAR_XFER_BUF_SIZE parameter specifies the size of each transfer buffer. The actual size of a transfer buffer is BAR_XFER_BUF_SIZE * PAGESIZE + 500 bytes. For example, if BAR_XFER_BUF_SIZE is `15`, the transfer buffer should be 61,940 bytes.

### BAR_BSALIB_PATH

| | |
|---|---|
| *default value* | **/usr/lib/ibsad001.xx** |
| *takes effect* | When **onbar** starts |

You usually need to configure your storage manager before you can use **onbar** to perform a backup or restore.

The **onbar** utility and the storage manager rely on a shared library to integrate with each other. Configure the BAR_BSALIB_PATH configuration parameter for the library of your storage manager. The default pathname of BAR_BSALIB_PATH is **/usr/lib/ibsad001.xx**; where **xx** is the shared library file extension. For example, / **usr/lib/ibsad001.so** is for Solaris. You can place the storage manager library in any directory that you choose and create a link to it from **/usr/lib/ibsad001.xx**.

## Specifying Automatic Backup of the Logical-Log Files

The **onbar** utility provides a shell script called **log_full.sh** that you can use to start backing up logical-log files when Universal Server issues a log-full event alarm.

Set the ALARMPROGRAM configuration parameter to **$INFORMIXDIR/etc/log_full.sh**.

If you do not want logical logs to be backed up automatically, set the ALARMPROGRAM configuration parameter to **$INFORMIXDIR/etc/no_log.sh**. For more information on the ALARMPROGRAM configuration parameter, see the *INFORMIX-Universal Server Administrator's Guide*.

***Important:*** *When you choose automatic backups, backup media should always be available for the backup process.*

Chapter

# Catalog Tables

4

This chapter describes the ON-Bar system catalog tables. You can query the catalogs for backup and restore data to evaluate performance or identify object instances for a restore.

## The bar_action Table

The **bar_action** table lists all backup and restore actions that have been attempted against an object. Use the information in this table to track objects that have been backed up.

| Column Name | Type | Explanation |
|---|---|---|
| **act_aid** | INTEGER | Action identifier. A unique number within the table. Can be used with **act_oid** to join with the **bar_instance** table. |
| **act_end** | DATETIME YEAR TO SECONDS | The date and time when the action finished. |
| **act_oid** | INTEGER | Object identifier. Identifies the backup object against which a backup or restore attempt was made. Can be used with **act_aid** to join with **bar_instance.** The **act_oid** column of the **bar_action** table equals the **obj_oid** column of the **bar_object** table. |

(1 of 2)

| Column Name | Type | Explanation |
|---|---|---|
| **act_start** | DATETIME YEAR TO SECONDS | The date and time when the action began. |
| **act_type** | SMALLINT | Identifies the action that was attempted: 1 for backup, 2 for restore, 3 for a foreign or imported restore, 4 for a simulated backup. |
| **act_status** | INTEGER | Identifies the action that was attempted: 1 for backup, 2 for restore, 3 for a foreign or imported restore, 4 for a simulated backup, 5 for a whole-system backup, 6 for a whole-system restore. |

(2 of 2)

## The bar_instance Table

ON-Bar writes a record to the **bar_instance** table for each successful backup. The table describes each object that was backed up. ON-Bar might later use the information for a restore operation.

| Column Name | Type | Explanation |
|---|---|---|
| **ins_aid** | INTEGER | Action identifier. Identifies the successful action that created this instance of the backup object. Combined with **ins_oid**, can be used to join with the **bar_action** table. |
| **ins_copyid_hi** | INTEGER | The high bits of the instance copy identifier. Combined with **ins_copyid_lo**, it is a unique value that is assigned by the storage manager to link the ON-Bar object identifier with the storage-manager object identifier. |
| **ins_copyid_lo** | INTEGER | The low bits of the instance copy identifier. Combined with **ins_copyid_hi**, it is a unique value that is assigned by the storage manager to link the ON-Bar object identifier with the storage-manager object identifier. |

(1 of 2)

| Column Name | Type | Explanation |
|---|---|---|
| **ins_level** | SMALLINT | Level of the backup action: 0 for a complete backup, 1 for a backup of any changes to this object since its last level-0 backup, 2 for a backup of any changes since the last level-1 backup. Always 0 for logical-log backups. |
| **ins_time** | INTEGER | Time stamp. Universal Server uses this value when it creates the next-level backup. |
| **ins_version** | CHAR(18) | ON-Bar version that created this instance. Indicates compatibility among ON-Bar versions, storage managers, and versions of XBSA. Can be used to join with the **bar_version** table. |
| **ins_oid** | INTEGER | Object identifier. Identifies the affected object. Can be used to join with the **bar_object** table. Combined with **ins_aid**, can be used to join with the **bar_action** table. |
| **ins_first_log** | INTEGER | Action identifier. Identifies the successful action that created this instance of the backup object. Combined with **ins_oid**, can be used to join with the **bar_action** table. |

