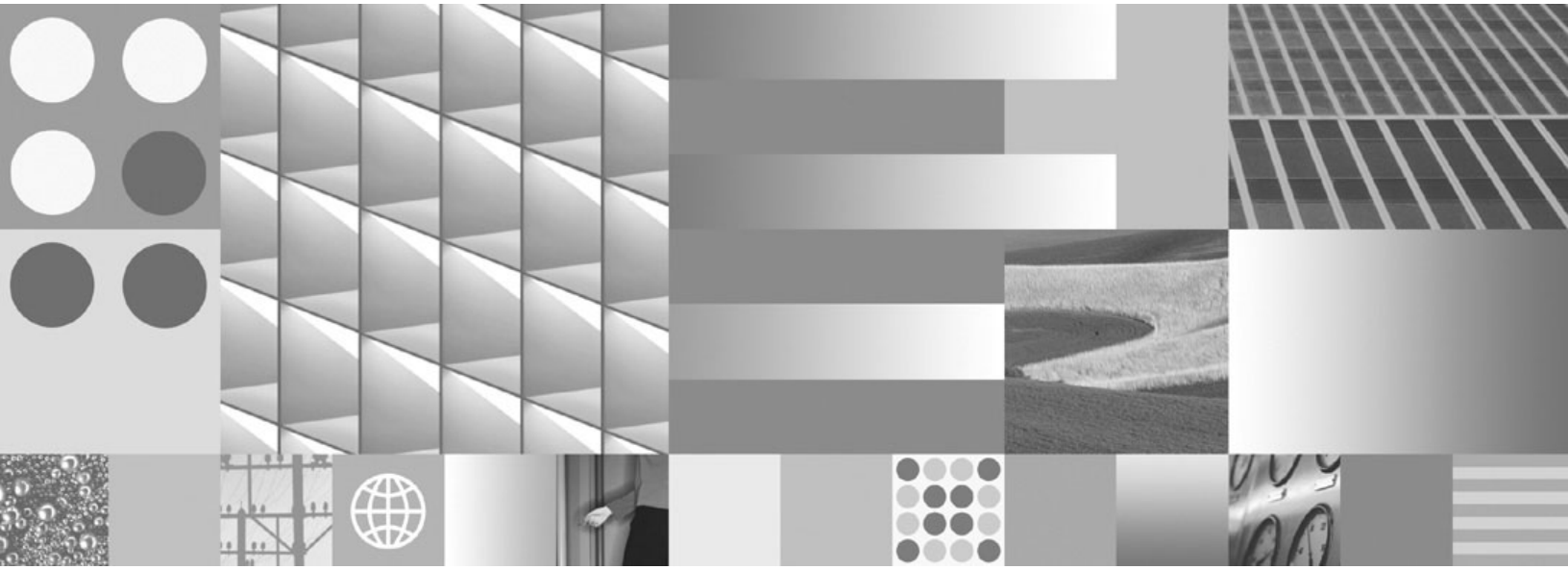
IBM Informix

Version 11.50

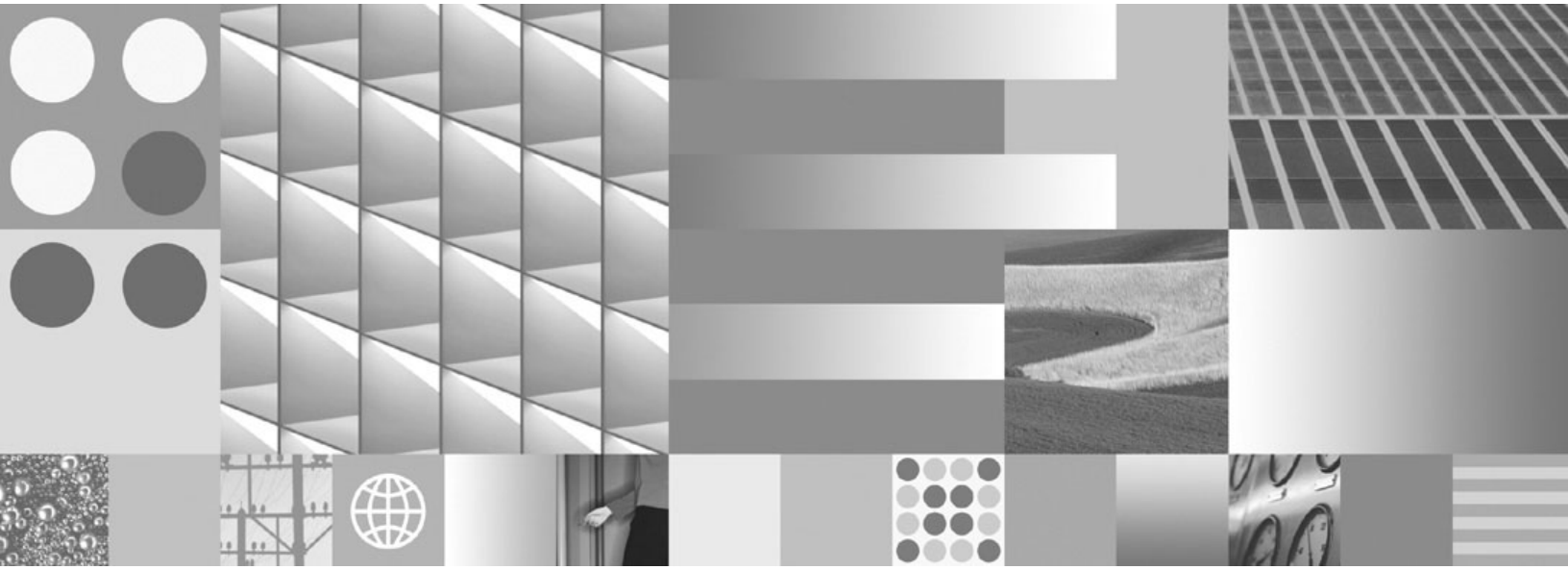


IBM Informix Backup and Restore Guide

SC23-7756-02

IBM Informix

Version 11.50



IBM Informix Backup and Restore Guide

SC23-7756-02

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

This edition replaces SC23-7756-00.

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

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

© Copyright International Business Machines Corporation 1996, 2008.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Introduction xi
In This Introduction xi
About This Manual xi
Types of Users xi
Software Dependencies xii
Assumptions About Your Locale xii
Demonstration Database xii
Documentation Conventions xiii
Typographical Conventions xiii
Feature, Product, and Platform Markup xiv
Example Code Conventions xiv
Additional Documentation xv
Compliance with Industry Standards xv
Syntax Diagrams xv
How to Read a Command-Line Syntax Diagram xvi
Keywords and Punctuation xvii
Identifiers and Names xvii
How to Provide Documentation Feedback xviii

Part 1. Overview of Backup and Restore

Chapter 1. Backup and Restore Concepts 1-1
In This Chapter 1-1
Recovery System 1-1
Backup Systems 1-1
Logical-Log Backup 1-2
Restore Systems 1-4
Comparing ON-Bar and ontape 1-6

Chapter 2. Planning for Backup and Restore 2-1
In This Chapter 2-1
Planning a Recovery Strategy 2-1
Types of Data Loss 2-1
Determining Failure Severity 2-1
Data Use Determines Your Backup Schedule 2-2
Scheduling Backups 2-2
Security Requirements for Label-based Access Control 2-3
Planning a Backup System for a Production Database Server 2-3
Evaluating Hardware and Memory Resources 2-3
Evaluating Backup and Restore Time 2-3
Evaluating Logging and Transaction Activity 2-4
Transforming Data with External Programs 2-4

Part 2. ON-Bar Backup and Restore System

Chapter 3. Overview of the ON-Bar Backup and Restore System 3-1
In This Chapter 3-1
Where to Find Information on Tasks for ON-Bar, ISM, and TSM 3-2
ON-Bar Components for Dynamic Server 3-4
ON-Bar Components for Extended Parallel Server 3-5
Database Server and Storage-Manager Communication 3-5
Backup Scheduler 3-6
ON-Bar Utilities 3-7
IBM Informix Storage Manager 3-7

© Copyright IBM Corp. 1996, 2008

IBM Tivoli Storage Manager . . . 3-8
Third-Party Storage Managers . . . 3-8
XBSA Interface . . . 3-9
ON-Bar Tables . . . 3-9
ON-Bar Boot Files . . . 3-10
ON-Bar Activity Log . . . 3-11
Transform Data with External Filter Programs . . . 3-12

Chapter 4. Configuring the Storage Manager and ON-Bar . . . 4-1
In This Chapter . . . 4-1
Configuring a Storage Manager . . . 4-1
Configuring a Third-Party Storage Manager . . . 4-1
Configuring ISM . . . 4-2
Configuring TSM . . . 4-2
Updating the sm_versions File . . . 4-4
Configuring Multiple Storage Managers on Coserver Nodes (XPS) . . . 4-5
Validating Your Storage Manager . . . 4-6
Configuring ON-Bar . . . 4-6
Creating the bargroup Group (UNIX) . . . 4-6
Your Customized onbar Script is Saved on New Installations . . . 4-6
Setting ISM Environment Variables and ONCONFIG Parameters . . . 4-7
Setting the Interface for TSM Environment Variables . . . 4-7
Specifying the Location of the XBSA Library . . . 4-8
Specifying ON-Bar Configuration Parameters . . . 4-9
ON–Bar Configuration Parameters on Dynamic Server . . . 4-9
Parameters on Extended Parallel Server . . . 4-10
Verifying the Configuration . . . 4-15
Choosing Storage Managers and Storage Devices . . . 4-15
Features That ISM Supports . . . 4-16
Features That ISM Does Not Support . . . 4-16
Features That TSM Supports . . . 4-16
Storage Device Requirements . . . 4-17
Considerations for Extended Parallel Server . . . 4-17

Chapter 5. Backing Up with ON-Bar . . . 5-1
In This Chapter . . . 5-2
Summary of ON-Bar Tasks . . . 5-2
Preparing for a Backup . . . 5-4
Data ON-Bar Backs Up . . . 5-4
Administrative Files to Back Up . . . 5-4
Installing and Configuring a Storage Manager . . . 5-5
Whole-System Backup (IDS) . . . 5-5
Parallel Backup . . . 5-5
Standard Backup . . . 5-5
Incremental Backup . . . 5-6
Physical Backup . . . 5-6
Choosing a Backup Level . . . 5-6
Collecting Information About Your System Before a Backup . . . 5-7
Backing Up Storage Spaces and Logical Logs . . . 5-8
Backup Syntax . . . 5-8
Backing Up After Changing the Physical Schema . . . 5-10
Using ISM During a Backup . . . 5-11
Using IBM Server Administrator to Back Up and Verify . . . 5-12
ON-Bar Backup Examples . . . 5-12
Backing Up Table Types . . . 5-17
Viewing Recent ON-Bar Activity (IDS) . . . 5-17
Backing Up Logical Logs . . . 5-18
Backing Up Logical Logs on Dynamic Server . . . 5-18
Backing Up Logical Logs on Extended Parallel Server . . . 5-22
Monitoring Logical-Log Backups . . . 5-23

Salvaging Logical-Log Files 5-24
Understanding ON-Bar Backup Processes 5-24
Backup Sequence on Dynamic Server 5-24
Backup Sequence on Extended Parallel Server 5-26

Chapter 6. Verifying Backups 6-1
In This Chapter 6-1
archecker Overview 6-1
Verification Process with archecker 6-2
Verifying Backups with archecker 6-3
Estimating the Amount of Temporary Space for archecker 6-4
Verifying Backups 6-5
Interpreting Verification Messages 6-6
Verification Failures 6-6
Fixing Backup Verification Problems 6-7

Chapter 7. Restoring Data with ON-Bar 7-1
In This Chapter 7-2
ON-Bar Restore Types 7-2
Warm Restore 7-2
Cold Restore 7-2
Mixed Restore 7-3
Parallel Restore 7-4
Whole-System Restore (IDS) 7-4
Point-in-Time Restore 7-4
Imported Restore 7-4
Rename Chunks Restore (IDS) 7-5
Restartable Restore (IDS) 7-5
Continuous Log Restore 7-5
Pre-Recovery Checklist 7-6
Monitoring Restores 7-7
Ensuring That Storage Devices Are Available 7-7
Restoring Save Sets with ISM 7-8
Performing a Complete Restore 7-8
Performing a Physical-Only or Logical-Only Restore 7-10
Using IBM Informix Server Administrator to Restore Data 7-13
Examples of ON-Bar Restore Commands 7-13
Configuring Continuous Log Restore using ON-Bar 7-25
Renaming Chunks During a Restore (IDS) 7-26
Key Considerations 7-26
New-Chunk Requirements 7-26
Syntax 7-27
Examples of Renaming Chunks During a Restore 7-28
Transferring Data with the Imported Restore 7-29
Importing a Restore 7-30
Initializing High-Availability Data Replication with ON-Bar (IDS) 7-32
Restoring Nonlogging Databases and Tables 7-33
Restoring Table Types 7-33
Restoring Tables with Large Objects 7-34
Using Restartable Restore to Recover Data (IDS) 7-35
Restartable Restore Example 7-35
Restarting a Restore 7-35
Resolving a Failed Restore 7-38
Understanding ON-Bar Restore Processes 7-40
Warm-Restore Sequence on Dynamic Server 7-40
Cold-Restore Sequence on Dynamic Server 7-41
Warm-Restore Sequence on Extended Parallel Server 7-42
Cold-Restore Sequence on Extended Parallel Server 7-43

Chapter 8. Performing an External Backup and Restore 8-1

In This Chapter . . . 8-1
External Backup and Restore Overview . . . 8-1
Blocking Before Backing Up . . . 8-2
Rules for an External Backup . . . 8-2
Preparing for an External Backup . . . 8-3
Performing an External Backup . . . 8-7
Data Restored in an External Restore . . . 8-10
Renaming Chunks . . . 8-11
Using External Restore Commands . . . 8-11
Rules for an External Restore . . . 8-12
Performing an External Restore . . . 8-13
Initializing HDR with an External Backup and Restore (IDS) . . . 8-16

Chapter 9. Customizing and Maintaining ON-Bar . . . 9-1
In This Chapter . . . 9-1
Customizing ON-Bar and Storage-Manager Commands . . . 9-2
Printing the Backup Boot Files . . . 9-2
Migrating Backed-Up Logical Logs to Tape . . . 9-3
Using start_worker.sh to Start onbar_worker Processes Manually (XPS) . . . 9-4
Expiring and Synchronizing the Backup Catalogs . . . 9-5
Choosing an Expiration Policy . . . 9-5
Using the onsmsync Utility . . . 9-6
Starting and Stopping ON-Bar Sessions (XPS) . . . 9-10
Using the onbar_w Utility . . . 9-10
Monitoring the Backup Scheduler Status (XPS) . . . 9-11
Using the ***onstat -g bus*** Option . . . 9-11
Using the ***onstat -g bus_sm*** Option . . . 9-12
Monitoring the Performance of ON-Bar and the Storage Managers . . . 9-13
Setting ON-Bar Performance Statistics Levels . . . 9-13
Viewing ON-Bar Backup and Restore Performance Statistics . . . 9-13

Chapter 10. ON-Bar Configuration Parameters . . . 10-1
In This Chapter . . . 10-2
archecker Configuration Parameters in AC_CONFIG . . . 10-2
AC_CONFIG File Environment Variable . . . 10-2
AC_MSGPATH . . . 10-3
AC_STORAGE . . . 10-3
AC_TIMEOUT . . . 10-4
AC_VERBOSE . . . 10-4
ON-Bar Parameters in ONCONFIG . . . 10-5
ALARMPROGRAM . . . 10-5
ALRM_ALL_EVENTS (IDS) . . . 10-5
BACKUP_FILTER . . . 10-6
BAR_ACT_LOG . . . 10-6
BAR_BOOT_DIR (XPS) . . . 10-7
BAR_BSALIB_PATH . . . 10-8
BAR_DBS_COSVR (XPS) . . . 10-9
BAR_DEBUG . . . 10-9
BAR_DEBUG_LOG . . . 10-10
BAR_HISTORY . . . 10-10
BAR_IDLE_TIMEOUT (XPS) . . . 10-10
BAR_IXBAR_PATH (IDS) . . . 10-11
BAR_LOG_COSVR (XPS) . . . 10-11
BAR_MAX_BACKUP (IDS) . . . 10-12
BAR_NB_XPORT_COUNT (IDS) . . . 10-12
BAR_PERFORMANCE . . . 10-13
BAR_PROGRESS_FREQ . . . 10-13
BAR_RETRY . . . 10-14
BAR_SIZE_FACTOR . . . 10-15
BAR_SM (XPS) . . . 10-16

BAR_SM_NAME (XPS) . . . 10-16
BAR_WORKER_COSVR (XPS) . . . 10-16
BAR_WORKER_MAX (XPS) . . . 10-16
BAR_XFER_BUF_SIZE (IDS) . . . 10-17
BAR_XFER_BUFSIZE (XPS) . . . 10-18
BAR_XPORT_COUNT (XPS) . . . 10-18
ISM_DATA_POOL . . . 10-19
ISM_LOG_POOL . . . 10-19
LOG_BACKUP_MODE (XPS) . . . 10-20
LTAPEDEV (IDS) . . . 10-20
RESTARTABLE_RESTORE (IDS) . . . 10-21
RESTORE_FILTER . . . 10-21
Files That ON-Bar, ISM, and TSM Use . . . 10-22

Chapter 11. ON-Bar Catalog Tables . . . 11-1
In This Chapter . . . 11-1
bar_action . . . 11-2
bar_instance . . . 11-2
bar_ixbar . . . 11-4
bar_object . . . 11-6
bar_server . . . 11-6
bar_syncdeltab . . . 11-6
ON-Bar Catalog Map . . . 11-7
Backup Scheduler SMI Tables (XPS) . . . 11-8
sysbuobject . . . 11-9
sysbuobjses . . . 11-9
sysbusession . . . 11-10
sysbusm . . . 11-10
sysbusmdbspace . . . 11-10
sysbusmlog . . . 11-10
sysbusmworker . . . 11-11
sysbuworker . . . 11-11

Chapter 12. ON-Bar Messages and Return Codes . . . 12-1
In This Chapter . . . 12-1
About ON-Bar Messages . . . 12-1
Message Format . . . 12-1
Message Numbers . . . 12-2
ON-Bar Usage Messages . . . 12-2
ON-Bar Return Codes . . . 12-6

Part 3. ontape Backup and Restore System (IDS)

Chapter 13. Configuring ontape . . . 13-1
In This Chapter . . . 13-1
ontape Configuration Parameters . . . 13-1
ontape Data Transformation Filter Parameters . . . 13-2
ontape Tape and Tape Device Parameters . . . 13-2
Setting the Tape-Device Parameters . . . 13-3
Specifying the Tape-Block-Size Parameters . . . 13-5
Specifying the Tape-Size Parameters . . . 13-5
Checking and Changing ontape Configuration Parameters . . . 13-5
Who Can Change ontape Parameters? . . . 13-6
When Can You Change ontape Parameters? . . . 13-6
Changing ontape Parameters . . . 13-6

Chapter 14. Backing Up with ontape . . . 14-1
In This Chapter . . . 14-1
Summary of ontape Tasks . . . 14-1

Starting ontape 14-2
Using ontape Exit Codes 14-2
Changing Database Logging Status 14-2
Creating a Backup 14-3
Backup Levels 14-3
Back Up After Changing the Physical Schema 14-3
Prepare for a Backup 14-4
Performing a Backup 14-7
When the Logical-Log Files Fill During a Backup 14-9
When a Backup Terminates Prematurely 14-10
Monitoring Backup History Using oncheck 14-10
Backing Up Logical-Log Files with ontape 14-10
Before You Back Up the Logical-Log Files 14-11
When to Back Up Logical-Log Files 14-12
Starting an Automatic Logical-Log Backup 14-12
Starting a Continuous Logical-Log File Backup 14-12
Ending a Continuous Logical-Log Backup 14-13
Devices that Logical-Log Backups Must Use 14-13

Chapter 15. Restoring with ontape 15-1
In This Chapter 15-1
Types of Physical Restore 15-2
Full-System Restores 15-2
Restores of Dbspaces, Blobspaces, and Sbspaces 15-2
Cold, Warm, or Mixed Restores 15-2
Cold Restores 15-2
Warm Restores. 15-3
Mixed Restores 15-3
Performing a Restore 15-4
Restoring the Whole System 15-5
Gathering the Appropriate Tapes 15-6
Deciding on a Complete Cold or a Mixed Restore 15-6
Verifying Your Database Server Configuration 15-6
Performing a Cold Restore. 15-7
Restoring Selected Storage Spaces 15-8
Restoring Raw Tables 15-9
Configuring Continuous Log Restore Using ontape 15-10
Renaming Chunks During a Restore 15-10
Validation Sequence for Renaming Chunks 15-11
New Chunk Requirements 15-11
Renaming Chunks with Command Line Options 15-12
Renaming Chunks with a File 15-12
Renaming Chunks While Specifying Other Options 15-12
Renaming a Chunk to a Nonexistent Device 15-12
Restoring from Standard Input 15-13
Restoring Data to a Remote Server 15-13
Simultaneous Backup and Restore Using Standard I/O 15-14

Chapter 16. Performing an External Backup and Restore 16-1
In This Chapter 16-1
Recovering Data Using an External Backup and Restore 16-1
Data That is Backed Up in an External Backup 16-1
Rules for an External Backup 16-2
Performing an External Backup 16-2
Preparing for an External Backup 16-3
Blocking and Unblocking Dynamic Server 16-3
Tracking an External Backup 16-3
Data That is Restored in an External Restore. 16-4
Using External Restore Commands 16-4
Rules for an External Restore 16-5

Renaming Chunks 16-5
Performing an External Restore 16-5
Initializing HDR with an External Backup and Restore 16-6

Part 4. Administering Backup and Restore

Chapter 17. Restoring Table-level Data 17-1
In This Chapter 17-2
Overview of archecker 17-2
Configuration File 17-3
Schema Command File 17-3
Table-Level Restore and Locales 17-3
Data Restore with archecker 17-3
Physical Restore 17-4
Logical Restore 17-4
Restoring Data with archecker 17-5
Syntax 17-5
Examples 17-7
The archecker Schema Reference 17-9
The CREATE TABLE Statement 17-9
The CREATE EXTERNAL TABLE Statement 17-10
The DATABASE Statement 17-11
The INSERT Statement 17-12
The RESTORE Statement 17-13
The SET Statement 17-13
SQL Comments 17-14
Schema Command File Examples 17-14
The archecker Configuration Parameter Reference 17-16
AC_DEBUG 17-17
AC_IXBAR 17-17
AC_LTAPEBLOCK 17-17
AC_LTAPEDEV (IDS) 17-18
AC_MSGPATH 17-18
AC_PAGESIZE (XPS) 17-18
AC_RESTORE_FILTER 17-18
AC_SCHEMA 17-18
AC_STORAGE 17-19
AC_TAPEBLOCK 17-19
AC_TAPEDEV (IDS) 17-19
AC_VERBOSE 17-19

Part 5. Appendixes

Notices F-1
Trademarks F-3

Index X-1

Introduction

In This Introduction xi
About This Manual xi
Types of Users xi
Software Dependencies xii
Assumptions About Your Locale xii
Demonstration Database xii
Documentation Conventions xiii
Typographical Conventions xiii
Feature, Product, and Platform Markup xiv
Example Code Conventions xiv
Additional Documentation xv
Compliance with Industry Standards xv
Syntax Diagrams xv
How to Read a Command-Line Syntax Diagram xvi
Keywords and Punctuation xvii
Identifiers and Names xvii
How to Provide Documentation Feedback xviii

In This Introduction

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

About This Manual

This manual describes how to use the IBM® Informix® ON–Bar and **ontape** utilities to back up and restore database server data. These utilities enable you to recover your databases after data is lost or becomes corrupt due to hardware or software failure or accident.

ON–Bar requires IBM Informix Storage Manager, Version 2.2, IBM Tivoli® Storage Manager, or a third-party storage manager to manage the storage devices. The backup and restore processes, the ON–Bar commands, and the ON–Bar configuration parameters are different for IBM Informix Dynamic Server (IDS) and IBM Informix Extended Parallel Server.

The **ontape** utility does not require a storage manager and works on Dynamic Server only.

Types of Users

This manual is written for the following users:

- Database administrators
- System administrators
- Backup operators
- Technical support personnel

This manual is written with the assumption that you have the following background:

- Some experience with storage managers, which are applications that manage the storage devices and media that contain backups

© Copyright IBM Corp. 1996, 2008

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

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

Software Dependencies

This manual is written with the assumption that you are using one of the following database servers:

- IBM Informix Dynamic Server (IDS), Version 11.50
- IBM Informix Extended Parallel Server, Version 8.51

ON–Bar works differently on Dynamic Server and on Extended Parallel Server.

Assumptions About Your Locale

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

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

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

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

Demonstration Database

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

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix manuals are based on the **stores_demo** database.

Extended Parallel Server

- The **sales_demo** database illustrates a dimensional schema for data-warehousing applications. For conceptual information about dimensional data modeling, see the *IBM Informix Database Design and Implementation Guide*.

Dynamic Server

- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

End of Dynamic Server

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

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

Documentation Conventions

This section describes the following conventions, which are used in the product documentation for IBM Informix Dynamic Server:

- Typographical conventions
- Feature, product, and platform conventions
- Syntax diagrams
- Command-line conventions
- Example code conventions

Typographical Conventions

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

Convention	Meaning
KEYWORD	Keywords of SQL, SPL, and some other programming languages appear in uppercase letters in a serif font.
italics	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface	Names of program entities (such as classes, events, and tables), environment variables, file names, path names, and interface elements (such as icons, menu items, and buttons) 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 a menu item. For example, "Choose **Tools > Options**" means choose the **Options** item from the **Tools** menu.

Technical changes to the text are indicated by special characters depending on the format of the documentation:

HTML documentation
New or changed information is surrounded by blue >> and << characters.

PDF documentation
A plus sign (+) is shown to the left of the current changes. A vertical bar (|) is shown to the left of changes made in earlier shipments.

Feature, Product, and Platform Markup

Feature, product, and platform markup identifies paragraphs that contain feature-specific, product-specific, or platform-specific information. Some examples of this markup follow:

Dynamic Server

Identifies information that is specific to IBM Informix Dynamic Server

End of Dynamic Server

Windows Only

Identifies information that is specific to the Windows operating system

End of Windows Only

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

Table Sorting (Windows)

Example Code Conventions

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

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

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

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

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

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

Additional Documentation

You can view, search, and print all of the product documentation from the IBM Informix Dynamic Server information center on the Web at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp.

For additional documentation about IBM Informix Dynamic Server and related products, including release notes, machine notes, and documentation notes, go to the online product library page at http://www.ibm.com/software/data/informix/pubs/library/. Alternatively, you can access or install the product documentation from the Quick Start CD that is shipped with the product.

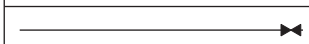
Compliance with Industry Standards

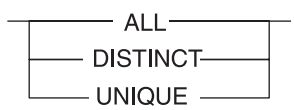
The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) have jointly established a set of industry standards for the Structured Query Language (SQL). IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992),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

Syntax Diagrams

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

Table 1. Syntax Diagram Components

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

	`---+-----ALL-------+---`
	`  +--DISTINCT-----+`
	`  '---UNIQUE------'`
		Required item with choice. One and only one item must be present.

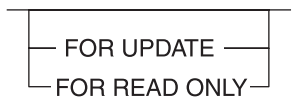
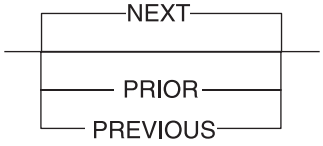
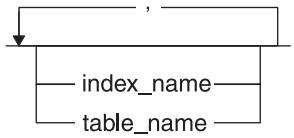
	`---+------------------+---`
	`  +--FOR UPDATE-----+`
	`  '--FOR READ ONLY--'`
		Optional items with choice are shown below the main line, one of which you might specify.

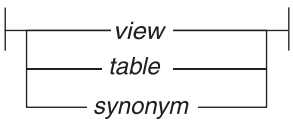
Table 1. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning

	<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 will be used as the default.

	<pre>.-------,-----------.
	V |
	---+-----------------+---
	+---index_name---+
	'---table_name---'</pre>
		Optional items. Several items are allowed; a comma must precede each repetition.

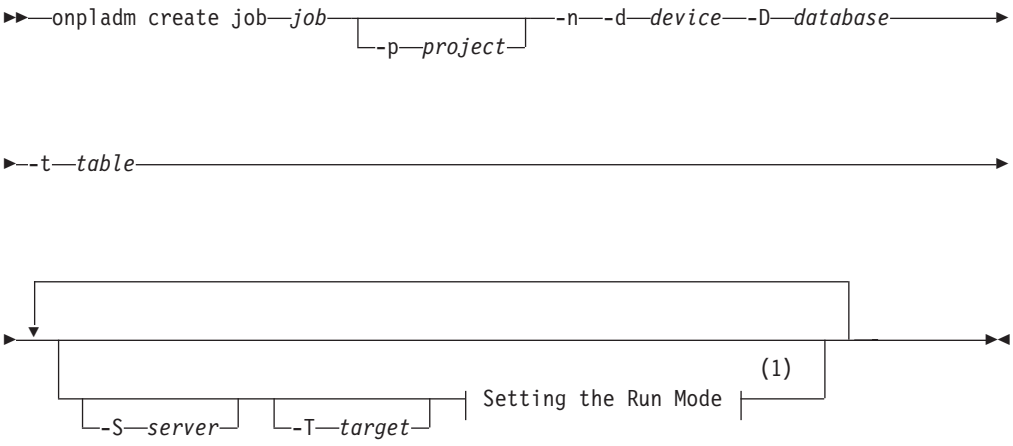
	<pre>>>-| Table Reference |-><</pre>	Reference to a syntax segment.

Table Reference
 
	<pre>Table Reference
	|--+-----*view*--------+--|
	+------*table*------+
	'----*synonym*------'</pre>
		Syntax segment.

How to Read a Command-Line Syntax Diagram

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

Creating a No-Conversion Job

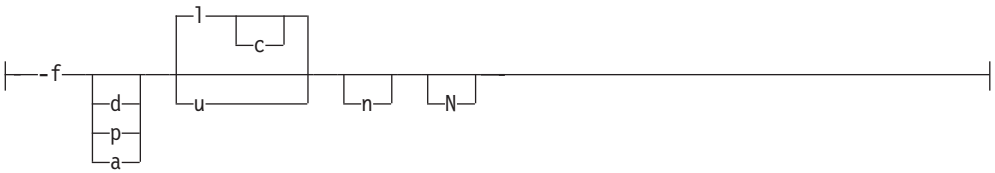


Notes:

1 See page Z-1

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

Setting the Run Mode:



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

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

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

Keywords and Punctuation

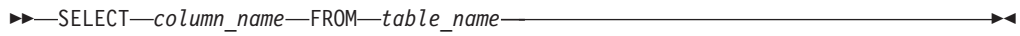
Keywords are words reserved for statements and all commands except system-level commands. When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

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

Identifiers and Names

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

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



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 by using one of the following methods:

- Send e-mail to docinf@us.ibm.com.
- Go to the Information Center at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp and open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.

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

We appreciate your suggestions.

Part 1. Overview of Backup and Restore

© Copyright IBM Corp. 1996, 2008

Chapter 1. Backup and Restore Concepts

In This Chapter 1-1
Recovery System 1-1
Backup Systems 1-1
Logical-Log Backup 1-2
When You Do Not Use Logging 1-3
Manual and Continuous Logical-Log Backups 1-3
Log Salvage 1-3
Why You Must Save Logical-Log Backups 1-3
Restore Systems. 1-4
Warm, Cold, and Mixed Restores 1-4
Physical and Logical Restores 1-5
Comparing ON-Bar and ontape 1-6

In This Chapter

Dynamic Server provides two utilities for backing up and restoring database server data. Both utilities back up and restore storage spaces and logical logs. However, they do support different features and it is important to know the differences. This chapter explains basic backup and restore concepts for Informix database servers and compares the ON–Bar and **ontape** utilities.

ON-Bar backs up and restores storage spaces (dbspaces) and logical file, using a storage manager, whereas **ontape** does not use a storage manager.

Utility	Storage Manager	Where Discussed
ON–Bar	IBM Informix Storage Manager (ISM) IBM Tivoli Storage Manager Third-party storage manager	Chapter 3 through Chapter 12
ontape	None	Chapter 13 through Chapter 15

Dynamic Server does not show errors in standard output (**stdout**) if an error occurs when you use **onbar -b** to backup storage spaces or **onbar-r** to restore storage spaces. Therefore, when you use **onbar -b** or **onbar -r**, you must check information in the **ON-Bar** activity log (**bar_act_log**). As ON–Bar backs up and restores data, it writes progress messages, warnings, and error messages to the **bar_act.log**.

Recovery System

A *recovery system* enables you to back up your database server data and subsequently restore it if your current data becomes corrupt or inaccessible. The causes of data corruption or loss can range from a program error to a disk failure to a disaster that damages the entire facility. A recovery system enables you to recover data that you already lost due to such mishaps.

Backup Systems

A *backup* is a copy of one or more *dbspaces* (also called storage spaces) and logical logs that the database server maintains. On Dynamic Server, you can also back up *blobspaces* and *sbspaces*. On Extended Parallel Server, you can also back up *dbslices*. For a description of storage spaces, see your *IBM Informix Administrator's Guide*.

© Copyright IBM Corp. 1996, 2008

The backup copy is usually written to a *secondary storage* medium such as disk, magnetic tape, or optical disk. We recommend that you store the media offline and keep a copy off site if possible.

Important: Database backups do not replace ordinary operating-system backups, which back up files other than Informix database files.

Figure 1-1 illustrates the basic concept of a database backup.

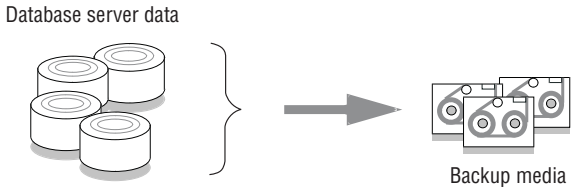


Figure 1-1. A Backup of Database Server Data

You do not always have to back up all the storage spaces. If some tables change daily but others rarely change, it is inefficient to back up the storage spaces that contain the unchanged tables every time that you back up the database server. You need to plan your backup schedule carefully to avoid long delays for backing up or restoring data.

To provide a more flexible backup environment, ON–Bar and **ontape** support the following three *backup levels*:

- Level 0 backs up all used pages that contain data for the specified storage spaces.

  You need all these pages to restore the database to the state that it was in at the time that you made the backup.
- Level 1 backs up only data that has changed since the last level-0 backup of the specified storage spaces.

  All changed table and index pages are backed up, including those with deleted data. The data that is copied to the backup reflects the state of the changed data at the time that the level-1 backup began.
- Level 2 backs up only data that has changed since the last level-1 backup of the specified storage spaces.

  A level-2 backup contains a copy of every table and index page in a storage space that has changed since the last level-1 backup.

Important: If disks and other media are completely destroyed and need to be replaced, you need at least a level-0 backup of all storage spaces and relevant logical logs to restore data completely on the replacement hardware.

For details, see Chapter 5, “Backing Up with ON-Bar,” on page 5-1, and Chapter 14, “Backing Up with ontape,” on page 14-1.

Logical-Log Backup

A *logical-log backup* is a copy to disk or tape of all full logical-log files. The logical-log files store a record of database server activity that occurs *between backups*.

To free full logical-log files, back them up. The database server reuses the freed logical-log files for recording new transactions. For a complete description of the logical log, see your *IBM Informix Administrator's Guide*.

When You Do Not Use Logging

Even if you do not specify logging for databases or tables, you need to back up the logical logs because they contain administrative information such as checkpoint records and additions and deletions of chunks. When you back up these logical-log files, you can do warm restores even when you do not use logging for any of your databases.

Manual and Continuous Logical-Log Backups

A *manual logical-log backup* backs up all the full logical-log files and stops at the current logical-log file.

If you turn on *continuous logical-log backup*, the database server backs up each logical log automatically when it becomes full. If you turn off continuous logical-log backup, the logical-log files continue to fill. If all logical logs are filled, the database server hangs until the logs are backed up.

Log Salvage

When the database server is offline (Dynamic Server) or in microkernel mode (Extended Parallel Server), you can perform a special kind of logical-log backup, called a *log salvage*. In a log salvage, the database server accesses the log files directly from disk. The log salvage backs up any logical logs that have not yet been backed up and are not corrupted or destroyed. The log salvage enables you to recover all of your data up to the last available and uncorrupted logical-log file and the last complete transaction.

Why You Must Save Logical-Log Backups

Perform frequent logical-log backups for the following reasons:

- To free full logical-log files
- To minimize data loss if a disk that contains logical logs fails
- To ensure that restores contain consistent and the latest transactions

Save the logical-log backups from the last two level-0 backups so that you can use them to complete a restore. If a level-0 backup is inaccessible or unusable, you can restore data from an older backup, if you have one. If any of the logical-log backups are also inaccessible or unusable, however, you cannot roll forward the transactions from those logical-log files or from any subsequent logical-log files.

Warning: You will lose transactions in logical-log files that are not backed up or salvaged.

To illustrate, as Figure 1-2 shows, suppose you perform a level-0 backup on Monday at 10:00 P.M. and then back up the logical logs on Tuesday at midnight. On Wednesday at 11:00 A.M., you suffer a mishap that destroys your databases. You would be unable to restore the transactions that occurred between midnight on Tuesday and 11:00 A.M. on Wednesday unless you had continuous logical-log backup set up.

If the disks that contain the storage spaces with the logical logs are damaged, the transactions after midnight on Tuesday might be lost. To restore these transactions from the last logical-log backup, try to salvage the logical logs before you repair or replace the bad disk and then perform a cold restore.

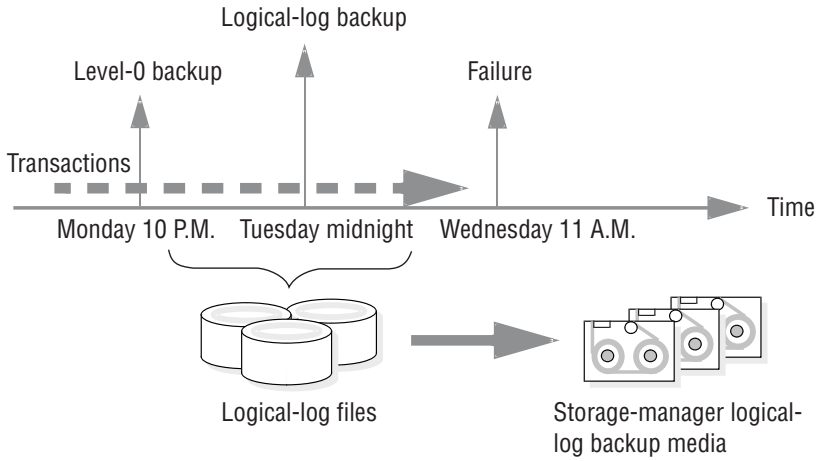


Figure 1-2. Storage Space and Logical-Log Backups

For more information, see "Backing Up Logical Logs" on page 5-18 and "Backing Up Logical-Log Files with ontape" on page 14-10.

Restore Systems

A *restore* recreates database server data from backed-up storage spaces and logical-log files. A restore recreates database server data that has become inaccessible because of any of the following conditions:

- You need to replace a failed disk that contains database server data.
- A logic error in a program has corrupted a database.
- You need to move your database server data to a new computer.
- A user accidentally corrupted or destroyed data.

To restore data up to the time of the failure, you must have at least one level-0 backup of each of your storage spaces from before the failure and the logical-log files that contain all transactions since these backups.

Warm, Cold, and Mixed Restores

When you restore data, you must decide whether to do so while the database server is in quiescent, online, offline (Dynamic Server), or microkernel mode (Extended Parallel Server). The types of restores are as follows:

- If you restore noncritical data while the database server is online or quiescent, that process is called a *warm restore.*
- When Dynamic Server is offline or Extended Parallel Server is in microkernel mode, you can perform a *cold restore*.
- A *mixed restore* is a cold restore of some storage spaces followed by a warm restore of the remaining storage spaces.

Warm Restore: As Figure 1-3 shows, a warm restore restores noncritical storage spaces. A warm restore consists of one or more physical restores, a logical-log backup, and a logical restore.

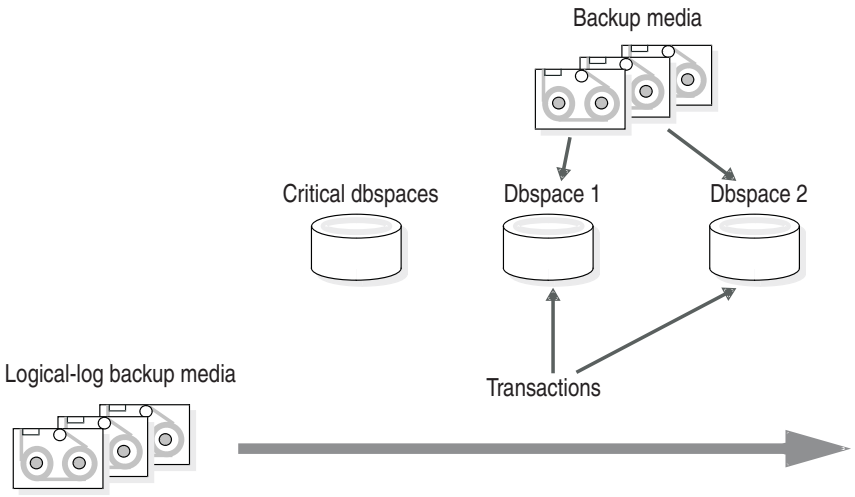


Figure 1-3. Warm Restore

You cannot perform more than one simultaneous warm restore.

Cold Restore: As Figure 1-4 shows, a cold restore salvages the logical logs, and restores the critical dbspaces (root dbspace and the dbspaces that contain the physical log and logical-log files), other storage spaces, and the logical logs.

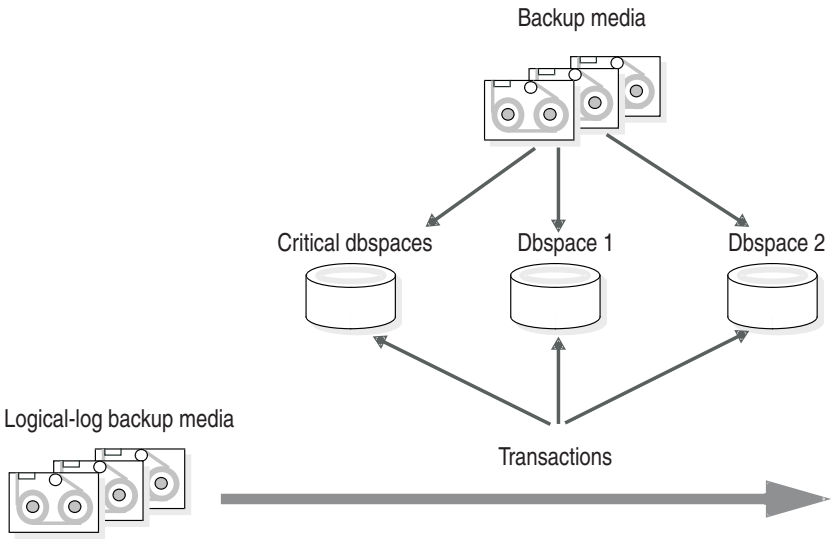


Figure 1-4. Cold Restore

You can perform a cold restore onto a computer that is not identical to the one on which the backup was performed by giving any chunk a new path name and offset during the restore.