(2 of 2)

## The bar_version Table

The **bar_version** table lists the compatible versions of ON-Bar, XBSA, and storage managers.

| Column Name | Type | Explanation |
|---|---|---|
| **bar_sm** | CHAR(18) | The name of the storage manager. |
| **bar_version** | CHAR(18) | The version of ON-Bar. Can be used to join with the **bar_instance** table. |
| **bsa_version** | CHAR(18) | The version of XBSA. |
| **sm_version** | CHAR(18) | The version of the storage manager. |

## The bar_object Table

The **bar_object** table describes each backup object.

| Column Name | Type | Explanation |
|---|---|---|
| **obj_srv_name** | CHAR(18) | The user name for the object, for example, **dbs1**, **log00000010** or **foo**. |
| **obj_oid** | SERIAL | The object identifier. A unique number within the table. This table is a list of all database objects from each Universal Server for which at least one backup attempt has been made. Can be used to join with the **bar_action** and **bar_instance** tables. |
| **obj_type** | CHAR(2) | Backup object type where:<br>B = blobspace<br>CD = critical dbspace<br>L = logical log<br>ND = noncritical dbspace<br>R = rootdbs |
| **obj_name** | CHAR(18) | Used in a multi-server, distributed system to ensure objects are restored to the correct server. Used when multiple database servers are on the node to ensure that objects are restored in the database server instance to which the object belongs. |

## The bar_server Table

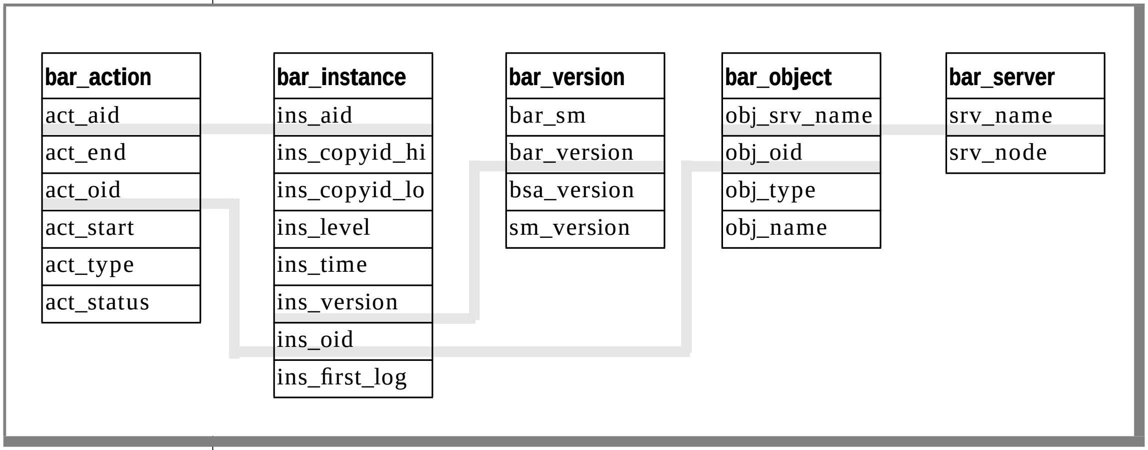
The **bar_server** table lists the database servers in an installation. This table is used to ensure that backup objects are returned to their proper places during a restore. This table is built from **$INFORMIXSQLHOSTS**.

| Column Name | Type | Explanation |
|---|---|---|
| **srv_name** | CHAR(18) | Database server name. Matches **$INFORMIXSERVER.** |
| **srv_node** | CHAR(18) | Name of the node where the database server resides. |

## ON-Bar Catalog Map

Figure 4-1 maps the ON-Bar tables. The grey lines show the referential constraints between tables. Read from the left to the right, the data needs to be present in the first table before you can insert it into the second table. For example, consider the **bar_object** and **bar_server** tables. Reading from the left to the right, the **bar_server** table is first and the **bar_object** table is second. If you are trying to insert data into the **obj_srv_name** column of the **bar_object** table, there has to be a matching name in the **srv_name** column of the **bar_server** table.