Physical and Logical Restores

ON–Bar and **ontape** restore database server data in two phases:

- The first phase is the *physical restore*, which restores data from backups of all or selected storage spaces.
- The second phase is the *logical restore*, which restores transactions from the logical-log backups. The database server automatically knows which logical logs to restore.

Physical Restore: During a physical restore, ON–Bar or **ontape** restores the data from the most recent level-0, level-1, and level-2 backups. When you suffer a disk failure, you can restore to a new disk only those storage spaces with chunks that resided on the failed disk. Figure 1-5 illustrates a physical restore.

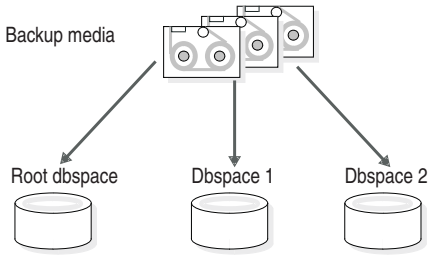


Figure 1-5. Physical Restore

Logical Restore: As Figure 1-6 shows, the database server *replays* the logical logs to reapply any database transactions that occurred after the last backup. The logical restore applies only to the physically-restored storage spaces.

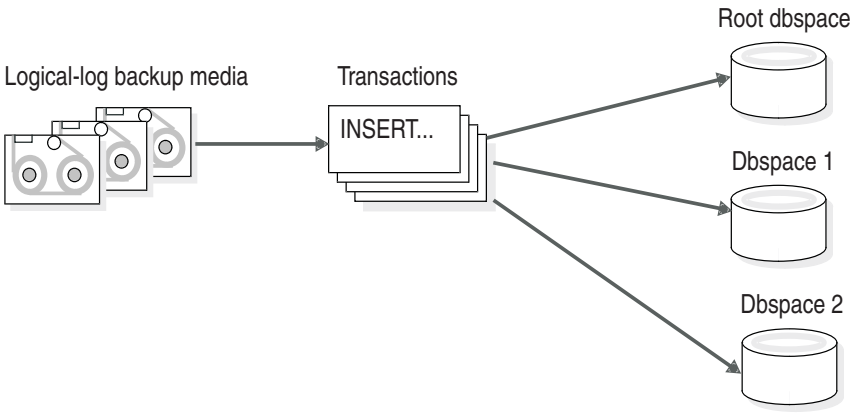


Figure 1-6. Logical Restore

For more information, see Chapter 7, “Restoring Data with ON-Bar,” on page 7-1 and Chapter 15, “Restoring with ontape,” on page 15-1.

Comparing ON-Bar and ontape

Informix provides two backup and restore utilities:

ON-Bar

Backs up and restores storage spaces (dbspaces) and logical files, using a storage manager to track backups and storage media. Use this utility when you need to:

- Select specific storage spaces
- Back up to a specific point in time
- Perform separate physical and logical restores
- Back up and restore different storage spaces in parallel
- Use multiple tape drives concurrently for backups and restores
- Perform imported restores
- Perform external backups and restores

ontape

Logs, backs up, and restores data, and enables you to change the logging status of a database. It does not use a storage manager. Use this utility when you need to:

- Backup and restore data without a storage manager
- Back up without selecting storage spaces
- Change the logging mode for databases

Important: The backup tapes that ontape and ON-Bar produce are not compatible. You cannot create a backup with ontape and restore it with ON-Bar, or vice versa.

Table 1-1 compares ON–Bar and **ontape**. If you are switching to ON–Bar and ISM from **ontape**, note that ON–Bar works differently.

Table 1-1. Differences Between ON-Bar and ontape

Can the utility...	ON-Bar	ontape
Use a storage manager to track backups and storage media?	yes	no
Back up all database server data?	yes	yes
Back up selected storage spaces?	yes	no
Back up logical-log files?	yes	yes
Perform continuous logical-log backups?	yes	yes
Perform continuous logical-log restore?	yes	yes
Back up while the database server is online?	yes	yes
Back up while the database server is in quiescent mode?	yes	yes
Restore all database server data?	yes	yes
Restore selected storage spaces?	yes	yes
Back up and restore storage spaces serially?	yes	yes
Perform cold restores with the database server offline or in microkernel mode?	yes	yes
Initialize high availability data replication?	yes	yes
Restore data to a specific point in time?	yes	no
Perform separate physical and logical restores?	yes	yes
Back up and restore different storage spaces in parallel?	yes	no
Use multiple tape drives concurrently for backups and restores?	yes	no
Restart a restore?	yes	no
Rename a chunk path name or device during a cold restore?	yes	yes
Perform imported restores?	yes	yes
Perform external backups and restores?	yes	yes
Monitor performance?	yes	no
Change logging mode for databases?	no	yes
Transform data with external programs?	yes	yes

Additional differences:

Emergency boot files and sysutils database

The **ontape** utility does not use the **sysutils** database or the emergency boot files.

Simultaneous sessions

ON-Bar, with ISM, supports up to four simultaneous sessions per ISM instance. The **ontape** utility supports two simultaneous sessions, one for physical backup or restore, and one for log backup.

Device support and storage management

- The **ontape** utility supports remote backup devices on other hosts. ON-Bar, with ISM, does not.
- ON-Bar, with ISM, supports different sets of tape drives on various hardware platforms.
- You can use ON-Bar with third party storage managers to obtain more sophisticated device support and storage management.

Changing the logging mode of a database

You cannot change the logging mode for ON-Bar; however you can use the **ondblog** utility to do this task when using ON-Bar.

You can also use the SQL administration API alternative, ALTER LOGMODE. See *IBM Informix Guide to SQL: Syntax* for more information.

Chapter 2. Planning for Backup and Restore

In This Chapter 2-1
Planning a Recovery Strategy 2-1
Types of Data Loss. 2-1
Determining Failure Severity 2-1
Data Use Determines Your Backup Schedule 2-2
Scheduling Backups 2-2
Security Requirements for Label-based Access Control 2-3
Planning a Backup System for a Production Database Server. 2-3
Evaluating Hardware and Memory Resources 2-3
Evaluating Backup and Restore Time 2-3
Evaluating Logging and Transaction Activity 2-4
Transforming Data with External Programs 2-4

In This Chapter

This chapter describes the following planning concepts for backup and restore:

- Planning a recovery strategy
- Planning a backup system for a production database server

Planning a Recovery Strategy

Before you use ON–Bar or **ontape**, plan your recovery goals.

Types of Data Loss

The first step is to determine how much data loss, if any, is acceptable. The following types of data loss can occur:

- Deletion of the following:
  - Rows, columns, tables, or databases
  - Chunks, storage spaces, or logical logs
- Data corruption or incorrect data created
- Hardware failure (such as a disk that contains chunk files fails or a backup tape that wears out)
- Database server failure
- Natural disaster

Determining Failure Severity

After you determine your recovery goals, create your recovery plan. Develop a recovery plan for multiple levels of failure, as Table 2-1 shows.

© Copyright IBM Corp. 1996, 2008

Table 2-1. Sample Recovery Plans

Failure Severity	Data Loss	Suggested Recovery Plan
Small	Noncritical data is lost.	Restore of the data can wait until a nonpeak time. Use a warm restore.
Medium	The data that is lost is critical for your business but does not reside in a critical dbspace.	Perform a warm restore of this data as soon as possible.
Large	Critical dbspaces are lost.	Use a mixed restore to restore the critical data right away and a warm restore to restore noncritical data during off-peak hours.
Disaster	All data is lost.	Perform a cold or mixed restore as soon as possible.

Data Use Determines Your Backup Schedule

After you develop your recovery plan, create a backup plan. How you use the data also determines how you plan your backup schedule, as follows:

- Data usage

  How do users use the data?

  - Critical dbspaces (root dbspace and dbspaces that contain the physical log and at least one logical-log file)
  - Critical business application data
  - Long-term data storage for legal or record-keeping reasons
  - Data sharing among groups
  - Test data
- Transaction Time

  How much transaction time can be lost? Also, how long might it take to re-enter lost transactions manually? For example, can you afford to re-enter all transactions that occurred over the past three hours?
- Quantity and Distribution

  How much data can you afford to lose? For example, you lost one fourth of your customer profiles, or you lost the Midwest regional sales figures but the West Coast figures are intact.

Ask the following questions to assist in deciding how often and when you want to back up the data:

- Does your business have down time where the system can be restored?
- If your system is 24x7 (no down time), is there a nonpeak time where a restore could occur?
- If a restore must occur during a peak period, how critical is the time?
- Which data can you restore with the database server online (warm restore)? Which data must be restored offline (cold restore)?
- How many storage devices are available to back up and restore the data?

Scheduling Backups

Table 2-2 on page 2-3 shows a sample backup plan for a small or medium-sized system. Tailor your backup plan to the requirements of your system. The more often the data changes and the more important it is, the more frequently you need to back it up. For more information, see "Choosing a Backup Level" on page 5-6.

Table 2-2. Sample Backup Plan

Backup Level	Backup Schedule
Complete (level-0) backup	Saturday at 6 P.M.
Incremental (level-1) backup	Tuesday and Thursday at 6 P.M.
Incremental (level-2) backup	Daily at 6 P.M.
Level-0 backup of storage spaces that are updated frequently	Hourly

Important: Perform a level-0 backup after you change the physical schema, such as adding a chunk to a storage space. (See "Collecting Information About Your System Before a Backup" on page 5-7.)

Security Requirements for Label-based Access Control

For label-based access control (LBAC), the person who runs ON-Bar or ontape does not require an exemption to security policies or additional privilege to back up or restore data. LBAC protection remains intact after you restore data using ON-Bar or ontape.

Planning a Backup System for a Production Database Server

To plan for adequate backup protection for your data, analyze your database server configuration and activity and the types of backup media available at your installation. Also, consider your budget for storage media, disks, computers and controllers, and the size of your network.

Evaluating Hardware and Memory Resources

Evaluate the following database server and hardware configuration elements to determine which storage manager and storage devices to use:

- The number of I/O virtual processors
- The amount of memory available and the distribution of processor activity

Evaluating Backup and Restore Time

How long your backup or restore takes depends on your database server configuration and the database size:

- The speed of disks or tape devices

  The faster the storage devices, the faster the backup or restore time.
- The number of incremental backups that you want to restore if a disk or system failure requires you to rebuild the database

  Incremental backups use less storage space than full backups and also reduce restore time.
- The size and number of storage spaces in the database

  Backups: Many small storage spaces take slightly longer to back up than a few large storage spaces of the same total size.

  Restores: A restore usually takes as long to recover the largest storage space and the logical logs.
- Whether storage spaces are mirrored

  If storage spaces are mirrored, you reduce the chance of having to restore damaged or corrupted data. You can restore the mirror at nonpeak time with the database server online.

- The length of time users are interrupted during backups and restores

  If you perform backups and warm restores while the database server is online, users can continue their work but might notice a slower response. If you perform backups and warm restores with the database server in quiescent mode, users must exit the database server. If you perform a cold restore with the database server offline, the database server is unavailable to users, so the faster the restore, the better. An external backup and restore eliminates system downtime.
- The backup schedule

  Not all storage spaces need to be included in each backup or restore session. Schedule backups so that you can back up more often the storage spaces that change rapidly than those that seldom or never change. Be sure to back up each storage space at level-0 at least once.
- The layout of the tables across the dbspaces and the layout of dbspaces across the disks

  When you design your database server schema, organize the data so that you can restore important information quickly. For example, you should isolate critical and frequently used data in a small set of storage spaces on the fastest disks. You also can fragment large tables across dbspaces to balance I/O and maximize throughput across multiple disks. For more information, see your *IBM Informix Performance Guide*.
- The database server and system workload

  The greater the workload on the database server or system, the longer the backup or restore time.
- The values of backup and restore configuration parameters

  For example, the number and size of data buffers that ON–Bar uses to exchange data with the database server can affect performance. Use the BAR_NB_XPORT_COUNT and BAR_XFER_BUF_SIZE (IDS) or BAR_XPORT_COUNT and BAR_XFER_BUFSIZE (XPS) configuration parameters to control the number and size of data buffers.

Evaluating Logging and Transaction Activity

The following database server usage requirements also affect your decisions about the storage manager and storage devices:

- The amount and rate of transaction activity that you expect
- The number and size of logical logs

  If you need to restore data from a database server with very little transaction activity, define many small logical logs. You are less likely to lose data because of infrequent logical-log backups.
- How fast the logical-log files fill

  Back up log files before they fill so that the database server does not hang
- Database and table logging modes

  When you use many nonlogging databases or tables, logical-log backups might become less frequent.

Transforming Data with External Programs

You can use external programs as filter plug-ins to transform data to a different format prior to a backup and transform it back following the restore.

To compress or transform data, use the BACKUP_FILTER and RESTORE_FILTER configuration parameters to call external programs.

The filter can be owned by anyone, but should not have write access to non-privileged users. Permission on the filters will be same as that of permission on any other executable that is called by an IDS server or IDS utilities.

Filters can use the following environment variables:

INFORMIXDIR
The installation directory of IDS.

INFORMIXSERVER
The name of the IDS server.

ONCONFIG
The configuration file (also called the onconfig file).

See "Transforming with Filters during Backup and Restore" on page 3-12 for more information.

Part 2. ON-Bar Backup and Restore System

© Copyright IBM Corp. 1996, 2008

Chapter 3. Overview of the ON-Bar Backup and Restore System

In This Chapter . . . 3-1
Where to Find Information on Tasks for ON-Bar, ISM, and TSM . . . 3-2
ON-Bar Components for Dynamic Server . . . 3-4
ON-Bar Components for Extended Parallel Server . . . 3-5
Database Server and Storage-Manager Communication . . . 3-5
Backup Scheduler . . . 3-6
ON-Bar Utilities . . . 3-7
IBM Informix Storage Manager . . . 3-7
IBM Tivoli Storage Manager . . . 3-8
Third-Party Storage Managers . . . 3-8
XBSA Interface . . . 3-9
ON-Bar Tables . . . 3-9
ON-Bar Boot Files . . . 3-10
ON-BAR Boot File on Dynamic Server . . . 3-10
ON-BAR Boot File Extended Parallel Server . . . 3-10
ON-Bar Activity Log . . . 3-11
Specifying the Location of the Activity Log . . . 3-11
Specifying the Level of ON-Bar Debugging (IDS) . . . 3-11
Specifying the Location of the Debug Log . . . 3-11
Monitoring the Progress of a Backup or Restore . . . 3-11
Monitoring Backup and Restore Performance . . . 3-12
Transform Data with External Filter Programs . . . 3-12
Transforming with Filters during Backup and Restore . . . 3-12

In This Chapter

This chapter introduces the components of ON–Bar and describes how it works. The following topics are covered:

- Where to find information on ON–Bar, ISM, and TSM
- ON–Bar for Dynamic Server
- ON–Bar for Extended Parallel Server
- ON–Bar utilities

Table 3-1 on page 3-2 shows which database server versions support ON–Bar, IBM Informix Storage Manager (ISM), Version 2.2, and Tivoli Storage Manager (TSM).

© Copyright IBM Corp. 1996, 2008

Table 3-1. Informix support for On-Bar and ISM and TSM

Database Server	Version	ON-Bar Support	ISM Support	TSM Support
Dynamic Server	Version 7.24	X		
Dynamic Server	Version 7.3x	X	X	
IBM Informix Universal Server	Version 9.1x	X	X	
Dynamic Server	Version 9.2x	X	X	
Dynamic Server	Version 9.30	X	X	
Dynamic Server	Version 9.40	X	X	
Dynamic Server	Version 10.00	X	X	5.1.6 or later
Dynamic Server	Version 11.50	X	X	5.3.2 or later
IBM Informix OnLine XPS	Version 8.11	X	X	
Informix Dynamic Server with AD and XP Options	Version 8.2x	X	X	
Extended Parallel Server	Version 8.3x	X	X	
Extended Parallel Server	Version 8.40	X	X	
Extended Parallel Server	Version 8.51	X	X	5.1.6 or later

Where to Find Information on Tasks for ON-Bar, ISM, and TSM

The task-documentation matrix in Table 3-2 provides a quick reference to locating ON–Bar commands and ISM and TSM information.

Table 3-2. ON-Bar, ISM, and TSM Task-Documentation Matrix

If You Want To:	Chapter or Manual
Run ON-Bar backup and restore commands from SQL	*Administrator's Guide* and *Guide to SQL: Syntax*
Learn backup and restore concepts	Chapter 1
Configure and use ON-Bar, ISM, TSM, or another storage manager	Chapter 4 of this manual *IBM Informix Storage Manager Administrator's Guide* *Tivoli Storage Manager Administrator's Guide* Third-party storage-manager manual
Use the **onbar** script to customize ON-Bar, ISM, and TSM operations	Chapter 4 (setup) Chapter 9 (customization)
Use ON-Bar, ISM, and TSM configuration parameters See a list of the files that ON-Bar, ISM, and TSM use	Chapter 10 of this manual *IBM Informix Storage Manager Administrator's Guide*
• Set up ISM, TSM, or other storage manager to use certain storage devices for backup and restore operations • Manage backup media and storage devices for ON-Bar • Track the location of all backup data • Move backup data through a managed life cycle	*IBM Informix Storage Manager Administrator's Guide* *Tivoli Storage Manager Administrator's Guide* Third-party storage-manager manual

Table 3-2. ON-Bar, ISM, and TSM Task-Documentation Matrix (continued)

If You Want To:	Chapter or Manual
Back up storage spaces and logical logs: • **onbar -b -L [0|1|2]** (standard backup) • **onbar -b -O** (override error checking) • **onbar -b -w** (whole-system backup, IDS) • **onbar -b -F** (fake backup, IDS) • **onbar -b -p** (physical backup, XPS)	"Backing Up Storage Spaces and Logical Logs" on page 5-8
Back up logical logs only: • **onbar -b -l** • **onbar -b -l -s** (log salvage) • **onbar -b -l -c** (backup includes current log, IDS) • **onbar -b -l -C** (continuous log backup, IDS)	"Backing Up Logical Logs" on page 5-18
View backed-up logical logs using **onbar -P**	"Viewing Backed-Up Logical Logs" on page 5-20
Verify backups before you use the data in a restore: • **onbar -v** (verify backup) • **onbar -b -v** (backup and verify, XPS)	Chapter 6
Perform warm or cold restores: • **onbar -r** (parallel restore) • **onbar -r -p** (physical restore) • **onbar -r -l** (logical restore) • **onbar -r -O** (override error checking) • **onbar -r -t** (point-in-time restore) • **onbar -r -n** (point-in-log restore, IDS) • **onbar -r -w** (whole-system restore, IDS) • **onbar -RESTART** (restartable restore, IDS) • **onbar -r rename** (rename chunks restore, IDS)	Chapter 7, "Restoring Data with ON-Bar," on page 7-1
Perform external backups and restores: • **onmode -c block|unblock** (external backup, IDS) • **onutil ebr block|unblock** (external backup, XPS) • **onbar -r -e** (external restore)	Chapter 16
Use the **onsmsync** utility to expire old backup objects	Chapter 9
Use the **onbar_w** utility or **start_worker** script to start **onbar-worker** processes manually (XPS)	Chapter 9
Monitor Backup Scheduler status (XPS)	Chapter 9
Refer to the tables in the **sysutils** database and the Backup Scheduler tables in the **sysmaster** database	Chapter 11
Find corrective actions to ON–Bar error messages	The **finderr** utility on UNIX or IBM Informix Error Messages on Windows®
Find ON–Bar return codes	Chapter 12
Use GLS with ON-Bar	Appendix D
• Create and delete storage spaces and chunks • Manage database-logging status, logical-log files, and the physical log • Perform fast recovery	*IBM Informix Administrator's Guide* for your database server

Table 3-2. ON-Bar, ISM, and TSM Task-Documentation Matrix (continued)

If You Want To:	Chapter or Manual
• Locate complete information on all database server configuration parameters • Use the **ondblog** utility to change the logging mode • Use the **onlog** utility to display logical-log records	*IBM Informix Administrator's Reference*
Restoring table-level data with **archecker**	Chapter 17, "Restoring Table-level Data," on page 17-1

ON-Bar Components for Dynamic Server

Figure 3-1 on page 3-5 shows the following components of ON–Bar for Dynamic Server:

- Storage spaces (dbspaces, blobspaces, and sbspaces) and logical logs to be backed up or restored
- The ON–Bar catalog tables in the **sysutils** database
- The **onbar** script (**onbar.sh** on UNIX or **onbar.bat** on Windows)
- The **onbar-driver** (**onbar_d**)
- The XBSA shared library for the storage manager on your system

  Use either ISM, TSM, or a storage manager that a third-party vendor provides.
- Backup data on storage media
- The ON–Bar activity log
- The ON–Bar emergency boot file

ON–Bar communicates with both the database server and the storage manager. Use the **onbar** command to start a backup or restore. For a backup session, ON–Bar requests the contents of storage spaces and logical logs from the database server and passes them to the storage manager. The storage manager stores the data on storage media. For a restore session, ON–Bar requests the backed up data from the storage manager and restores it on the database server.

If you specify a parallel backup or restore, the **onbar-driver** (**onbar_d**) creates child **onbar_d** processes that perform backup and restore operations. Each child processes one storage space, then returns. ON–Bar processes log files serially. If you specify a serial backup or restore, the **onbar-driver** performs the operation one object at a time.

The **onbar_d** processes write status and error messages to the ON–Bar activity log and write information to the emergency boot file that is used in a cold restore. For more details, see "Backup Sequence on Dynamic Server" on page 5-24.

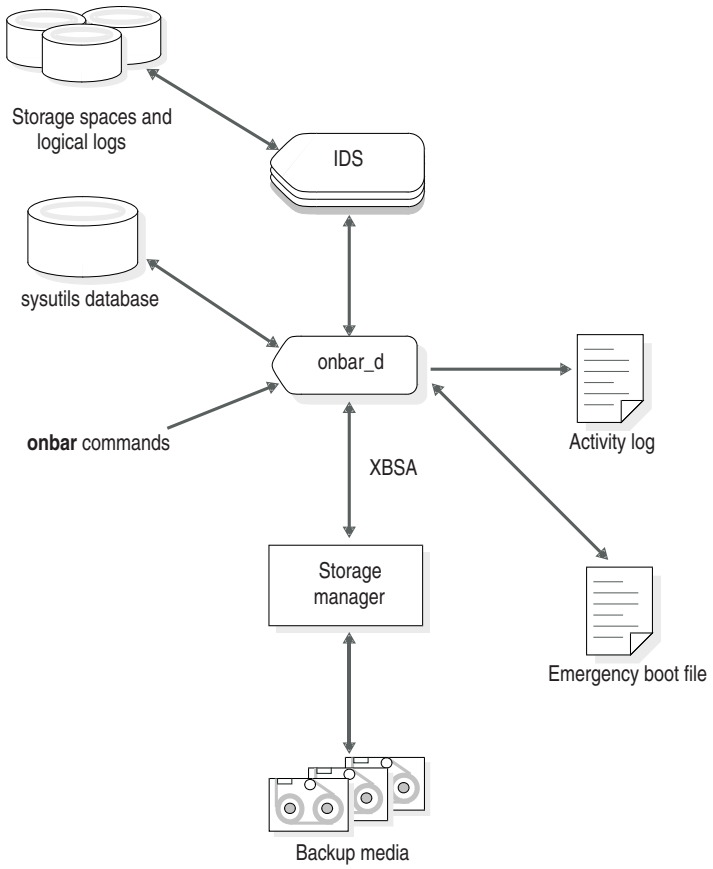


Figure 3-1. ON-Bar Components for Dynamic Server

ON-Bar Components for Extended Parallel Server

Figure 3-2 on page 3-6 shows the following components of ON–Bar for Extended Parallel Server:

- Storage spaces (dbspaces and dbslices) and logical logs to be backed up or restored
- The ON–Bar catalog tables in the **sysutils** database
- The **onbar** script and **onbar-driver** (**onbar_d**), onbar-worker (onbar_w), and onbar-merger (onbar_m)
- The XBSA shared library for each storage manager on your system

  Use either ISM, TSM, or a storage manager that a third-party vendor provides.
- Backup data on storage media
- The ON–Bar activity log
- The ON–Bar emergency boot files

Database Server and Storage-Manager Communication

For a backup session, ON–Bar requests the contents of storage spaces and logical logs from the database server and passes them to the storage manager. The storage

manager stores the data on storage media. For a restore session, ON–Bar requests the backed-up data from the storage manager and restores it on the database server.

The **onbar_d**, **onbar_w**, and **onbar_m** processes write status and error messages to the ON–Bar activity log. The **onbar_w** and **onbar_m** processes write information to the emergency boot files that are used in a cold restore.

Backup Scheduler

The **onbar-driver** (**onbar_d**) communicates backup or restore requests to the Backup Scheduler on Extended Parallel Server. The *Backup Scheduler* tracks all active and scheduled backup and restore activities for all the coservers. The Backup Scheduler creates one or more sessions, each of which contains a list of objects to back up or restore. It starts **onbar-worker** processes to back up or restore the objects and coordinates the session activity. A *session* is a single backup or restore request.

For details, see "Backup Sequence on Extended Parallel Server" on page 5-26.

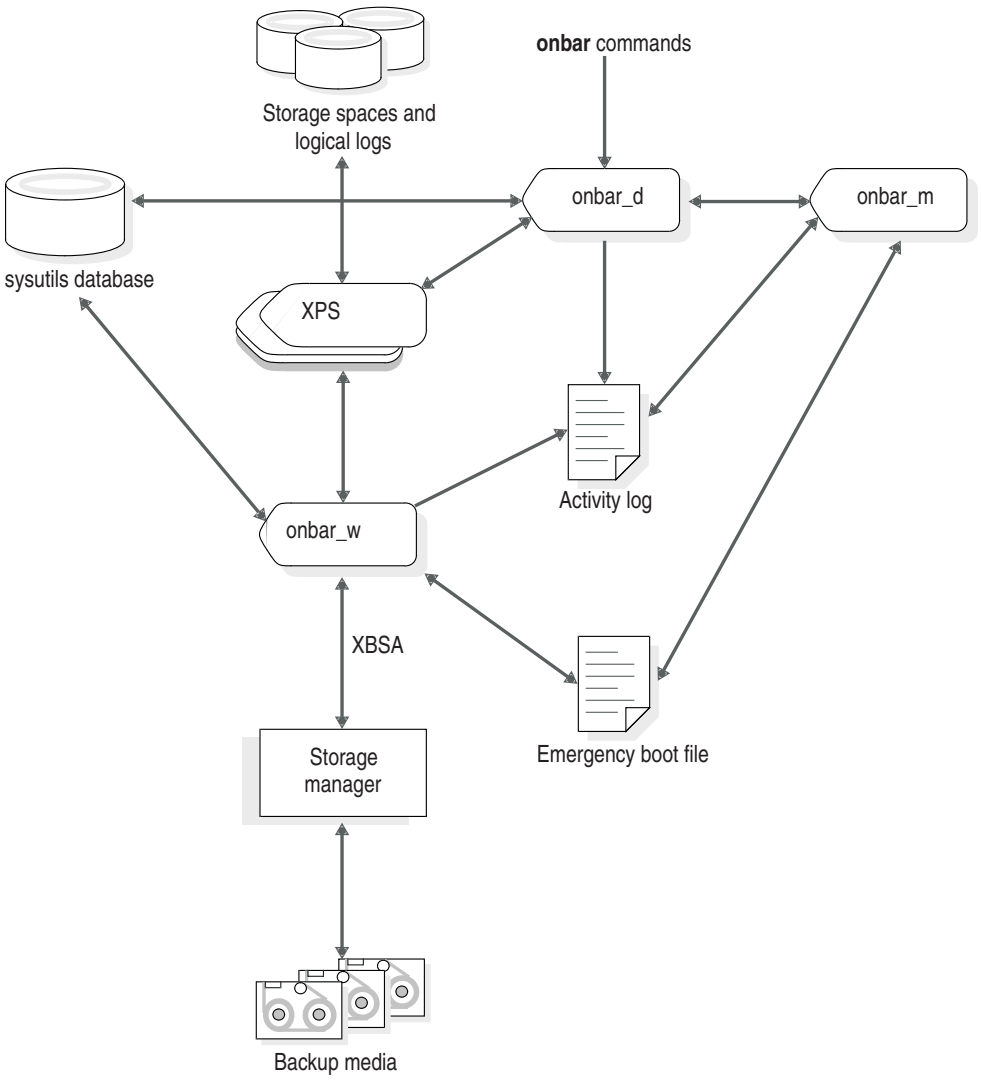


Figure 3-2. ON-Bar Components for Extended Parallel Server

ON-Bar Utilities

ON–Bar includes the following utilities. You can call **onbar** from the command line, a script, a scheduler such as **cron** (UNIX), or a storage-manager process.

Utility	Description	IDS	XPS
onbar	Is an editable shell script on UNIX and a batch file (**onbar.bat**) in Windows that starts the **onbar-driver**. Use the **onbar** script or batch file to check the storage-manager version and customize backup and restore operations.	X	X
onbar_d	When you use the **onbar** command, it calls the **onbar_d** utility that starts the **onbar-driver**. The **onbar-driver** starts and controls backup and restore activities. The **onbar_d** utility transfers data between Dynamic Server and the storage manager.	X	
onbar_w	Transfers data between an Extended Parallel Server coserver and the storage manager until the backup or restore request is fulfilled.		X
onbar_m	Collects and processes the backup emergency boot files from each coserver and creates the restore boot file.		X
start_worker.sh	Starts onbar-worker processes manually.		X
onsmsync	Synchronizes the contents of the **sysutils** database, the emergency boot files, and the storage manager catalogs. Can be used to purge backups that are no longer needed according to user-selectable policies.	X	X
ondblog	Changes the database-logging mode.	X	X
archecker	Verifies backups and restores table-level data	X	X

IBM Informix Storage Manager

ON–Bar is packaged with ISM. However, you can purchase a third-party storage manager if you prefer. You must use a storage manager to perform backups and restores with ON–Bar. The *storage manager* is an application that manages the storage devices and media that contain backups. The storage manager handles all media labeling, mount requests, and storage volumes.

The ISM server resides on the same computer as ON–Bar and the Informix database server; your storage devices are attached to this computer as well. ISM can store data on simple tape drives, optical disk devices, and file systems. ISM also performs the following functions:

- Configures up to four storage devices
- Adds, changes, and deletes administrative users
- Labels and mounts storage volumes on your storage devices
- Manages storage volumes
- Compresses and decompresses data
- Encrypts and decrypts data

For more information, see “Configuring a Third-Party Storage Manager” on page 4-1, “Choosing Storage Managers and Storage Devices” on page 4-15, Chapter 10, “ON-Bar Configuration Parameters,” on page 10-1, and the *IBM Informix Storage Manager Administrator's Guide*.

IBM Tivoli Storage Manager

IBM Tivoli Storage Manager (TSM) is a client/server program that provides storage management solutions to customers in a multivendor computer environment. TSM provides an automated, centrally scheduled, policy-managed backup, archive, and space-management facility for file servers and workstations.

TSM stores data on separate TSM servers. Your Informix database servers are TSM clients using the library that implement XBSA functions for using TSM with Informix database servers (Informix Interface for TSM). Informix Interface for TSM is part of your Informix database server installation. TSM is distributed separately.

TSM efficiently manages disk, optical, and tape library resources. TSM provides the following functions:

- Reduces network complexity with interfaces and functions that span network environments.
- Increases administrator productivity by automating repetitive processes, scheduling unattended processes, and administering TSM from anywhere in the network
- Reduces risk of data loss with scheduled routine backups
- Optimizes existing storage resources with automated movement of files from client file systems to TSM storage

TSM provides the following services:

- Backup and restore services to generate scheduled backups and restore data when required
- Archive and retrieve services to provide point-in-time copies of data for long-term storage
- Server hierarchical storage management services to automate migration from expensive storage media to less expensive storage media
- Automation services to automate common storage administration tasks
- Administration services to support routine monitoring, administration, and accounting, including the following functions:
  - Set client and server options
  - Define devices
  - Format storage volumes
  - Add additional clients
  - Label tape volumes
- Security services to control user access
- Disaster recovery management to implement a comprehensive backup and recovery procedures

Third-Party Storage Managers

Some third-party storage managers can manage stackers, robots, and jukeboxes as well as simple tape and disk devices. These storage managers might perform these additional functions:

- Schedule backups

- Support networked and distributed backups and restores

Find information on the third-party storage managers that ON–Bar supports at http://www.ibm.com/software/data/informix/support.

Make sure that the storage manager has passed the Informix validation process. The validation process is specific to the backup and restore product version, the operating-system version, and the Informix database server version.

XBSA Interface

ON–Bar and the storage manager communicate through the X/Open Backup Services Application Programmer's Interface (XBSA), which enables the storage manager to manage media for the database server. By using an open-system interface to the storage manager, ON–Bar can work with a variety of storage managers that also use XBSA.

Each storage manager develops and distributes a unique version of the XBSA shared library. You must use the version of the XBSA shared library provided with the storage manager. For example, if you use ISM, use the XBSA shared library provided with ISM. ON–Bar and the XBSA shared library must be compiled the same (32-bit or 64-bit).

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 the database server and in the proper order, and to track the history of backup objects.
- **Backup or restore data.** During backups and restores, ON–Bar and the storage manager use XBSA to exchange data from specified storage spaces or logical-log files.

ON–Bar uses XBSA transactions to ensure data consistency. All operations included in a transaction are treated as a unit. All operations within a transaction must succeed for objects transferred to the storage manager to be restorable.

ON-Bar Tables

ON–Bar uses the following catalog tables in the **sysutils** database to track backup and restore operations:

- The **bar_server** table tracks instances of the database server.
- The **bar_object** table tracks backup objects. A *backup object* is a backup of a dbspace, blobspace, sbspace, or logical-log file.
- The **bar_action** table tracks all backup and restore attempts against each backup object, except some log salvage and cold restore events.
- The **bar_instance** table describes each object that is backed up during a successful backup attempt.

The **onsmsync** utility uses the following tables to track its operations:

- The **bar_ixbar** table contains history of all unexpired successful backups in all timelines. It is maintained and used by **onsmsync** only.
- The **bar_syncdeltab** table is normally empty except when **onsmsync** is running. It is maintained and used by **onsmsync** only.

For a description of the content of these tables, see Chapter 11, "ON-Bar Catalog Tables," on page 11-1.

ON-Bar Boot Files

The ON–Bar *emergency boot files* reside in the **$INFORMIXDIR/etc** directory on UNIX and in the **%INFORMIXDIR%\etc** directory on Windows. The emergency boot files contain the information that you need to perform a cold restore and are updated after every backup.

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, the database server is not available to access **sysutils**, so ON–Bar obtains the information it needs for the cold restore from the emergency boot file.

Warning: Do **NOT** modify the emergency boot file(s) in any way. Doing so might cause ON-Bar to select the wrong backup as part of a restore, possibly leading to data corruption or system failure. Removing or modifying emergency boot file entries for logical log files is particularly discouraged.

ON-BAR Boot File on Dynamic Server

ON–Bar uses one emergency boot file on Dynamic Server. The filename for the boot file is **ixbar.***servernum*, where **servernum** is the value of the SERVERNUM configuration parameter.

You can override the default path and name of the boot file by changing the information specified in the BAR_IXBAR_PATH configuration parameter.

ON-BAR Boot File Extended Parallel Server

Table 3-3 lists the types of emergency boot files that Extended Parallel Server uses. Each node with a storage manager contains one backup boot file and one restore boot file. Multiple coservers on a node share a backup boot file and a restore boot file. The database server has one merge boot file.

Use the BAR_BOOT_DIR configuration parameter to specify the location of the emergency boot files. For more information, see "BAR_BOOT_DIR (XPS)" on page 10-7. If you do not specify BAR_BOOT_DIR, the database server stores the boot files in **$INFORMIXDIR/etc**.

Table 3-3. XPS Emergency Boot Files

Boot File Type	Boot Filename	Description
Backup	**Bixbar_***hostname.servernum*	This file contains backup information and is updated after every backup.
Restore	**Rixbar_***hostname.servernum*	The **onbar-merger** process recreates the restore boot files, which the **onbar-worker** processes use during a cold restore.
Merge	**Mixbar_***hostname.servernum*	The **onbar-merger** process recreates the merge boot file during a cold restore. The merge boot file is temporary and is removed when the **onbar_d** process that created it exits.

During the cold-restore process, ON–Bar follows these steps to create a restore boot file and restore data.

To create the emergency boot files in a cold restore:

1. It merges the backup boot files from all coservers and creates a merge boot file for the restore.
2. It distributes the merge boot file to each coserver, renaming it as the restore boot file and overwriting the old restore boot files.
3. It uses the information in the restore boot file instead of the information in the **sysutils** database to determine which backup copy of each storage space and log to use.

ON-Bar Activity Log

ON–Bar writes informational, progress, warning, error, and debugging messages to the ON–Bar *activity log*. You can use the activity log to:

- Monitor backup and restore activities such as, which storage spaces and logical logs were backed up or restored, the progress of the operation, and approximately how long it took.
- Verify whether a backup or restore succeeded.
- Track errors from the **ondblog** utility.
- Track ON-Bar performance statistics

For a list of ON–Bar informational, warning, and error messages, use the **finderr** or **Find Error** utility or view *IBM Informix Error Messages* at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp.

Specifying the Location of the Activity Log

For information on how to change the location of the ON–Bar activity log, see "BAR_ACT_LOG" on page 10-6.

Specifying the Level of ON-Bar Debugging (IDS)

You can set the level of debugging messages the ON-Bar utility prints in the ON-Bar debug log with the BAR_DEBUG configuration parameter. The ON-Bar debug log is specified by the BAR_DEBUG_LOG configuration parameter. By default, no debugging messages are printed.

You can update the value of BAR_DEBUG by editing the ONCONFIG file. When the updated value of BAR_DEBUG takes affect depends on your database server:

- For Dynamic Server, the new value of BAR_DEBUG takes affect immediately for any currently executing ON-Bar command and any subsequent commands. Any ON-Bar command that is currently executing when you update BAR_DEBUG reads the new value of BAR_DEBUG and prints debug messages at the new level.
- For Extended Parallel Server, the new value of BAR_DEBUG takes affect for ON-Bar commands executed after you update the value of BAR_DEBUG.

Specifying the Location of the Debug Log

For information on how to change the location of the ON-Bar debug log, see "BAR_DEBUG_LOG" on page 10-10.

Monitoring the Progress of a Backup or Restore

If your backup or restore operations take a long time to complete, knowing the progress is especially useful. Use the BAR_PROGRESS_FREQ configuration parameter to specify, in minutes, the frequency of the progress messages written to

the ON–Bar activity log. For information on how to change the frequency of the progress messages, see “BAR_PROGRESS_FREQ” on page 10-13.

Monitoring Backup and Restore Performance

You can monitor ON-Bar performance with the BAR_PERFORMANCE configuration parameter. This parameter helps you determine the level of performance reporting in the ON-Bar activity log. You can track the performance of ON-Bar processing and the performance of transferring objects between ON-Bar and the storage manager. See “BAR_PERFORMANCE” on page 10-13 to learn how to set the level of reporting.

Transform Data with External Filter Programs

You can use external programs to transform data to a different format prior to a backup and transform the data back to its original format following a restore. These programs are called *filters*. Filters can be used for compression or other data transformations.

ON-Bar and **ontape** both call the filters with the path specified by the BACKUP_FILTER and RESTORE_FILTER configuration parameters.

Transforming with Filters during Backup and Restore

You can transform data during backup and restore by calling external programs as filter plug-ins. To use filters:

1. Define the BACKUP_FILTER and RESTORE_FILTER configuration parameters in the `onconfig` file.

   ```
   BACKUP_FILTER path_name options
   RESTORE_FILTER path_name options
   ```

2. Optionally, to identify a unique server instance, use the three following parameters:

   INFORMIXSERVER
   Name of the IDS server

   SERVERNUM
   A configuration parameter in the onconfig file.

   Hostname
   The machine name.

For example, you can use the UNIX compression utility to compress data before backing it up and decompress it following restore:

- To compress the data, use the BACKUP_FILTER parameter. For example:

  ```
  BACKUP_FILTER  /bin/compress
  ```

  With this backup filter configuration, the backup filter is called from **ontape** or ON-Bar as:

  ```
  /bin/compress
  ```

  The output produced by this filter is saved as a single object to the storage manager.

- To restore transformed data, use the RESTORE_FILTER parameter.

  Prerequisite: The data must have previously been transformed with the BACKUP_FILTER parameter,

  For example:

  ```
  RESTORE_FILTER  /bin/uncompress
  ```

If this object is used for restoring, the RESTORE_FILTER is called as follows during restore:

```
/bin/uncompress
```

The data passed to the filter to be on uncompressed, was compressed by the BACKUP_FILTER.

Chapter 4. Configuring the Storage Manager and ON-Bar

In This Chapter 4-1
Configuring a Storage Manager 4-1
Configuring a Third-Party Storage Manager 4-1
Configuring ISM 4-2
Configuring TSM 4-2
Editing the TSM Client Options Files 4-3
Assigning a TSM Management Class for a Backup 4-3
Registering with the TSM Server 4-4
Initializing the Informix Interface for TSM Password 4-4
Updating the sm_versions File 4-4
Configuring Multiple Storage Managers on Coserver Nodes (XPS) 4-5
Validating Your Storage Manager 4-6
Configuring ON-Bar 4-6
Creating the bargroup Group (UNIX) 4-6
Your Customized onbar Script is Saved on New Installations 4-6
Setting ISM Environment Variables and ONCONFIG Parameters 4-7
Setting the Interface for TSM Environment Variables 4-7
Specifying the Location of the XBSA Library 4-8
Specifying ON-Bar Configuration Parameters 4-9
ON–Bar Configuration Parameters on Dynamic Server 4-9
Parameters on Extended Parallel Server 4-10
Configuring Multiple Storage Managers 4-10
Global Configuration Parameters 4-11
Storage-Manager Specific Configuration Parameters 4-11
Examples of ON-Bar and Storage-Manager Configurations 4-12
Verifying the Configuration 4-15
Choosing Storage Managers and Storage Devices 4-15
Features That ISM Supports 4-16
Features That ISM Does Not Support 4-16
Features That TSM Supports 4-16
Storage Device Requirements 4-17
Considerations for Extended Parallel Server 4-17

In This Chapter

This chapter provides the information that you need to plan and to set up ON–Bar with a storage manager:

- Installing and configuring a storage manager
- Configuring ON–Bar
- Steps to take before making a test backup
- Choosing storage managers and storage devices

Configuring a Storage Manager

This section discusses installing and configuring a storage manager.

Configuring a Third-Party Storage Manager

Storage managers have slightly different installation and configuration requirements. Make sure that you follow the manufacturer's instructions carefully. If you have difficulty with the storage-manager installation and configuration, please contact the manufacturer directly. For the list of certified storage managers for your ON–Bar version, consult your sales representative.

© Copyright IBM Corp. 1996, 2008

Important: Some storage managers let you specify the kind of data to back up to specific storage devices. Configure the storage manager to back up logical logs to one device and storage spaces to a different device for more efficient backups and restores.