**Figure 4-1**
*ON-Bar Catalog Map*



## The Catalog Tables and the Emergency Boot File

The emergency boot file resides in the **$INFORMIXDIR/etc** directory and is called **ixbar.*server_id***, where ***server_id*** is the value of the SERVERNUM configuration parameter. Each line of the emergency boot file corresponds to one backup object.

# !ON-Bar Messages

This appendix describes the ON-Bar message file and the ON-Bar messages, which include informational messages, warnings, and error messages.

You can find descriptions of Informix product error messages in the *Informix Error Messages* manual. You can also use the **finderr** utility program, which is described in that same manual, to view Informix error messages on your terminal or workstation.

## The ON-Bar Message File

ON-Bar writes informational messages, warnings, and error messages to the ON-Bar message file with the exception of messages 43013, 43014, 43016, and 43039, which are written to standard error. The ON-Bar message file is intended to help you determine whether a backup or restore attempt succeeded. The ON-Bar message file also records approximately how long an operation took and lists the objects that ON-Bar backed up or restored.

The default location and name of the ON-Bar message file is **/tmp/bar_act.log**. You can specify a different location and name for the ON-Bar message file by setting the BAR_ACT_LOG configuration parameter. For information on setting the BAR_ACT_LOG configuration parameter, see "BAR_ACT_LOG" on page 3-4.

## About ON-Bar Messages

This section explains how to read and interpret messages in the ON-Bar message file.

### Message Format

A message in the ON-Bar message file has the following format:

```
    timestamp_process_idparent_process_idmessage
```

The *timestamp* provides the date and time when ON-Bar wrote the message. The *process id* is the number that the operating system uses to identify this instance of ON-Bar. The *parent process id* is the number that the operating system uses to identify the process that executed this instance of ON-Bar. The *message* field contains the ON-Bar message. The following example illustrates a typical entry in the ON-Bar message file:

```
  Jan 17, 1995 10:09:591217 1259 43046 Unable to connect to database server
```

## ON-Bar Messages

43002 No more available memory.

Attempt to allocate memory failed. Ask your System Administrator to either increase your swap space or to install more memory in your system.

43003 WARNING: Can’t build where clause of *query* because there is no data.

No data was passed to the build-where-clause function, so no where clause can be built. The query will proceed without a where clause and will affect all rows in the table.

43004 ERROR: *where_clause* for *query* exceeds it’s maximum allowed length of *maximum_length* characters.

No data was passed to the build-where-clause function, so no where clause can be built. The query will proceed without a where clause and will affect all rows in the table.

43005 WARNING: No data to insert into *table_name.*

No data was passed to the insert function, so no insert was attempted.

43006 ERROR: Unable to convert datetime to string *ESQL_return_value.*

Date string is in an invalid format. Consult your Informix manual for the proper ANSI-style date format.

43007 ERROR: *Data* required to insert a row into *table_name.*

An insert into this table cannot happen without the specified data. Verify that the required data exists before attempting another insert in this table.

43008 ERROR: Failed to build where clause for *query.*

Attempt to build a where clause for the specified query failed. Verify that the data needed to create the where clause exists.

43009 WARNING: Failed to add selected row to linked list for *query.*

Attempt to add the selected row to the linked list failed. Reenter the indicated row.

43010 ERROR: Missing data for *table_name*.

Required data is missing. Verify that the data exists.

43011 ERROR: Updates to *table_name* primary key are not allowed.

Updating the primary key for a table is not allowed. First, delete the row and then insert a new row with the new primary key.

43012 ERROR: Unable to open connection to server.

The database server is in an incorrect state. Bring the server to the correct state. For a backup, the server should be in on-line or quiescent mode. For a warm restore, the server should be in on-line, quiescent, backup, or recovery mode. For a cold restore, the server should be off-line. This can be done with the **onmode** or **oninit** commands.

43013 WARNING: Physical restore complete. Logical restore required before work can continue.

A logical restore must be performed to bring all restored dbspace and blobspaces to a consistent state. Perform a logical restore.

43014 ERROR: Unable to read $ONCONFIG parameters.

The ONCONFIG file is inaccessible. It may be missing or have incorrect permission values. Verify that an ONCONFIG file exists and that its permissions are correct. See the *INFORMIX-Universal Server Administrator's Guide* for details.

43015 ERROR: Unable to set INFORMIXSHMBASE.

Unable to attach to shared memory. Contact your database administrator.

43016 Shared memory not initialized for INFORMIXSERVER *servername*.

Database server is not running. Start up a database server.

43017 Running as Informix for testing.

43018 ERROR: Must be user root or informix.

Only users **informix** and **root** are allowed to execute ON-Bar. Log in as **informix** or **root** before you attempt the backup, restore, or database logging mode change.