To configure a third-party storage manager:

1. Set ON–Bar configuration parameters and environment variables.
2. Configure the storage manager so that ON–Bar can communicate correctly with it. For information, see your storage-manager documentation.
3. Configure your storage devices.

   To configure your storage devices, follow instructions in your storage-manager documentation. The storage manager must know the device names of the storage devices that it should use.
4. Label your storage volumes.
5. Mount the storage volumes on the storage devices.
6. Update the storage-manager definition in the **sm_versions** file. For more information, see "Updating the sm_versions File" on page 4-4.
7. Verify that the BAR_BSALIB_PATH configuration parameter points to the correct XBSA shared library for your storage manager. For more information, see "Specifying the Location of the XBSA Library" on page 4-8.

After you configure the storage manager and storage devices and label volumes for your database server and logical-log backups, you are ready to initiate a backup or restore operation with ON–Bar.

Configuring ISM

For instructions on how to set up ISM to work with ON–Bar, see the *IBM Informix Storage Manager Administrator's Guide*. The ISM server is installed with the Informix database server on UNIX or Windows. Several database server instances can share one ISM instance.

Warning: Install one copy of ISM on each computer to prevent possible conflicts with the XBSA shared library. Do not run ISM and Legato NetWorker on the same computer because they conflict with each other.

Configuring TSM

To use Tivoli Storage Manager with Informix databases, you must install and configure the Tivoli Storage Manager client on your database server computer and Tivoli Storage Manager on your storage computer.

For more information about TSM, read the following manuals:

- *Tivoli Storage Manager Backup-Archive Clients Installation and User's Guide*
- *Tivoli Storage Manager Using the Application Program Interface*
- *Tivoli Storage Manager Administrator's Guide*
- *Tivoli Storage Manager Administrator's Reference*

In addition, you must configure Informix Interface for TSM and perform other TSM configuration tasks on your Informix database server computer. These tasks are explained in the following sections.

Editing the TSM Client Options Files

The Informix Interface for TSM communicates with the TSM server using the TSM API. By default, Informix Interface for TSM uses the client user options file (**dsm.opt**) and client system options file (**dsm.sys**) located in the TSM API installation directory:

- Specify the TSM server to use in the client user options file, **dsm.opt**.
- Identify the TSM server name, communication method, and server options in the client system options file, **dsm.sys**.

Use the sample **dsm.opt.smp** and **dsm.sys.smp** files distributed with the TSM API to help you get started quickly.

You must be the **root** user to perform edits to the **dsm.opt** and **dsm.sys** files.

See *TSM Installing the Clients* and *TSM Trace Facility Guide* for information regarding options you can specify in these files.

Editing the TSM Client User Options File: The TSM client user options file, **dsm.opt**, must refer to the correct TSM server instance, as listed in the **dsm.sys** file.

Set the following options in the **dsm.opt** file:

- SERVERNAME: to identify which TSM server instance, as listed in the **dsm.sys** file, that Informix Interface for TSM contacts for services.
- TRACEFILE: to send trace output information to a designated file.
- TRACEFLAG: to set specific trace flags.

Editing the TSM Client System Options File: The TSM client systems options file, **dsm.sys**, must refer to the correct TSM server address and communication method.

The following TSM options are the most important to set in the **dsm.sys** file:

- SERVERNAME: to specify the name you want to use to identify a server when it is referred to in the **dsm.opt** file and to create an instance that contains options for that server
- COMMMETHOD: to identify the communication method
- TCPSERVERADDRESS: to identify the TSM server
- PASSWORDACCESS: to specify GENERATE to store the TSM password

The SERVERNAME option in the **dsm.opt** and **dsm.sys** files define server instance names only. The TCPSERVERADDRESS option controls which server is actually contacted.

You can set up multiple server instances in the **dsm.sys** file. See the *Tivoli Storage Manager Backup-Archive Client Installation and User's Guide* for information about multiple server instances.

Assigning a TSM Management Class for a Backup

When you back up a database, the default management class for your node is *used*. You can override the default value with a different value that is specified in the INCLUDE option. This option is placed in the **include-exclude** options file. The file name of the **include-exclude** options file is in the client system options file (**dsm.sys**). For more information, see the *Tivoli Storage Manager Backup-Archive Client Installation and User's Guide*.

Use the following naming conventions for ON-Bar files:

- A database backup:
  /dbservername/dbservername/dbspacename/level
- A log backup:
  /dbservername/dbservername/server_number/unique_logid

For a database backup, an example of the INCLUDE statement is as follows:

```
Include /dbserverA/dbserverA/dbspaceA/* InformixDbMgmt
```

For a logical log back up, an example of the INCLUDE statement is as follows:

```
Include /dbserverA/dbserverA/55/* InformixLogMgmt
```

where the number 55 is the value of the SERVERNUM parameter in the ONCONFIG file.

Registering with the TSM Server

Before backing up to and recovering from a TSM server, you must have a TSM registered node name and a password. The process of setting up a node name and password is called registration. After the Informix Interface for TSM node is registered with a TSM server, you can begin using the Informix Interface for TSM to back up and restore your Informix storage spaces and logical logs. If your workstation has a node name assigned to the TSM backup-archive client, you should have a different node name for Informix Interface for TSM. For information about performing the registration process, see the *Tivoli Storage Manager Backup-Archive Client Installation and User's Guide*.

Initializing the Informix Interface for TSM Password

To initialize the password for Informix Interface for TSM, use the **txbsapswd** program. This program sets up a connection with the server instance that you specified in the **dsm.opt** file. You must run the **txbsapswd** program as user **root** before using Informix Interface for TSM.

To initialize the password:

1. Start the **txbsapswd** program located in the **$INFORMIXDIR/bin** directory.
2. Enter the password and press Return. To retain your current password, press Return without a value.

Updating the sm_versions File

The storage manager must have an entry in the **sm_versions** file. If you are using ISM, put **ism** in the **sm_name** field of **sm_versions**. To find out which code name to use in **sm_versions** for third-party storage managers, see the storage-manager documentation.

The storage-manager definition in the **sm_versions** file uses this format:

```
1|XBSA_ver|sm_name|sm_ver
```

XBSA_ver is the release version of the XBSA shared library for the storage manager. *sm_name* is the name of the storage manager. *sm_ver* is the storage-manager version. The maximum field length is 128 characters.

The following example shows the ISM definition in the **sm_versions** file:

```
1|1.0.1|ism| ISM.2.20.UC1.114|
```

The following example shows the TSM definition in the **sm_versions** file:

```
1|5.3|adsm|5
```

Before ON–Bar starts a backup or restore process, it calls the currently installed version of the storage-manager-specific XBSA shared library to get its version number. If this version is compatible with the current version of ON–Bar and is defined in the **sm_versions** file, ON–Bar begins the requested operation.

To update the storage-manager definition in sm_versions:

1. Copy the **sm_versions.std** template to a new file, **sm_versions** in the **$INFORMIXDIR/etc** directory on UNIX or the **%INFORMIXDIR%\etc** directory on Windows.
2. If you are using ISM, issue the **ism_startup -init** command to automatically update **sm_versions** with the correct version number and storage-manager name or manually edit **sm_versions**.

   If you are installing an ISM patch, you must manually edit **sm_versions**.

   Warning: The **ism_startup -init** command erases records of previous backups.
3. If you are using a third-party storage manager, the vendor supplies the definition for the **sm_versions** file. Create your own **sm_versions** file with the correct data for the storage manager using the format in **sm_versions.std** as a template.

Extended Parallel Server

4. If all coservers share the **sm_versions** file in the **etc** subdirectory, the **sm_versions** file should have an entry for each storage-manager brand.

   If the **etc** subdirectory is not shared between coserver nodes, specify one line in the **sm_versions** file for the storage manager in use on that coserver node.

End of Extended Parallel Server

5. Stop any ON–Bar processes (**onbar_d**, **onbar_w**, or **onbar_m**) that are currently running and restart them for the changes to take effect.

Configuring Multiple Storage Managers on Coserver Nodes (XPS)

Extended Parallel Server allows multiple storage-manager instances, but only one instance per node. You can configure and use different storage-manager brands for different purposes. For best performance, run **onbar-worker** processes on all nodes that have storage devices. For example, you have two storage devices, a tape drive and a jukebox, and want to connect them on different nodes. When an **onbar-worker** is on each node, the data moves faster because it does not have to travel over the network. For complex examples, see "Parameters on Extended Parallel Server" on page 4-10.

For information on how to update the **sm_versions** file when using multiple storage managers, see step 4 in "To update the storage-manager definition in sm_versions" on page 4-5.

Some third-party storage managers have a client/server architecture. For these storage managers, *one instance per node* means one storage-manager client.

Important: Each hardware MPP node that contains a coserver that runs **onbar-worker** processes must have a local copy of the storage-manager version of the XBSA shared library.

Validating Your Storage Manager

When you convert or revert an Informix database server, the storage manager that you used on the old version might not be validated for the version that you are migrating to. Verify that the storage-manager vendor has successfully completed the Informix validation process for the database server version and platform. If not, you need to install a validated storage manager before you perform backups with ON–Bar.

Configuring ON-Bar

ON–Bar is installed with your Informix database server software. To use ON–Bar with installed storage managers, you set specific parameters in the ONCONFIG file. The following section describes the required ON–Bar configuration parameters.

Dynamic Server

Use the **onconfig.std** file as a template.

End of Dynamic Server

Extended Parallel Server

Use the **onconfig.std** file as a template for single coservers. Use the **onconfig.xps** file as a template for multiple coservers.

End of Extended Parallel Server

Creating the bargroup Group (UNIX)

If you want users other than or **root** to execute ON–Bar commands, you can create a **bargroup** group. Members of **bargroup** can execute ON–Bar commands. The **bargroup** group on UNIX is similar to the **-Admin** group on Windows. For instructions on how to create a group, see your UNIX documentation.

Important: For security, it is recommended that onbar commands not be run by the **root** user.

Your Customized onbar Script is Saved on New Installations

When the installation program installs the database server files, including the ON–Bar files, the **onbar** script is distributed as a shell script so that you can add preprocessing or postprocessing steps to the script.

The **onbar** script is distributed as a file named **onbar.sh** (UNIX) or **onbar.bat** (Windows). When the install program installs the database server files over an existing installation, it checks whether any difference exists between the new **onbar** script and the old **onbar** script to prevent the loss of your existing **onbar** script.

- If the two scripts are the same, the installation program renames the **onbar.sh** or **onbar.bat** file to **onbar**, the new **onbar** script overwrites the old **onbar** script, and no data is lost.

- If a difference exists between the new **onbar** script and the old **onbar** script, the installation program renames the **onbar.sh** or **onbar.bat** file to **onbar**, renames the old **onbar** script to the form **onbar.*date***, and issues a message that the existing **onbar** script was renamed.

If you see a message that the old **onbar** script has been renamed by appending a date, look at the new **onbar** script (filename **onbar**) and integrate the contents of the old **onbar** script into the new **onbar** script. For example, if **onbar** has been renamed to **onbar.2000.12.15**, integrate the contents of **onbar.2000.12.15** into **onbar**.

For information on using the **onbar** script, see "Customizing ON-Bar and Storage-Manager Commands" on page 9-2. For information on installing the database server, see your *IBM Informix Installation Guide*.

Setting ISM Environment Variables and ONCONFIG Parameters

When you use ISM, you need to set certain environment variables. For information, see the *IBM Informix Storage Manager Administrator's Guide*.

Dynamic Server

You can set these environment variables in the **onbar** script or in your environment.

End of Dynamic Server

Extended Parallel Server

You can set these environment variables in your environment if you start **onbar -w** manually or before you start the database server, or set them in **start_worker.sh**.

End of Extended Parallel Server

If you use ISM, you can specify the volume pool names for storage spaces and logical logs in the ISM_DATA_POOL and ISM_LOG_POOL parameters in the ONCONFIG file. The ISM_DATA_POOL configuration parameter specifies the volume pool that you use for backing up storage spaces. The ISM_LOG_POOL configuration parameter specifies the volume pool that you use for backing up logical logs.

If you do not set these configuration parameters, they default to the volume pool names ISMData and ISMLogs, respectively.

Setting the Interface for TSM Environment Variables

When you use the Interface for TSM, you need to set certain environment variables in the user's environment. The following table describes these environment variables.

Table 4-1. Interface for TSM Environment Variables

Environment Variable	Description
DSMI_CONFIG	The fully qualified name for the client user option file (**dsm.opt**). The default value is **dsm.opt** in the TSM API installation directory.

Table 4-1. Interface for TSM Environment Variables (continued)

Environment Variable	Description
DSMI_DIR	Points to the TSM API installed path. This environment variable needs to be defined only if the TSM API is installed in a different path from the default path. The DSMI_DIR environment variable is also used to find the **dsm.sys** file.
DSMI_LOG	Points to the directory that contains the API error log file (**dsierror.log**). For error log files, create a directory for the error logs to be created in, then set the DSMI_LOG environment variable to that directory. The API error log file must have writable rights by the user performing the backup.

The following example shows how to set up these environment variables for Solaris 32-bit if the TSM API is installed in the **/opt/Tivoli/tsm/client/api** directory:

```
export DSMI_CONFIG=/opt/Tivoli/tsm/client/api/bin/dsm.opt
export DSMI_DIR=/opt/Tivoli/tsm/client/api/bin
export DSMI_LOG=/home/user_a/logdir
```

Specifying the Location of the XBSA Library

UNIX Only

By default, ON–Bar looks for the XBSA shared library in **$INFORMIXDIR/lib/ibsad001.s[ol]** on UNIX. To specify a different name or location of the XBSA shared library, use the BAR_BSALIB_PATH configuration parameter. You can also make **$INFORMIXDIR/lib/ibsad001.s[ol]**a symbolic link to the correct library.

For example, if you are using ISM, you can do either of the following:

- Link **$INFORMIXDIR/lib/ibsad001.so** to **$INFORMIXDIR/lib/libbsa.so**
- Set BAR_BSALIB_PATH to **$INFORMIXDIR/lib/libbsa.so**

For example, if you are using TSM, you can do either of the following:

- Link **$DIR/lib/ibsad001.so** to **$DIR/lib/libtxbsa.so**
- Set BAR_BSALIB_PATH to **$DIR/lib/libtxbsa.so**

End of UNIX Only

Windows Only

On Windows, because no default XBSA shared library name exists, you must specify its name and location in the BAR_BSALIB_PATH configuration parameter. If you are using ISM, set BAR_BSALIB_PATH to **%ISMDIR%\bin\libbsa.dll**. If you are using TSM, set BAR_BSALIB_PATH to %DIR%\bin\libtxbsa.dll.

End of Windows Only

If you are using a third-party storage manager, ON–Bar must use the version of the XBSA library that the storage-manager manufacturer provides. For more

information, see “BAR_BSALIB_PATH” on page 10-8 and your release notes.

Extended Parallel Server

The XBSA library must be present on each coserver node where you are running **onbar-worker** processes. Each **onbar_worker** needs to dynamically link to the functions in the XBSA library. The XBSA library must be either on a local disk or NFS-mounted disk.

To find out whether **onbar_worker** processes can share libraries and whether the sharing can be done through NFS, check your storage-manager documentation.

You can specify BAR_BSALIB_PATH in the global section of the ONCONFIG file if you configure:

- The XBSA shared library to have the same path on all nodes
- Storage managers from more than one vendor if each shared XBSA library has the same path on each node, which is not NFS-mounted

  If each XBSA library uses a different path, you must specify BAR_BSALIB_PATH in each storage-manager-specific section of the ONCONFIG file.

End of Extended Parallel Server

Important: To set the path name of the XBSA library with the BAR_BSALIB_PATH configuration parameter in the ONCONFIG file, specify the absolute path name. If you specify a relative path name, then the following message is written to the ON-Bar activity log: `BAR_BSALIB_PATH is ONCONFIG is not an absolute path name.`

Specifying ON-Bar Configuration Parameters

Before you begin your first backup, review the default ON–Bar parameters in the ONCONFIG file and adjust the values as needed. For more information, see “Verifying the Configuration” on page 4-15. For the complete list of database server configuration parameters and their default values, see the *IBM Informix Administrator's Reference*.

ON–Bar Configuration Parameters on Dynamic Server

ON–Bar on Dynamic Server uses the following configuration parameters.

Configuration Parameter	Purpose
“ALARMPROGRAM” on page 10-5	Specifies a script that handles alarms For ON–Bar, set this script to **log_full.sh** to automatically back up log files when they become full.
“ALRM_ALL_EVENTS (IDS)” on page 10-5	Causes ALARMPROGRAM to execute every time an alarm event is invoked
“BACKUP_FILTER” on page 10-6	Specifies the location and name of an external filter program used in data transformation.
“BAR_ACT_LOG” on page 10-6	Specifies the location and name for the ON–Bar activity log file
“BAR_BSALIB_PATH” on page 10-8	Specifies the full path and name of the XBSA shared library provided by the storage manager to communicate between ON-Bar and the storage manager.

Configuration Parameter	Purpose
"BAR_DEBUG" on page 10-9	Specifies the level of debugging information to display in the ON-Bar activity log file You can dynamically update the value of BAR_DEBUG in the ONCONFIG file during a session
"BAR_DEBUG_LOG" on page 10-10	Specifies the location and name of the ON-Bar debug log
"BAR_IXBAR_PATH (IDS)" on page 10-11	Specifies the location where the ON-Bar **ixbar** boot file is created. You can change the file name and path.
"BAR_HISTORY" on page 10-10	Specifies whether the **sysutils** database maintains the backup history
"BAR_MAX_BACKUP (IDS)" on page 10-12	Specifies the maximum number of processes per **onbar** command
"BAR_NB_XPORT_COUNT (IDS)" on page 10-12	Specifies the number of shared-memory data buffers for each **onbar_d** worker or child process
"BAR_PERFORMANCE" on page 10-13	Specifies whether timestamps and transfer rates of storage-manager operations are recorded in the activity log.
"BAR_PROGRESS_FREQ" on page 10-13	Specifies in minutes how frequently the backup or restore progress messages display in the activity log
"BAR_RETRY" on page 10-14	Specifies how many times ON–Bar should retry a backup, logical-log backup, or restore operation if the first attempt fails
"BAR_XFER_BUF_SIZE (IDS)" on page 10-17	Specifies the size in pages of the buffers that the database server uses to exchange data with each **onbar_d** worker or child process
"ISM_DATA_POOL" on page 10-19	Specifies the volume pool that you use for backing up storage spaces (ISM)
"ISM_LOG_POOL" on page 10-19	Specifies the volume pool that you use for backing up logical logs (ISM)
"LTAPEDEV (IDS)" on page 10-20	For **ontape**, specifies the tape device where logical logs are backed up For ON–Bar, specifies whether to back up logs. If this configuration parameter is set to **/dev/null** on UNIX or NUL on Windows, the backup utility will been able to back up the logical logs.
"RESTARTABLE_RESTORE (IDS)" on page 10-21	Turns restartable restore on or off
"RESTORE_FILTER" on page 10-21	Specifies the location and name of an external filter program that restores transformed data to its original state

Important: ON–Bar does not use the TAPEDEV, TAPEBLK, TAPESIZE, LTAPEBLK, and LTAPESIZE configuration parameters.

Parameters on Extended Parallel Server

The **ONCONFIG** file contains a section for global parameters and individual sections for each storage-manager instance. You might need to specify multiple instances of storage managers to back up and restore data to all the coservers in Extended Parallel Server.

Configuring Multiple Storage Managers

In Extended Parallel Server, you can use more than one storage-manager product, as follows:

- Different versions of a particular storage manager
- Storage managers from different vendors

If you use the **onconfig.std** template to configure a single coserver with one storage manager, copy the section "**Storage-Manager instances**" from **onconfig.xps** into your **ONCONFIG** file. Use the **onconfig.xps** template to configure multiple storage managers.

Warning: Be careful not to get the shared libraries for the two products or versions mixed up.

Global Configuration Parameters

The global section includes parameters that apply to all storage managers. You can include ON–Bar parameters in the global section if they are the same for all storage-manager instances.

Put these parameters in the storage-manager section between the BAR_SM and END pair if they are different for each storage-manager instance.

Storage-Manager Specific Configuration Parameters

You must define each storage-manager client that you install and configure in the storage-manager section, as illustrated in "Defining a Storage Manager on a Five-Coserver System" on page 4-13.

Configuration Parameter	Reference	Purpose	Can Be Storage-Manager Specific	Always Storage-Manager Specific	Always Global
BAR_ACT_LOG	10-6	Specifies the location and name for the ON–Bar activity log file.			X
BAR_BOOT_DIR	10-7	Specifies the directory for the emergency boot files			X
BAR_BSALIB_PATH	10-8	Specifies the path of the storage-manager library To determine if BAR_BSALIB_PATH is supported on your platform, check your release notes. Specify BAR_BSALIB_PATH in the storage-manager section if the libraries are not in the same location on all nodes.	X		
BAR_DBS_COSVR	10-9	Specifies coservers that send backup and restore data to the storage manager		X	
BAR_HISTORY	10-10	Specifies whether the **sysutils** database maintains a backup history			X
BAR_IDLE_TIMEOUT	10-10	Specifies the maximum number of minutes that an **onbar-worker** process is idle before it is shut down	X		
BAR_LOG_COSVR	10-11	Specifies coservers that send log backup data to the storage manager		X	

Configuration Parameter	Reference	Purpose	Can Be Storage-Manager Specific	Always Storage-Manager Specific	Always Global
BAR_PROGRESS_FREQ	10-13	Specifies in minutes how frequently the backup or restore progress messages display in the activity log			X
BAR_RETRY	10-14	Specifies how many times ON–Bar should retry a backup, logical-log backup, or restore operation if the first attempt fails			X
BAR_SM	10-16	Specifies the storage-manager number Is required; starts the storage-manager section.		X	
BAR_SM_NAME	10-16	Specifies the storage-manager name		X	
BAR_WORKER_COSVR	10-16	Lists the coservers that can access the storage manager This parameter is required.		X	
BAR_WORKER_MAX	10-16	Specifies the maximum number of **onbar-worker** processes that the Backup Scheduler can start for this storage-manager instance You can start additional **onbar-worker** processes manually.	X		
BAR_XFER_BUFSIZE	10-18	Specifies the size in pages of the buffers used between XPS and each **onbar-worker** process			X
BAR_XPORT_COUNT	10-18	Specifies the number of shared-memory data buffers for each **onbar-worker** process			X
ISM_DATA_POOL	10-19	Specifies the volume pool that you use for backing up storage spaces	X		
ISM_LOG_POOL	10-19	Specifies the volume pool that you use for backing up logical logs	X		
LOG_BACKUP_MODE	10-20	Specifies whether to back up full logical-log files automatically or manually			X

Examples of ON-Bar and Storage-Manager Configurations

This section shows sample storage-manager configurations for Extended Parallel Server. For more information about each configuration parameter, refer to Chapter 10, "ON-Bar Configuration Parameters," on page 10-1.

Creating a Storage-Manager Definition: The following configuration example is for a storage manager that can run on coservers 1, 2, 3, 4, and 7. In this configuration, you have to start **onbar-worker** processes manually on coservers 1,

2, 3, 4, and 7, because BAR_WORKER_MAX is not set. All storage spaces and logical logs are backed up to this storage-manager instance.

```
# Backup/Restore Variables
BAR_ACT_LOG       /tmp/bar_act.log       # Path of activity log
BAR_RETRY         2     # Number of times to retry failures
BAR_XPORT_COUNT   10    # Number of transport buffers per worker
BAR_XFER_BUFSIZE  8     # Size of each transport buffer in pages
LOG_BACKUP_MODE   CONT  # Backup as soon as logical log fills
BAR_IDLE_TIMEOUT  5     # How long onbar-workers wait
BAR_BSALIB_PATH  /usr/lib/ibsad001.so  # XBSA shared lib path

# Storage-Manager Section
BAR_SM 1
BAR_WORKER_COSVR 1-4,7
END
```

Defining a Storage Manager on a Five-Coserver System: The following example is a simple storage-manager definition that automatically starts a single **onbar-worker** process on coserver 1. Data on coservers 1 through 5 is backed up or restored to the storage manager on coserver 1. If you omit the BAR_WORKER_MAX parameter, you must start **onbar-worker** processes manually. For more information, see "Using start_worker.sh to Start onbar_worker Processes Manually (XPS)" on page 9-4.

```
# Storage Manager instances
BAR_SM           1      # Storage manager ID
  BAR_SM_NAME      A    # Storage manager name
  BAR_WORKER_COSVR 1    # Storage mgr is on coserver 1
  BAR_DBS_COSVR    1-5  # Route dbspaces to this storage mgr
  BAR_LOG_COSVR    1-5  # Route logs to this storage mgr
  BAR_WORKER_MAX   1    # Number of onbar-workers
END
```

Defining the Number of onbar-worker Processes on Two Storage Managers: The following example defines different storage managers on two coservers. Because the global BAR_WORKER_MAX value is 3, the Backup Scheduler will start up to three **onbar-worker** processes on coserver 2 for storage-manager **BAKER**. For storage-manager **ABEL**, the local BAR_WORKER_MAX value overrides the global setting, so the Backup Scheduler will start only one **onbar-worker** process.

```
# Global section
BAR_WORKER_MAX     3   # Global value for no. of onbar-workers

# Storage Manager ABEL
BAR_SM  1
  BAR_WORKER_MAX     1 # only one onbar-worker defined
  BAR_DBS_COSVR      1
  BAR_LOG_COSVR      1
  BAR_WORKER_COSVR   1
END

# Storage Manager BAKER
BAR_SM  2
  BAR_DBS_COSVR      2
  BAR_LOG_COSVR      2
  BAR_WORKER_COSVR   2
END
```

Important: Do not switch the BAR_SM identification numbers between different storage managers when you reconfigure the storage managers, or else

ON–Bar cannot find the backup objects. For example, do not reassign BAR_SM 1 to storage-manager **BAKER** and BAR_SM 2 to storage-manager ABEL.

Defining Three Storage Managers and Storage Devices: The configuration in Figure 4-1 on page 4-14 shows how you might set up three storage managers and three backup devices:

- A, a silo with two drives and two connections, one to coserver 1 and the other to coserver 3
- B, a tape autochanger connected to coserver 4
- C, a simple tape drive connected to coserver 2

Storage-manager **ABEL** can back up and restore storage spaces on coservers 1 and 3. Storage-manager **BAKER** can back up and restore storage spaces on coservers 4 and 2. Storage-manager **CHARLY** can back up and restore logs on all four coservers.

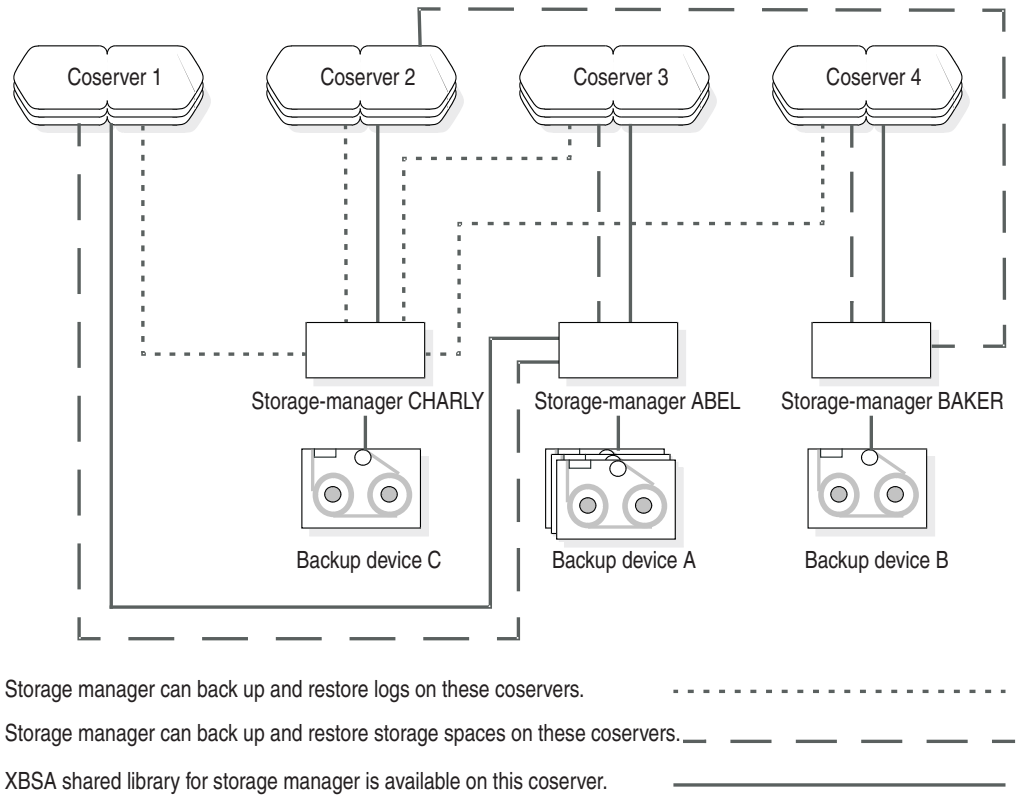


Figure 4-1. Storage-Manager Configuration

The ONCONFIG definitions for storage-managers **ABEL**, **BAKER**, and **CHARLY** appear in the following example:

```
# Storage manager section for storage manager A
BAR_SM 1
BAR_SM_NAME ABEL
BAR_WORKER_COSVR 1,3
BAR_DBS_COSVR 1,3
BAR_LOG_COSVR 0
BAR_WORKER_MAX 2
END
# Storage manager section for storage manager B
BAR_SM 2
```

```
BAR_SM_NAME BAKER
BAR_WORKER_COSVR 4
BAR_DBS_COSVR 2,4
BAR_LOG_COSVR 0
BAR_WORKER_MAX 1
END
# Storage manager section for storage manager C
BAR_SM 3
BAR_SM_NAME CHARLY
BAR_WORKER_COSVR 2
BAR_DBS_COSVR 0
BAR_LOG_COSVR 1 - 4
BAR_WORKER_MAX 1
END
```

Verifying the Configuration

Check the items in the following list to make sure that ON–Bar and your storage manager are set up correctly:

- The storage manager is installed and configured to manage specific storage devices.

UNIX Only

- Make sure that the BAR_BSALIB_PATH configuration parameter specifies correctly the XBSA shared library or it is not set and the library is in the default location.

End of UNIX Only

Windows Only

- Make sure that the BAR_BSALIB_PATH configuration parameter specifies correctly the XBSA shared library.

End of Windows Only

- The **sm_versions** file contains a row that identifies the version number of the storage-manager-specific XBSA shared library.

Extended Parallel Server

- The BAR_WORKER_MAX parameter is set to a number greater than `0` in the storage-manager-specific section of the ONCONFIG file.

End of Extended Parallel Server

After you verify that ON–Bar and your storage manager are set up correctly, run ON–Bar on your test database to make sure that you can back up and restore data. For more information, follow the instructions in Chapter 5, "Backing Up with ON-Bar," on page 5-1.

Choosing Storage Managers and Storage Devices

The storage manager manages the storage devices to which the backed-up data is written. ISM is included with your database server. For information on how to use ISM, refer to the *IBM Informix Storage Manager Administrator's Guide*.

If you choose a different storage manager, consider whether it has the features that you need to back up your storage spaces and logical logs. When you choose storage devices, make sure that they are compatible with the storage manager that you choose. The storage devices should have the speed and capacity that your backups require. The storage manager should be easy to use and work on your operating system.

Features That ISM Supports

ISM fulfills the following storage-manager requirements:

- ISM allows you to back up logical logs and storage spaces to different devices and to specify whether to use encryption or compression for data.
- ISM can write the output of parallel backups to a single device, medium, or volume. Some backup devices can write data faster than the disks used to hold storage spaces can be read.
- ISM can automatically switch from one tape device to another when the volume in the first device fills.
- ISM allows migration of data from one backup medium to another.

  For speed, you can back up logical logs or storage spaces to disk, but you must move them later to tape or other removable media or your disk will become full.
- ISM allows you to clone copies of backup data for on-site and off-site storage.
- ISM uses automatic expiration of data. Once all data on a backup media expires, you can reuse the media.

Features That ISM Does Not Support

ISM does not support the following features. Third-party storage managers might support these features.

- Distributing a single data stream across multiple devices simultaneously, which improves throughput if you have several slow devices
- Using different encryption or compression methods for specified storage spaces or databases
- Scheduling backups
- Support for devices such as tape libraries, jukeboxes, silos, tape autochangers, and stackers
- Remote host operations

  You can install some storage managers on a different host from the database server. However, ISM must be installed on the same host as the database server.

Features That TSM Supports

TSM supports all the features that ISM supports and all the features listed in the previous section that ISM does not support. TSM supports the following additional features:

- TSM allows you to create policies to automate storage management and enforce data management goals.
- TSM allows automated circulation of media through the storage management process.
- TSM allows you to implement a progressive backup methodology so that files are backed up incrementally to reduce network traffic, while recovery media is consolidated to provide better performance.
- TSM can use the Network Data Management Protocol to back up and restore file systems stored on a network-attached storage file server.

Storage Device Requirements

Ask the following interrelated questions to determine what storage devices you need. For example, the speed and type of storage devices partly determine the number of storage devices that you need.

- What kind of storage devices do you need?

  The transaction volume and the size of your database are major factors in determining the kind of storage devices that you need.

  ISM supports simple tape devices such as QIC, 4mm, 8mm, DLT, optical devices, and disk backups. If ISM cannot manage the storage devices that you need, you need to purchase a different storage manager. For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

- What is the availability requirement for each device?

  Is it important for your storage devices to allow random as well as sequential access? If so, you cannot use tape storage devices.

- How many storage devices do you need?

  ISM supports up to four devices per host. The number of storage devices that you need depends on the kind of storage devices you have, how much transaction activity occurs on the database server, how fast throughput is, how much time you can allow for backups, and other similar factors.

Extended Parallel Server

- How many and what type of storage-manager instances should you configure?

  You can have one storage manager on each coserver node.

End of Extended Parallel Server

Considerations for Extended Parallel Server

Because backing up data on multiple coservers is much more complex, some Extended Parallel Server users use third-party storage managers. The following usage requirements also affect your decisions about the storage manager and storage devices:

- The number of hardware nodes, the number of coservers on those nodes, and how the coservers are distributed across the nodes

  Balance these factors against the number of storage devices and storage-manager instances. The architecture of some platforms limits where you can attach devices, but the number of coservers increases processing requirements. The storage-manager sections of the ONCONFIG file should reflect these considerations.

  Although some nodes in a massively parallel processing (MPP) system might not be running coservers, they might be able to run part of the storage manager.

- The kind of high-speed interconnect on the MPP system

  Because disks are slower than the high-speed interconnect, they could create a bottleneck in the interconnect. Distributing devices across multiple nodes might reduce the amount of traffic across the interconnect and allow more parallelism.

- If you isolate tables or databases in a single storage space or in a dbslice across coservers, you can restore single tables or databases.

- Whether it is possible to restore data from external sources

  Although Decision Support System (DSS) databases might not be mirrored, they might be easier to recreate from the original external source than to restore from

backups if they are corrupted or damaged. For Online Transaction Processing (OLTP) systems, regenerating data from external sources is rarely possible.

- The size of each logical-log stream, how the transaction activity is distributed across logical-log streams, and when it occurs

  If you are running an OLTP system with many transactions evenly distributed across coservers, your storage manager and storage-device requirements are different from a DSS, which usually generates few transactions.

  In addition, if logstreams are the same size on each coserver but activity is not evenly distributed, space and resources are wasted. You should adjust them for efficiency.

Chapter 5. Backing Up with ON-Bar

In This Chapter . . . 5-2
Summary of ON-Bar Tasks . . . 5-2
Preparing for a Backup . . . 5-4
Data ON-Bar Backs Up . . . 5-4
Administrative Files to Back Up . . . 5-4
Installing and Configuring a Storage Manager . . . 5-5
Whole-System Backup (IDS) . . . 5-5
Parallel Backup . . . 5-5
Standard Backup . . . 5-5
Incremental Backup . . . 5-6
Physical Backup . . . 5-6
Choosing a Backup Level . . . 5-6
Level-0 Backups . . . 5-6
Level-1 Backups . . . 5-7
Level-2 Backups . . . 5-7
Collecting Information About Your System Before a Backup . . . 5-7
Ensuring That You Have Enough Logical-Log Space . . . 5-7
Copying Database Server Configuration Information . . . 5-8
Verifying Database Integrity . . . 5-8
Backing Up Storage Spaces and Logical Logs . . . 5-8
Backup Syntax . . . 5-8
Backing Up After Changing the Physical Schema . . . 5-10
When to Back Up the Root Dbspace and Modified Storage Spaces . . . 5-10
When to Back Up the Modified Storage Spaces Only . . . 5-11
Using ISM During a Backup . . . 5-11
Backing up the ISM Catalog . . . 5-12
Using IBM Server Administrator to Back Up and Verify . . . 5-12
ON-Bar Backup Examples . . . 5-12
Performing a Level-0 Backup of All Storage Spaces . . . 5-13
Performing a Level-0 Backup of Specified Storage Spaces . . . 5-13
Performing an Incremental Backup . . . 5-13
Backing Up a List of Storage Spaces Specified in a File . . . 5-13
How to Avoid Repeatedly Archiving Read-Only Storage Spaces . . . 5-14
Backing Up Specific Tables . . . 5-15
Retrying Skipped Storage Spaces During a Backup . . . 5-15
Performing a Whole-System Backup (IDS) . . . 5-15
Backing Up Smart Large Objects in Sbspaces (IDS) . . . 5-15
Using Fake Backups in a Data Warehouse (IDS) . . . 5-16
Backing Up Blobspaces in a Logging Database (IDS) . . . 5-16
Backing Up Logical Logs When Blobspaces Are Offline (IDS) . . . 5-16
Assigning a Name to a Backup Session (XPS) . . . 5-16
Performing a Physical Backup (XPS) . . . 5-17
Backing Up a Dbslice (XPS) . . . 5-17
Backing Up Table Types . . . 5-17
Viewing Recent ON-Bar Activity (IDS) . . . 5-17
Backing Up Logical Logs . . . 5-18
Backing Up Logical Logs on Dynamic Server . . . 5-18
Performing a Continuous Backup of Logical Logs . . . 5-19
Performing a Manual Backup of Logical Logs . . . 5-19
Using ALARMPROGRAM to Set the Log Backup Mode . . . 5-19
Viewing Backed-Up Logical Logs . . . 5-20
Backing Up Logical Logs on Extended Parallel Server . . . 5-22
Performing a Manual Backup of Logical Logs . . . 5-22
Starting Continuous Logical-Log Backups. . . . 5-23
Preventing Logical-Log Backups in a Test System . . . 5-23

© Copyright IBM Corp. 1996, 2008

Monitoring Logical-Log Backups. 5-23
Salvaging Logical-Log Files 5-24
Understanding ON-Bar Backup Processes. 5-24
Backup Sequence on Dynamic Server 5-24
Backup Sequence on Extended Parallel Server 5-26

In This Chapter

This chapter explains how to use the **onbar** utility to back up and verify storage spaces (dbspaces, blobspaces, and sbspaces) and logical-log files. The **onbar** utility is a wrapper to **onbar_d**, the ON–Bar driver. Use any of the following methods to execute ON–Bar backup and restore commands:

- Issue ON–Bar commands.

  To execute ON–Bar commands, you must be user or a member of the **bargroup** group on UNIX or a member of the **-Admin** group or user in Windows. (For more information, see "Creating the bargroup Group (UNIX)" on page 4-6.)
- Include ON–Bar and ISM commands in a shell or batch script.

  For information, see "Customizing ON-Bar and Storage-Manager Commands" on page 9-2.
- Call ON–Bar from a job-scheduling program.
- Set event alarms that trigger a logical-log backup.

  For information, see "Backing Up Logical Logs on Dynamic Server" on page 5-18 and "Backing Up Logical Logs on Extended Parallel Server" on page 5-22.

Summary of ON-Bar Tasks

You can use ON–Bar to complete a variety of tasks, including the following tasks:

- "Backing Up Storage Spaces and Logical Logs" on page 5-8
- "Viewing Recent ON-Bar Activity (IDS)" on page 5-17
- "Backing Up Logical Logs" on page 5-18
- "Viewing Backed-Up Logical Logs" on page 5-20
- "Verifying Backups with archecker" on page 6-3
- "Performing a Restore" on page 7-13
- "Performing an External Restore" on page 8-13
- "Starting and Stopping ON-Bar Sessions (XPS)" on page 9-10

The following table summarizes frequently-used commands:

Table 5-1. ON-Bar Command Summary

Command	Action
onbar -b	Back up storage spaces
onbar -b -w	Back up the whole system. "Performing a Whole-System Backup (IDS)" on page 5-15
onbar -b -w -L 1	Perform a level-1 (incremental) whole system backup.
onbar -b -w -L 2	Perform a level-2 (incremental) whole system backup.
onbar -b -O	Override error checks during backup
onbar -b -F	Perform a fake backup
onbar -b -l -c	Include the current log in the log backup

Table 5-1. ON-Bar Command Summary (continued)

Command	Action
onbar -l -O	Override error checks to back up logs when blobspaces are offline
onbar -b -l -C	Start a continuous logical-log backup
onbar -b -L 1	Performs a level-1 backup, backing up all changes in the storage spaces since the last level-0 backup. Also performs a level-0 backup of used logical logs. "Performing an Incremental Backup" on page 5-13
onbar -b -f *pathname*	Backs up a list of storage spaces and logical logs that are specified in a file
onbar -b -S	Backs up all operational dbspaces. "How to Avoid Repeatedly Archiving Read-Only Storage Spaces" on page 5-14
onbar -m	Prints 20 lines of recent ON-Bar activity from the activity log file.
onbar -m *num_of_lines*-r *num_of_sec*	Prints a specified number of lines of recent ON-Bar activity at an interval of every specified number of seconds.
onbar -b -l	Performs a backup of full logical-log files.
onbar -b -l -c	Closes and backs up the current logical log and the other full logical logs.
onbar -b -l -C	Starts a continuous log backup of logical logs. "Performing a Continuous Backup of Logical Logs" on page 5-19
onbar -b -l -O	Overrides normal logical backup restrictions such as when a blobspace is offline.
onbar -b -l -s	Salvages any logical logs that are still on disk after a database server failure. "Salvaging Logical-Log Files" on page 15-7
onbar -P	View logical logs that have been backed up using the ON-Bar utility. "Viewing Backed-Up Logical Logs" on page 5-20
onbar -V	Displays the software version number and the serial number of IDS
onbar -v	Use the archecker utility to verify that a backup is usable. "Verification Process with archecker" on page 6-2
onbar -v -f *pathname*	Verify the backed-up storage spaces listed in the file **bkup1**. "Performing Verification Only" on page 6-5
onbar -v -t "*YYYY-MM-DD HH:MM:SS*"	Perform a point-in-time verification. "Performing a Point-in-Time Verification" on page 6-5
onbar -v -w	Verify a whole-system backup. "Verifying a Whole-System Backup" on page 6-5
onbar -version	Displays version information on the build operation system, build number, and build date of IDS.

Preparing for a Backup

This section explains the preliminary steps that you must take before you back up storage spaces and logical logs.

Data ON-Bar Backs Up

ON–Bar backs up the following types of data. ON–Bar backs up the critical dbspaces first, then the remaining storage spaces, and finally the logical logs. (The *critical dbspaces* are the **rootdbs** and the dbspaces that contain the logical logs and physical log.) ON–Bar can back up and restore the largest storage space that your database server supports.