43019 ERROR: User is not a member of the Informix-Admin group.

Only users listed in the **Informix-Admin Group** are allowed to execute ON-Bar. Ask your system administrator to add your user name to the **Informix-Admin Group**.

43020 ERROR: Unable to set process group ID.

43021 ERROR: Unrecognized command *command*.

43022 Unable to open file *filename*.

The file or its directory permissions prevent it from being created or opened. Verify the permissions on the file and its directory.

43023 ERROR: Invalid serial number. Please consult Installation Instructions.

43024 WARNING: Unable to read backup level, defaulting to level 0.

The backup level entered on the command line is not valid. Verify that the backup level is correct and retry the command.

43025 WARNING: Unable to read logical-log ID.

The logical-log ID entered on the command line is not valid. Verify that the logical-log ID is correct and retry the command.

43026 WARNING: Unable to read backup / restore session IDs.

43027 WARNING: DB / BLOBspace *dbspace_name / blobspace_name* is online and won't be restored.

43028 WARNING: The maximum allowed number of DB / BLOBspaces per ON-Bar command has been exceeded. The last dbspace or blobspace is *dbspace_name / blobspace_name*.

The backup or restore of all dbspaces and blobspaces up to and including the one specified will occur. Issue a new ON-Bar command for the remaining dbspaces and blobspaces.

43029 WARNING: The maximum allowed number of logical-log stream IDs per ON-Bar command has been exceeded. The last log stream ID is *log stream ID*.

The backup of all logical-log streams up to and including the one specified will occur. Issue a new ON-Bar command for the remaining log streams.

43030 WARNING: *dbspace / blobspace* does not have a previous level backup. Defaulting to level *level* backup.

43031 ERROR: Unable to start the DB/BLOBspace backup: *dbspace_name / blobspace_name*.

43032 ERROR: Unable to get backup data from the database server: *servername*.

43033 ERROR: Unable to commit the backup: *dbspace_name / blobspace_name*.

43034 ERROR: Unable to update in_time to *numeric_value* for in_aid *numeric_value*.

Failure to update the *bar_instance* table. Ask your database administrator to repair the data.

43035 ERROR: Unable to start the logical-log backup: *dbspace_name / blobspace_name*.

43036 ERROR: Unable to get backup data from the database server: *servername*.

43037 ERROR: Unable to commit the backup: *dbspace_name / blobspace_name*.

43038 ERROR: simulated backup failed. *dbspace_name / blobspace_name.*

43039 ERROR: Version *version number* of the XBSA shared library is not compatible with version *version number* of ON-Bar.

Either the XBSA shared library provided by the storage management vendor has not been certified by Informix or there was an error during installation of ON-Bar. Verify that ON-Bar was installed properly. Verify that the XBSA library has been certified.

43040 ERROR: DB/BLOBspace *dbspace_name / blobspace_name* does not exist.

Verify that the dbspace or blobspace exists in this database server.

43041 ERROR: Unable to determine if *dbspace_name* is critical media or not.

43042 ERROR: Unable to start the logical restore: *dbspace_name*.

43043 ERROR: Must restore logical logs from *date_time* or later.

User wishes to stop the restore at a logical log that is too early. A dbspace or blobspace backup occurred after the log specified by the user. Retry the restore up to the specified logical log or later.

43044 ERROR: Unable to write restore data to the database server: *servername*.

43045 ERROR: Unable to commit the restore: *dbspace_name* / *blobspace_name*.

43046 ERROR: Unable to start the physical restore: *dbspace_name* / *blobspace_name*.

43047 ERROR: Cannot warm restore-critical media: *dbspace_name*.

User wishes to stop the restore at a logical log that is too early. A dbspace or blobspace backup occurred after the log specified by the user. Retry the restore up to the specified logical log or later.

43048 WARNING: All DB / BLOBspaces are online, so no restore is needed.

43049 WARNING: Exceeded maximum allowed buffer size. Changing buffer size to *buffersize*.

43050 Begin cold level *restore_level* restore *dbspace_name* / *blobspace_name*.

43051 Completed cold level *restore_level* restore *dbspace_name* / *blobspace_name*.

43052 Begin simulated backup.

43053 Completed simulated backup.

43054 Begin level *backup_level* backup *dbspace_name* / *blobspace_name*.

43055 Completed level *backup_level* backup *dbspace_name* / *blobspace_name*.

43056 Begin salvage of logical logs.

43057 Completed salvage of logical logs.

43058 Begin warm level *restore_level* restore *dbspace_name* / *blobspace_name*.

43059 Completed warm level *restore_level* restore *dbspace_name* / *blobspace_name*.

43060 Begin backup logical log *logical_log_filename*.

43061 Completed backup logical log *logical_log_filename*.

43062 Begin restore logical log *logical_log_filename*.

43063 Completed restore logical log *logical_log_filename*.

43064 Successfully connected to Storage Manager.

43065 Process *process_ID* successfully forked.

43066 Process *process_ID* completed.

43067 Active object does not exist. Attempt to deactivate it failed.

No active object matched the name that was specified for a BSADeactivate-Object( ) call. Refer to the storage manager manual for information on active and inactive objects.