Data Type	Description
Dbspaces that contain tables or indexes	See "Backing Up Storage Spaces and Logical Logs" on page 5-8. ON–Bar also backs up the reserved pages in the root dbspace.
Blobspaces (IDS)	See "Backing Up Blobspaces in a Logging Database (IDS)" on page 5-16.
ISM catalog	If you use ISM, the ISM catalog is in **$DIR/ism** on UNIX and **%ISMDIR%** on Windows.
Logical-log files	See "Backing Up Logical Logs" on page 5-18.
Sbspaces (IDS)	See "Backing Up Smart Large Objects in Sbspaces (IDS)" on page 5-15.

Administrative Files to Back Up

ON–Bar backups safeguard your data. They do not replace normal operating-system backups of important configuration files.

Important: For use in an emergency, you should have a backup copy of the current version of the following administrative files. You will need to restore these files if you need to replace disks or if you restore to a second computer system (imported restore).

The following table lists the administrative files that you should back up.

Administrative Files	IDS	XPS
ONCONFIG file	X	X
Emergency boot files	X	X
sm_versions file	X	X
The **sqlhosts** file (UNIX)	X	X
The **oncfg_*servername.servernum.coserverid*** file from each coserver		X
Storage-manager configuration and data files	X	X
Simple-large-object data in blobspaces that are stored on disks or optical platters	X	
The **xcfg_*servername.servernum*** file in the **etc** subdirectory		X
Externally stored data such as external tables that a DataBlade maintains	X	

Although ON–Bar does not back up the following items, ON–Bar automatically recreates them during a restore. You do not need to make backup copies of these files:

- The *dbspace pages* that are allocated to the database server but that are not yet allocated to a tblspace extent
- Mirror chunks, if the corresponding primary chunks are accessible
- Temporary dbspaces

  ON–Bar does not back up or restore the data in temporary dbspaces. Upon restore, the database server recreates empty temporary dbspaces.

Installing and Configuring a Storage Manager

Before you can create a backup with ON–Bar, you must configure your storage manager and start it. For information about configuring ISM, see the *IBM Informix Storage Manager Administrator's Guide*. For information about configuring third-party storage managers, see Chapter 4, "Configuring the Storage Manager and ON-Bar," on page 4-1, and your storage-manager manuals.

Make sure your storage manager is ready to receive data before you begin a backup or restore. To improve performance, it is recommended that you reserve separate storage devices for storage-space and logical-log backups. If you are backing up to tape or optical disk, label and mount all volumes in the storage device. The backup or restore might pause until you mount the requested tape or disk.

Whole-System Backup (IDS)

A *whole-system* backup (**onbar -b -w**) is a serial backup of all storage spaces and logical logs based on a single checkpoint. That time is stored with the backup information. The advantage of using a whole-system backup is that you can restore the storage spaces with or without the logical logs. Because the data in all storage spaces is consistent in a whole-system backup, you do not need to restore the logical logs to make the data consistent. Level 0, 1, or 2 backups are supported. For an example, see "Performing a Whole-System Backup (IDS)" on page 5-15.

Parallel Backup

A *parallel backup* is a standard backup (**onbar -b**) or whole-system backup (**onbar -b -w**) that runs multiple simultaneous processes, each process backing up a different dbspace. For parallel backup, dbspaces are backed up by number of used pages.

Set the number of simultaneous processes that can be run with the BAR_MAX_BACKUP configuration parameter. In most cases, parallel backups complete faster than serial backups, which use only one process.

To force a serial backup, set BAR_MAX_BACKUP to 1.

Standard Backup

A *standard backup* (**onbar -b**) is a parallel backup of selected or all storage spaces and the logical logs. In a standard backup, the database server performs a checkpoint for each storage space as it is backed up. Therefore, you must restore the logical logs from a standard backup to make the data consistent. For an example, see "Performing a Level-0 Backup of All Storage Spaces" on page 5-13.

Incremental Backup

An *incremental backup* of a storage space backs up only those pages that have been modified since the last backup of the storage space. ON-Bar supports the following types of backups:

- Full backups, also called level-0 backups
- Incremental backups of full backups, also called level-1 backups
- Incremental backups of incremental backups, also called level-2 backups
- Incremental whole system backups

Use the **-L** option to the ON-Bar backup command to select the level of backup. By default ON-Bar performs a level-0 backup.

Note: You cannot perform incremental backups on logical logs.

Physical Backup

A *physical backup* (**onbar -b -p**) backs up just the storage spaces. You can back up specific or all storage spaces.

Example command:

```
onbar -b -p -L 0
```

Using **-p** for a dbspace backup with ON-Bar will only backup the storage spaces, but it will not start an implicit logical log file backup. Instead, a warning message will be written to the ON-Bar activity log file to let the user know, that log file backup was not initiated. The message will also contain the log unique ID of the latest log file that will be required for a restore of the storage spaces. This is the log file that contains the archive checkpoint of the last dbspace backed up.

Example message:

```
2006-12-14 09:30:35 14277  14275 (-43354) WARNING: Logical logs were
     not backed up as part of this operation. Logs through log unique ID 9
     are needed for restoring this backup. Make sure these logs are backed
     up separately.
```

If necessary, then a log switch is initiated, so that this log can be backed up. If the current log is already newer than the log with the archive checkpoint of the last storage space, then no log switch is initiated.

Choosing a Backup Level

ON–Bar supports level-0, level-1, and level-2 backups. It is good practice to create a backup schedule that keeps level-1 and level-2 backups small and to schedule frequent level-0 backups. With such a backup schedule, you avoid having to restore large level-1 and level-2 backups or many logical-log backups.

Level-0 Backups

Level-0 backups can be time-consuming because ON–Bar writes all the disk pages to backup media. Level-1 and level-2 backups might take almost as much time as a level-0 backup because the database server must scan all the data to determine what has changed since the last backup. It takes less time to restore data from level-0, level-1, and level-2 backups than from level-0 backups and a long series of logical-log backups.

Level-1 Backups

A level-1 backup takes less space and might take less time than a level-0 backup because only data that changed since the last level-0 backup is copied to the storage manager.

If you request an incremental backup where no previous incremental backup exists, ON–Bar automatically performs the lower-level backup. For example, if you request a level-1 backup but no level-0 backup exists for one of the storage spaces, ON–Bar automatically performs a level-0 backup of that dbspace and a level-1 backup of the other storage spaces.

Dynamic Server

If you request a whole-system level-1 backup and no level-0 backup exists, ON–Bar performs a whole-system level-0 backup. If you request a whole-system level-2 backup but the level-1 backup does not exist, ON–Bar performs a whole-system level-1 backup.

End of Dynamic Server

Level-2 Backups

A level-2 backup takes less space and might take less time than a level-1 backup because only data that changed since the last level-1 backup is copied to the storage manager.

Collecting Information About Your System Before a Backup

To ensure that you can restore the data, perform the following tasks:

- Print or keep a copy of essential database server configuration information.
- Verify data consistency.
- Keep track of the number of rows in each table (optional).

After you complete the backup, verify it with the **archecker** utility. For more information, see Chapter 6, "Verifying Backups," on page 6-1.

Ensuring That You Have Enough Logical-Log Space

ON–Bar checks for available logical-log space at the beginning of a backup. If the logs are nearly full, ON–Bar backs up and frees the logs before attempting to back up the storage spaces. If the logs contain ample space, ON–Bar backs up the storage spaces, then the logical logs.

Monitor the logs so that you can back them up before they fill. If insufficient space exists in the logical log, the database server will stop responding. If the database server stops responding, add more logical-log files and retry the ON–Bar command.

Extended Parallel Server

Extended Parallel Server keeps one logical log reserved for backups. That way, enough log space exists for you to back up the logical logs to free enough log space to back up storage spaces.

End of Extended Parallel Server

Copying Database Server Configuration Information

Copy the following database server configuration files. For more information, see "Administrative Files to Back Up" on page 5-4.

- The **sqlhosts** file (UNIX only)
- The emergency boot files
- The ONCONFIG file
- **sm_versions** file

Extended Parallel Server

- The **xcfg** file for the database server

End of Extended Parallel Server

Verifying Database Integrity

To ensure the integrity of your backups, periodically verify that all database server data is consistent before you create a level-0 backup. You do not need to check for consistency before every level-0 backup. It is recommended that you do not discard a backup that is known to be consistent until the next time that you verify the consistency of your databases. For information on using the **oncheck** or **onutil** CHECK commands, see the *IBM Informix Administrator's Reference*.

Backing Up Storage Spaces and Logical Logs

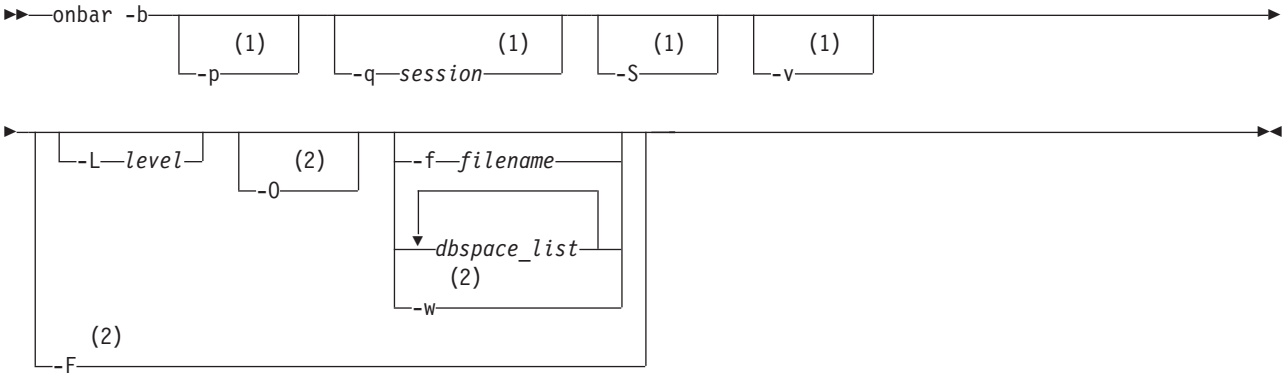
You can back up storage spaces and logical logs only when the database server is in online, quiescent, or fast-recovery mode. However, you can salvage logical logs with the database server offline. The storage-space chunks can be stored on raw disk storage space, in cooked files, or on an NTFS file system (Windows).

Only online storage spaces are backed up. Use the **onstat -d** utility to determine which storage spaces are online. After you begin the backup, monitor its progress in the ON–Bar activity log and database server message log.

Important: You must back up each storage space at least once. ON–Bar cannot restore storage spaces that it has never backed up.

Backup Syntax

Backing Up Storage Spaces and Logical Logs



Notes:

1 Extended Parallel Server Only

2 Dynamic Server Only

Element	Purpose	Key Considerations
-b	Specifies a backup Backs up the storage spaces, logical logs, including the current logical log, and the ISM catalog, if it exists.	You must specify the **-b** parameter first. **Important:** During a backup, if ON–Bar encounters a down dbspace, it skips it and later returns an error.
dbspace_list	Names storage spaces to be backed up On XPS, if you enter a dbslice name, it backs up all the dbspaces in that dbslice.	If you do not enter ***dbspace_list*** *or* **-f *filename***, ON-Bar backs up all online storage spaces on the database server. If you enter more than one storage-space name, use a space to separate the names.
-f *filename*	Backs up the storage spaces that are listed in the text file whose path name *filename* provides Use this option to avoid entering a long list of storage spaces every time that you back up.	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile2**), and absolute (**/usr//backup_lists/listfile3** or **c:\\backup_lists\listfile3**) filenames. For the format of this file, see Figure 5-1 on page 5-13. The file can list multiple storage spaces per line.
-F	Performs a fake backup (IDS)	You can execute this option whether or not a storage-manager application is running. ON–Bar ignores *dbspace_list* if you specify it. Use fake backups to change 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 fake backup. It is recommended that you use fake backups sparingly, if at all. Alternatively, you can use the SQL administration API equivalent: ARCHIVE FAKE. See *IBM Informix Guide to SQL: Syntax* for more information.
-L *level*	Specifies the level of backup to perform on storage spaces: • 0 for a complete backup (The default.) • 1 for changes since the last level-0 backup • 2 for changes since the last level-1 backup	If you request an incremental backup and ON-Bar finds that no previous level backup has been performed for a particular storage space, ON-Bar backs up that storage space at the previous level. For example, if you request a level-1 backup, and ON-Bar finds no level-0 backup, it makes a level-0 backup instead. For more information, see "Performing an Incremental Backup" on page 5-13.
-O	Overrides normal backup restrictions	Use this option to back up logical logs when blobspaces are offline. If a log backup occurs when blobspaces are offline, return code 178 displays in the ON–Bar activity log.
-p	Performs a physical backup (XPS)	Backs up storage spaces and the ISM catalog. (If you omit the **-p** option, ON-Bar also backs up the logical logs.)

Element	Purpose	Key Considerations
-q *session*	Allows you to assign a name to the backup session (XPS)	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 128 characters. This name appears in the output from the **onstat** utility so that you can follow the progress of the backup. For more information, see "Monitoring the Backup Scheduler Status (XPS)" on page 9-11.
-S	Skips backing up read-only storage spaces if archives taken after the storage spaces became read-only exist (XPS)	By default, read-only (static) dbspaces are backed up. Use this option to avoid backing up read-only storage spaces if they have already been backed up since becoming read-only. If you use this option, ON-Bar checks if a storage space is static, and if so, whether the storage space has an existing backup that was performed after the storage space became read-only. If both conditions are satisfied, then ON-Bar does not back up the storage space, even if you explicitly specify that storage space with the **-f** option or by including the storage space name in the backup command. For more information on static storage spaces, see "How to Avoid Repeatedly Archiving Read-Only Storage Spaces" on page 5-14 and the *IBM Informix Administrator's Reference*.
-w	Performs a whole-system backup (IDS)	Backs up all storage spaces, critical dbspaces, and logical logs serially. If you do not save the logical logs, you must use the **-w** option.
-v	Verifies a backup (XPS) If verification is successful, you can restore the storage spaces safely.	Specify **onbar -v** to verify the backup. You cannot verify the logical logs.

Backing Up After Changing the Physical Schema

This section describes what to back up after you change the physical schema.

When to Back Up the Root Dbspace and Modified Storage Spaces

You must perform a level-0 backup of, at minimum, the root dbspace and the modified storage spaces to ensure that you can restore the data after you:

- Add or drop mirroring.
- Move, drop, or resize a logical-log file.
- Change the size or location of the physical log.
- Change your storage-manager configuration.
- Add, move, or drop a dbspace.
- Add, move, or drop a chunk to any type of storage space.

Dynamic Server

- Add, move, or drop a blobspace or sbspace.

End of Dynamic Server

Extended Parallel Server

It is recommended that you perform a complete level-0 backup after changing the physical configuration.

End of Extended Parallel Server

For example, if you add a new dbspace **dbs1**, you will see a warning in the message log that asks you to perform a level-0 backup of the root dbspace and the new dbspace. If you attempt an incremental backup of the root dbspace or the new dbspace instead, ON–Bar automatically performs a level-0 backup of the new dbspace.

Extended Parallel Server

For Extended Parallel Server, back up the root dbspace and the new or modified dbspace on the coserver where the change occurred.

End of Extended Parallel Server

Tip: Although you no longer need to backup immediately after adding a log file, your next backup should be level-0 because the data structures have changed.

Warning: If you create a new storage space with the same name as a deleted storage space, perform a level-0 backup twice:

1. Back up the root dbspace after you drop the storage space and before you create the new storage space with the same name.
2. After you create the new storage space, back up the root dbspace and the new storage space.

When to Back Up the Modified Storage Spaces Only

You must perform a level-0 backup of the modified storage spaces to ensure that you can restore the data when you:

- Convert a nonlogging database to a logging database.
- Convert a raw, static, or operational table to standard. This backup ensures that the unlogged data is restorable before you switch to a logging table type.

Using ISM During a Backup

Use the **ism_watch** command to monitor backups and restores sent to the ISM server. During a backup, the ISM server automatically routes storage-space data to volumes in the ISMData volume pool and logical-log files to volumes in the ISMLogs volume pool.

Always keep the volumes from the ISMLogs pool mounted to ensure that a storage device is always available to accept logical-log data. If the volume is not mounted, the backup will pause. For more information on using devices and ISM commands, see the *IBM Informix Storage Manager Administrator's Guide*.

During the backup operation, ISM creates *save sets* of the backed up data and enters information about the backed up data in the ISM catalog. You can also use this command to back up the ISM catalog:

```
ism_catalog -create_bootstrap
```

If you use the **onbar** script to back up storage spaces and logical logs, it backs up the ISM catalog automatically. If you call **onbar_d** directly, you must use the **ism_catalog -create_bootstrap** command.

Backing up the ISM Catalog

The **ism_catalog -create_bootstrap** command creates a full backup of the ISM catalog when it is first used, but subsequently performs incremental backups. Therefore, the media or volume containing the first bootstrap creation is always necessary in order to restore the bootstrap successfully.

To avoid the need for the first media or volume during bootstrap restore, a full bootstrap backup must be performed every time. To accomplish this, the level must be specified explicitly by using, the underlying **savegrp** command that **ism_catalog -create_bootstrap** calls.

To perform a full backup of the ISM catalog every time, run the following command as the user root:

```
$DIR/bin/savegrp -O -l full ISMData
```

In this command: `-O` is a capital O (not a zero), and `ISMData` is the pool name where the bootstrap data is saved.

You can edit **the $DIR/bin/onbar** shell script to replace the **ism_catalog -create_bootstrap** command with the **savegrp** command.

Redirect the output (both **stdout** and **stderr**) of the **savegrp** command to the ON-Bar activity log file (specified by the BAR_ACT_LOG configuration parameter in onconfig file).

Using IBM Server Administrator to Back Up and Verify

IBMInformix Server Administrator (ISA) is a browser-based tool for performing administrative tasks such as ON–Bar and **onstat** commands. You can use ISA to perform the following ON–Bar tasks:

- View messages in the ON–Bar activity log.
- Perform level-0, level-1, or level-2 backups.
  - Back up storage spaces (**onbar -b**).
  - Back up the whole system (**onbar -b -w**).
  - Override error checks during the backup (**onbar -b -O**).
  - Perform a fake backup (**onbar -b -F**).
- Back up the logical logs.
  - Include the current log in the log backup (**onbar -b -l -c**).
  - Override error checks to back up logs when blobspaces are offline (**onbar -l -O**).
  - Start a continuous logical-log backup (**onbar -b -l -C**).
- Verify backups.
- Edit the **onbar** script.

For more information, see the ISA online help.

ON-Bar Backup Examples

The following sections contain examples of ON–Bar syntax for backing up storage spaces.

Performing a Level-0 Backup of All Storage Spaces

To perform a standard, level-0 backup of all online storage spaces and used logical logs, use one of the following commands:

```
onbar -b
onbar -b -L 0
```

ON–Bar never backs up offline storage spaces, temporary dbspaces, or temporary sbspaces.

Important: Save your logical logs so that you can restore from this backup.

Performing a Level-0 Backup of Specified Storage Spaces

To perform a level-0 backup of specified storage spaces and all logical logs (for example, two dbspaces named fin_dbspace1 and fin_dbspace2), use the -b option as the following example shows. You could also specify the **-L 0** option, but it is not necessary.

```
onbar -b fin_dbspace1 fin_dbspace2
```

Performing an Incremental Backup

An incremental backup backs up all changes in the storage spaces since the last level-0 backup and performs a level-0 backup of used logical logs. To perform a level-1 backup, use the **-L 1** option, as the following example shows:

```
onbar -b -L 1
```

Backing Up a List of Storage Spaces Specified in a File

To back up a list of storage spaces specified in a file and the logical logs, use the following command:

```
onbar -b -f /usr/informix/backup_list/listfile3
```

The format of the file is as follows:

- Each line can list more than one storage space, separated by spaces or a tab.
- Comments begin with a # or ; symbol and continue to the end of the current line.
- ON–Bar ignores all comment or blank lines in the file.
- Specify only spaces (dbspace, blobspace, and so forth) names to ON-Bar in a file or command line. ON-Bar backs up all the chunks that belong to the spaces. Do not specify storage space names with paths.

Figure 5-1 shows a sample file that contains a list of storage spaces to be backed up (**blobsp2.1**, **my_dbspace1**, **blobsp2.2**, **dbsl.1**, **rootdbs.1**, and **dbsl.2**). You can also use this file to specify a list of storage spaces to be restored.

```
blobsp2.1
# a comment                      ignore this text

       my_dbspace1           # back up this dbspace
; another comment
blobsp2.2                       dbsl.1
rootdbs.1      dbsl.2   ; backing up two spaces
```

Figure 5-1. Sample File with a List of Storage Spaces

How to Avoid Repeatedly Archiving Read-Only Storage Spaces

Because the contents of read-only (static) storage spaces do not change, you do not need to archive them as often as other storage spaces. Use the **-S** option with the ON-Bar backup command to avoid repeatedly archiving read-only storage spaces.

The following information assumes that your system contains a mixture of storage spaces that you have designated as read-only using the **onutil** ALTER DBSPACE STATIC command, and dbspaces that you have not designated as read-only (also called operational dbspaces).

By default, the **onbar -b** command backs up all storage spaces in the system, including read-only dbspaces.

The **onbar -b -S** command backs up all operational dbspaces, whether or not an earlier archive of those dbspaces exists. This command also backs up read-only storage spaces that have not been backed up previously. This command does not back up read-only storage spaces for which an unexpired archive that was performed after the storage space became read-only already exists.