43068 A system error occurred. Aborting XBSA session.

A system error prevents further processing. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43069 Attempt to authorize *user_name* failed.

Verify that the user is **informix** or **root**. Login as **informix** or **root** and try again.

43070 Invalid XBSA function call sequence.

The sequence of XBSA function calls is out of order. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43071 Invalid XBSA session handle *handle_ID.*

An XBSA session handle has been previously closed or corrupted. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43072 XBSA buffer is too small for the object.

The storage manager has a problem. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43073 Description of the object exceeds the maximum allowed value: *maximum_ value.*

Shorten the description of the object and retry.

43074 Database server name exceeds maximum allowed size *maximum_ value.*

Shorten the name of the database server and retry.

43075 The new security token name is invalid.

The storage manager has a problem. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43076 Invalid vote value: Must be BSA_Vote_COMMIT or BSA_Vote_ABORT.

It is unclear whether the transaction should be committed or aborted.

43077 Invalid environment keyword.

Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43078 That object already exists.

An attempt was made to create an object twice. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43079 A new security token must be created.

Create a new security token. Refer to the storage manager manual for instructions.

43080 *dbspace* / *blobspace* does not exist.

43081 Exceeded available resources.

All backup and / or restore resources are in use. Wait until a previous backup and / or restore session is complete and retry.

43082 A DataBlock pointer is required.

An attempt was made to backup and / or restore with no data. Specify data and try again.

43083 An object name is required.

An attempt was made to backup and / or restore an unnamed object. Name the object and retry.

43084 Unable to access null pointer.

A required value was set to null. Reset the value and try again.

43085 Rule ID is required.

A required value was set to null. Create an ID for the rule and retry. Refer to the storage manager manual for instructions.

43086 The object is not empty.

43087 There is no backup copy of *dbspace_name* / *blobspace_name*.

An attempt was made to restore an object that has not been backed up.

43088 Object information data exceeds maximum allowed size *maximum_size*.

Shorten information data for the object and retry.

43089 Object name exceeds maximum allowed size *maximum_size*.

Shorten name of the object and retry. Refer to the storage manager manual for instructions.

43090 Operation is not authorized for *user_ID*.

The specified user does not have permission to perform this operation. Ask your system administrator to change your permissions.

43091 A value for the old security token is required.

Fill in the old security token and retry.

43092 The security token has expired. Please create a new one.

The security token is stale. Create a new security token and retry. Refer to the storage manager manual for instructions.

43093 The transaction was aborted.

An error caused the backup and / or restore transaction to abort. Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43094 A quote is missing from an environment keyword.

Insert the missing quote and retry.

43095 A username cannot be deleted while it owns objects.

Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43096 An unspecified XBSA error has occurred: *numeric_value*.

Refer to the storage manager activity log (or equivalent) for a more detailed description of the problem.

43097 ERROR: There are no DB/BLOBspaces to backup/restore.

43098 WARNING: All DB/BLOBspaces are temporary, so no backup/restore is needed.

43099 ERROR: Unable to set PDQPRIORITY.

An attempt to change the value of PDQPRIORITY failed.

43100 The -f command is ignored for whole system backup/restore and fake backup.

43101 No filename was entered, performing a backup of all DB/BLOBspaces instead.

No filename was included with the -f command.

43102 WARNING: Setting backup level to 0 for a fake backup.

Only level 0 is supported for fake backup.

43103 WARNING: DB/BLOBspace names ignored for fake backup or whole system backup/restore.

A fake backup or whole system restore backs up/restores all DB/BLOBspaces.

43104l WARNING: Linked list operation failed.

A linked list operation failed.

43106 ERROR: One or more BLOBspaces are down. Log backup has been aborted. A BLOBspace is down. Backing up or salvaging the logical logs would make it impossible to restore this BLOB in the future.

Bring all BLOBspaces online and retry the logical log backup or salvage.

43107 ERROR: Unable to start logical log salvage from server.

43108 ERROR: Unable to get logical log salvage data from disk.

43109 ERROR: Unable to commit the backup from server.

# Index