The semantics of the **-S** option are not changed by the **-L** option that allows incremental backups. Thus, the command **onbar -b -S -L 1** performs the following tasks:

- For read-only dbspaces, performs a level-0 backup if no backup of the storage space that was taken after the storage space became read-only exists; otherwise skips the storage spaces.
- For operational dbspaces, performs a level-1 archive of the storage space if a level-0 backup of the storage space exists; otherwise performs a level-0 backup of the storage space.

The semantics of the **-S** option also remain unchanged if a read-only storage space is explicitly specified to the ON-Bar backup command, as part of a dbspace list in the command or as part of a dbspace list in a file specified by the **-f** option. For example, if the storage space **dbs_static** has been flagged as read-only and has already been backed up since being flagged read-only, the command **onbar -b -S dbs_static** does not back up the **dbs_static** storage space again.

Although use of the **-S** option can greatly improve backup performance in a system containing large amounts of static data, you should still force backups of read-only dbspaces with some regularity by omitting the **-S** option. Otherwise, you run the risk of having only a single backup of your static data, and risk data loss if a restore is necessary but the single backup becomes unreadable.

If you have a large number of read-only dbspaces, consider backing up a subset of your read-only dbspaces as part of your regular backup cycles. To do so, run the ON-Bar backup command without the **-S** option and specify the subset of read-only dbspaces to back up, either in the command or with the **-f** option. Cycling through different subsets of your read-only dbspaces as part of your regular backup schedule provides you the performance advantages of not backing up all read-only storage spaces in each backup, while minimizing the risk of data loss in the event backup media becomes unreadable.

Backing Up Specific Tables

To back up a specific table or set of tables in ON-Bar, store these tables in a separate dbspace and then back up this dbspace.

Extended Parallel Server

Tip: If your tables are read-only and not modified, consider placing them in read-only (static) dbspaces. Use the **-S** option to the ON-Bar backup command to avoid repeatedly archiving read-only dbspaces.

End of Extended Parallel Server

Warning: If you need to restore only that table, you must warm restore the entire dbspace to the current time (**onbar -r**). This procedure does not allow you to recover from accidentally dropping or corrupting a table because it would be dropped again during logical restore.

Retrying Skipped Storage Spaces During a Backup

You cannot back up storage spaces that are down or temporarily inaccessible. If a storage space is down, ON–Bar skips it during the backup and writes a message to the activity log. Take one of the following actions:

- Retry the backup later when the storage space is back online.
- Restore these storage spaces from an older backup, if available. Make sure that at least one level-0 backup of each storage space exists or else it might not be restorable. For details, see "Restoring from an Older Backup" on page 7-22.

Performing a Whole-System Backup (IDS)

A whole system backup can help you restore your system without a log restore. You can perform an incremental (level 1 or level 2) whole system backup in conjunction with a level 0 whole system backup.

To perform a serial, level-0 backup of all online storage spaces and logical logs, use one of the following commands:

```
onbar -b -w
onbar -b -w -L 0
```

You can run a level 1 whole system backup with the following command:

```
onbar -b -w -L 1
```

For more details, see "Performing a Whole-System Restore (IDS)" on page 7-17.

Backing Up Smart Large Objects in Sbspaces (IDS)

You can back up smart large objects in one or more sbspaces or include them in a whole-system backup. In a level-0 backup, the entire sbspace is backed up. In a level-1 or level-2 backup, the modified sbpages in the sbspace are backed up.

The following example performs a level-0 backup of the **s9sbpace** sbspace:

```
onbar -b -L 0 s9sbspace
```

When you turn on logging for a smart large object, you must immediately perform a level-0 backup to ensure that the object is fully recoverable. For details on logging sbspaces, see the *IBM Informix Administrator's Guide*.

Using Fake Backups in a Data Warehouse (IDS)

The High-Performance Loader (HPL) in Express mode loads tables in read-only mode. A backup changes the table to update mode. Use one of the following commands:

```
onbar -b      # the recommended way
onbar -b -F
```

Warning: It is recommended that you use fake backups on test systems, not on production systems. You cannot restore data from a fake backup. (If you performed a fake backup, you must reload the table to be able to restore the data.)

Backing Up Blobspaces in a Logging Database (IDS)

Follow these steps when you back up blobspaces in a database that uses transaction logging:

1. Before you back up a new blobspace, make sure that the log file that recorded the creation of the blobspace is no longer the current log file.

   To verify the logical-log status, use the **onstat -l** or **xctl onstat -l** command. To switch to the next log file, use the **onmode -l** command.

   Blobspaces are not available for use until the log file is not the current log file. For more information on verifying the logical-log status, see "onstat -l" on page B-3. For information on switching log files, see the **onmode** section in the *IBM Informix Administrator's Reference*.
2. If you update or delete simple large objects in a blobspace, you must back up all the log files, including the current log file. If the blobspace is online, use the **onbar -b -l -c** command.

   When users update or delete simple large objects 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.
3. Back up the blobspaces with the **onbar -b** or **onbar -b -w** command.

Warning: If you perform a warm restore of a blobspace without backing up the logical logs after updating or deleting data in it, that blobspace might not be restorable.

Backing Up Logical Logs When Blobspaces Are Offline (IDS)

To back up the logical logs when a blobspace is offline, use the **onbar -b -l -O** or **onbar -b -O** command. If this backup is successful, ON–Bar returns 178.

To salvage the logical logs, use the **onbar -b -s -O** command.

Warning: If you back up logical logs that contain changes to a blobspace while it is offline, the simple large objects in that blobspace will not be restorable. If an offline blobspace has not changed, it will be restorable.

Assigning a Name to a Backup Session (XPS)

To assign a name to a backup session, use the following command:

```
onbar -b -q session1
```

To monitor the backup session, use the **onstat -g bus** command.

Performing a Physical Backup (XPS)

Use the **-p** option to perform either a level-0 or incremental backup of storage spaces only without also backing up the logical logs, as the following command shows. In the backup command, you can list the storage spaces on the command line or in a file.

```
onbar -b -p
```

To restore data, you must have previously backed up the logical logs. For more information, see "Backing Up Logical Logs on Extended Parallel Server" on page 5-22.

Backing Up a Dbslice (XPS)

You can specify storage spaces individually or with a *dbslice* name. Specifying storage spaces with a dbslice name simplifies backup commands. The dbslice name is translated to the names of its component storage spaces. If you back up a dbslice, ON–Bar backs up all the storage spaces in that dbslice.

To back up all dbspaces in a dbslice named **fin_slice**, use the following command:

```
onbar -b fin_slice
```

Backing Up Table Types

Table 5-2 discusses backup scenarios for different table types. For more information about the table types, see the chapter on data storage in the *IBM Informix Administrator's Guide*.

Table 5-2. Backing Up Table Types

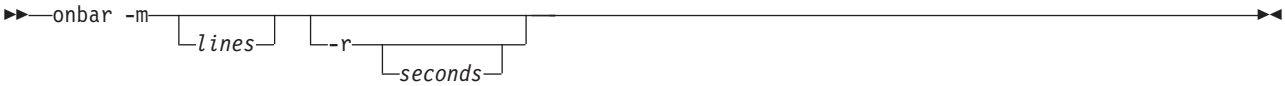
Table Type	IDS	XPS	Can You Back Up This Type of Table?
Standard	X	X	Yes.
Temp	X	X	No.
Scratch		X	No.
Operational		X	Yes. If you use **pload** express mode to load an operational table, you cannot reliably restore the data unless you have done a level-0 backup after the load. It is not enough to just back up the logical logs.
Raw	X	X	Yes. If you update a raw table, you cannot reliably restore the data unless you perform a level-0 backup after the update. Backing up only the logical logs is not enough.
Static		X	Yes.

Important: Perform a level-0 backup before you alter a raw, static, or operational table to type STANDARD.

Viewing Recent ON-Bar Activity (IDS)

You can view recent ON-Bar activity using the **onbar -m** utility. Only users who have permission to perform backup and restore operations can use this option.

Viewing Recent ON-Bar Activity



Element	Purpose	Key Considerations
-m	Prints ON-Bar's recent activity from the activity log file	Default is 20 lines
lines	Specifies the number of lines to output	None
-r	Causes the **onbar -m** utility to repeat	Default is 5 seconds
seconds	Specifies the number of seconds to wait before repeating.	None

Backing Up Logical Logs

For background information, see "Logical-Log Backup" on page 1-2. You can either back up the logical logs separately or with storage spaces. It is recommended that you back up the logical logs as soon as they fill so that you can reuse them and to protect against data loss if the disks that contain the logs are lost. If the log files fill, the database server pauses until you back up the logical logs.

You can either back up the logical logs manually or start a continuous logical-log backup. Logical-log backups are always level 0. After you close the current logical log, you can back it up.

Backing Up Logical Logs on Dynamic Server

If you do not use whole-system backups, you must back up logical logs because you must restore both the storage spaces and logical logs.

If you perform whole-system backups and restores, you can avoid restoring logical logs. It is recommended that you also back up the logical logs when you use whole-system backups. These log backups allow you to recover your data to a time after the whole-system backup, minimizing data loss. The following diagram shows the syntax for **onbar -b -l** commands.

Backing Up Logical Logs

Command	Purpose	Key Considerations
onbar -b -l	Performs a backup of full logical-log files	The current logical-log file is not backed up. If you are using ISM, it also backs up the ISM catalog.
onbar -b -l -c	Closes and backs up the current logical log as well as the other full logical logs	None.
onbar -b -l -C	Starts a continuous log backup	Reserve a dedicated storage device and terminal window because the continuous log backups run indefinitely waiting for logical logs to fill. To stop a continuous log backup, kill the ON–Bar process with an interrupt (^C or SIGTERM).
onbar -b -l -O	Overrides normal logical backup restrictions such as when a blobspace is offline	If a log backup occurs when blobspaces are offline, return code 178 displays in the ON–Bar activity log.
onbar -b -l -s	Salvages any logical logs that are still on disk after a database server failure	If possible, use this option before you replace a damaged disk. If you use **onbar -r** to perform a cold restore on an undamaged disk, ON-Bar automatically salvages the logical logs. For information, see "Salvaging Logical Logs" on page 7-16.

Performing a Continuous Backup of Logical Logs

You can start a continuous logical-log backup in the following ways:

- Specify **onbar -b -l -C**.
- Set the ALARMPROGRAM parameter to the full path for **log_full.sh** on UNIX or **log_full.bat** on Windows.
- Set the ALARMPROGRAM parameter to the full path for **alarmprogram.sh** on UNIX or **alarmprogram.bat** on Windows and set the BACKUPLOGS parameter within the file to Y.
- You can write your own event alarm and set ALARMPROGRAM to it. For more information, see "ALARMPROGRAM" on page 10-5 and the *IBM Informix Administrator's Reference*.

After the continuous logical-log backup starts, it runs indefinitely waiting for logical logs to fill. To stop the continuous logical-log backup, kill the ON–Bar process.

If an error occurs while the continuous logical-log backup is running, it stops. If it stops, reissue the **onbar -b -l -C** command.

Tip: Reserve a dedicated storage device to improve performance during continuous logical-log backups.

Performing a Manual Backup of Logical Logs

To start a manual logical-log backup, use the **onbar -b -l** command. If you set ALARMPROGRAM to **no_log.sh** or **no_log.bat**, you must initiate a logical-log backup manually.

To back up the current logical-log file, use the **onbar -b -l -c** command.

Using ALARMPROGRAM to Set the Log Backup Mode

Use the ALARMPROGRAM configuration parameter to control continuous log backups. If ALARMPROGRAM is set to **log_full.sh** or **log_full.bat**, when a logical-log file fills, the database server triggers event alarm 23. This event alarm

calls **onbar -b -l** to back up the full logical-log file. Restart the database server after you change the value of ALARMPROGRAM.

To turn off continuous log backups, set ALARMPROGRAM to **$INFORMIXDIR/etc/no_log.sh** or **%INFORMIXDIR%\etc\no_log.bat**.

To generate email or pager messages to a designated DBA when a specific error level is triggered, set ALARMPROGRAM to **$INFORMIXDIR/etc/alarmprogram.sh** or **%INFORMIXDIR%\etc\alarmprogram.bat**. Then edit the file to turn automatic logging on, set the level of errors, and insert the email address of the DBA.

Dynamic Server

Additionally, you can set the ALRM_ALL_EVENTS configuration parameter to allow the ALARMPROGRAM script to execute every time any alarm event is triggered.

End of Dynamic Server

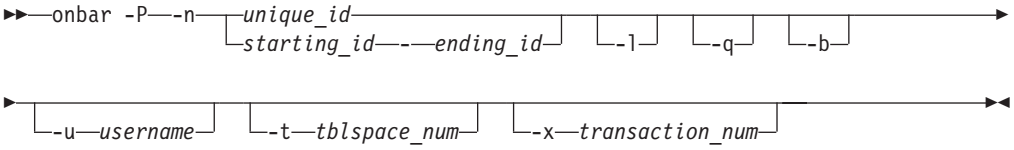
Viewing Backed-Up Logical Logs

You can use the **onbar -P** to view logical logs that have been backed up using the ON-Bar utility. In order to view the backed-up logical logs, the storage manager must be running. This command can be used by anyone who has permission to perform backup and restore operations.

The output of this command is displayed to **stdout**.

You cannot view logs that have not been backed up, which are still on the disk or in shared memory. The contents of these logical logs can only be viewed with the **onlog** utility.

Viewing Backed-Up Logical Logs



Command	Purpose	Key Considerations
-b	Display logical-log records associated with blobspace blobpages	**Additional Information:** The database server stores these records on the logical-log backup media as part of blobspace logging.
ending_id	The ID of the last log to display	Must be a later ID than *starting_id*
-l	Display the long listing of the logical-log record	**Additional Information:** The long listing of a log record includes complex hexadecimal and ASCII dumps of the entire log record. The listing is not intended for casual use.
-n	Display the logical-log records contained in the log file that you specify with *unique_id*	**Additional Information:** The *unique_id* is the unique ID number of the logical log. To determine the *unique_id* of a specific logical-log file, use the **onstat -l** command.
-P	Print backed-up logical log information	This option can only be used to view logical logs that have been backed-up
-q	Do not display program header	None
starting_id	The ID of the first log to display	Must be an earlier ID than *ending_id*
-t *tblspace_num*	Display the records associated with the tblspace that you specify	**Restrictions:** Unsigned integer. The number must be greater than zero and must exist in the **partnum** column of the **systables** system catalog table. **Additional Information:** Specify *tblspace_num* as either an integer or hexadecimal value. If you do not use a prefix of `0x`, the value is interpreted as an integer. To determine the tblspace number of a particular tblspace, query the **systables** system catalog table. For more information, see the *IBM Informix Dynamic Server Administrator's Reference*.
-u *username*	Displays the records for a specific user.	**Restrictions:** The user name must be an existing login name and conform to operating-system-specific rules for login names.
unique_id	ID of the log to display	None
-x *transaction_num*	Display only the records associated with the transaction that you specify	**Restrictions:** The *transaction_num* must be an unsigned integer between zero and TRANSACTIONS -1, inclusive. **Additional Information:** Use the **-x** option only in the unlikely situation of an error being generated during a rollforward. When this situation occurs, the database server sends a message to the message log that includes the transaction ID of the offending transaction. You can use this transaction ID with the **-x** option to investigate the cause of the error.

The following example shows how to use this command with all options:

```
onbar -P -n 2 -l -q -b -u "informix" -t 1048722 -x 1
```

The output for this command might be the following:

```
log uniqid: 2.
1665d0   120  DPT      1        2  0        5
         00000078 0002006c 00000010 0000fefe ...x...l ........
         00000001 00000000 000077e3 00000000 ........ ..w.....
         00000005 00000005 00002a24 00000001 ........ ..*$....
         00100004 0a0c21b8 00002a48 00000001 ......!. ..*H....
         00100006 0a0c2288 00002ea1 00000001 ......". ........
         0010001b 0a0c3810 00002bee 00000001 ......8. ..+.....
```

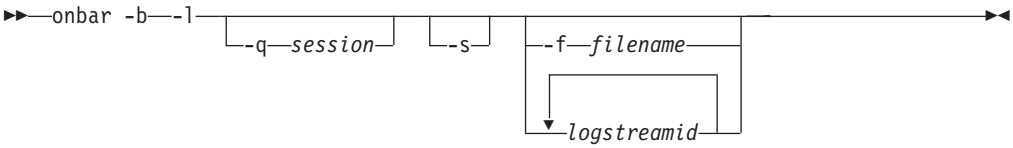
```
        00100015 0a0c3a18 00002a3d 00000001 ......:. ..*=....
        00100005 0a0c57c0                   ......W.
166648  60   CKPOINT  1        0  1665d0   1
        0000003c 00000042 00000010 0000fefe ...<...B ........
        00000001 001665d0 000077e3 00000000 ......e. ..w.....
        00010005 00000002 00000002 001665a0 ........ ......e.
        00000007 ffffffff 00084403          ........ ..D.
```

Backing Up Logical Logs on Extended Parallel Server

You must back up logical logs even when you use nonlogging tables because you must restore both storage spaces and logical logs.

Use **onstat -g bus** or **onstat -g bus_sm** to monitor logical logs and **onbar-worker** processes on each coserver in the current backup session. For more information, see "Monitoring the Backup Scheduler Status (XPS)" on page 9-11.

Backing Up Logical Logs



Command	Purpose	Key Considerations
onbar -b -l	Performs a backup of full logical-log files	The current logical-log file is not backed up. If you are using ISM, it also backs up the ISM catalog.
onbar -b -l -f *filename*	Backs up the logstream IDs that are listed in the text file whose path name *filename* provides Use this option to avoid entering a long list of ***logstreamids*** every time that you use this option.	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile**), and absolute (**/usr/informix/backup_lists/listfile3** or **c:\informix\backup_lists\listfile3**) filenames. The file can list multiple logstreamids per line.
onbar -b -l ***logstreamid***	Uniquely identifies a logical-log stream that a given XPS coserver generates	If you supply more than one ***logstreamid***, separate each item in the list with a space. A logstreamid is the same as a coserver ID.
onbar -b -l -q *session*	Allows you to assign a name to the log backup session This name appears in the **onstat** utility so that you can follow the progress of the log backup.	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 128 characters.
onbar -b -l -s	Salvages any logical logs that are still on disk after a database server failure	If possible, use this option before you replace a damaged disk. If you use **onbar -r** to perform a cold restore on an undamaged disk, ON–Bar automatically salvages the logical logs. For more information, see "Salvaging Logical Logs" on page 7-16.

Performing a Manual Backup of Logical Logs

Use the LOG_BACKUP_MODE configuration parameter to specify whether to back up full logical-log files automatically or manually. If you change the value of LOG_BACKUP_MODE, you must restart the database server before the change takes effect.

If you set LOG_BACKUP_MODE to MANUAL, you must initiate a logical-log backup manually. To back up filled logical-log files manually, use the following command:

```
onbar -b -l
```

If you want to assign a session name to the log backup, use the **-q** option, as follows:

```
onbar -b -l -q "my_logbackup"
```

To back up the current log, use the following commands:

```
xctl onmode -l
onbar -b -l
```

Starting Continuous Logical-Log Backups

On Extended Parallel Server, you can start a continuous logical-log backup by setting the LOG_BACKUP_MODE configuration parameter to CONT. Whenever a logical-log file fills, the Backup Scheduler automatically starts an **onbar-worker** process, if one is not already active, and assigns the log backup to it.

To stop a continuous logical-log backup, you must suspend the backup session. For example, the following command turns off continuous logical-log backup for the session "**log backup 1**" on coserver 1. In a multi-coserver environment, you might need to turn off logical-log backup on each coserver.

```
onbar off -q "log backup 1"
```

To find the logical-log backup session, use the **onstat -g bus** command. For more information, see "Starting and Stopping ON-Bar Sessions (XPS)" on page 9-10.

Preventing Logical-Log Backups in a Test System

If you set LOG_BACKUP_MODE to NONE, you cannot back up or restore logical logs, and log salvage does not work. Although you can continue to back up storage spaces, you cannot restore them. The only reason to set LOG_BACKUP_MODE to NONE is to test your system. Do not use LOG_BACKUP_MODE = NONE in a production system.

Monitoring Logical-Log Backups

To find out if a logical-log file is ready to be backed up, check the flags field of **onstat -l**. After the logical-log file is marked as backed up, it can be reused. When the flags field displays any of the following values, the logical-log file is ready to be backed up:

```
U------
U-----L
```

The value U means that the logical-log file is used. The value L means that the last checkpoint occurred when the indicated logical-log file was current. The value C indicates the current log. If B appears in the third column, the logical-log file is already backed up and can be reused.

```
U-B---L
```

The following example shows the output of **onstat -l** when you use it to monitor logical logs on the database server:

```
Logical Logging
Buffer bufused  bufsize  numrecs  numpages numwrits recs/pages pages/io
  L-2  0        16       1        1        1        1.0        1.0
   Subsystem    numrecs  Log Space used
   OLDRSAM      1        32
address  number   flags    uniqid   begin        size     used    %used
```

```
a038e78  1        U-B----  1        10035f        500      500   100.00
a038e94  2        U-B----  2        100553        500      500   100.00
a038eb0  3        U---C-L  3        100747        500      366    73.20
a038ecc  4        F------  0        10093b        500        0     0.00
a038ee8  5        F------  0        100b2f        500        0     0.00
a038f04  6        F------  0        100d23        500        0     0.00
```

For information about the **onstat -l** fields, see "onstat -l" on page B-3.

Warning: If you turn off continuous logical-log backup, you must monitor your logical logs carefully and start logical-log backups as needed.

Dynamic Server

The flag values `U---C-L` or `U---C--` represent the current logical log. While you are allowed to back up the current logical log, doing so forces a log switch that wastes logical-log space. Wait until a logical-log file fills before you back it up.

End of Dynamic Server

Extended Parallel Server

On Extended Parallel Server, use the **xctl onstat -l** utility to monitor logical logs on all coservers.

End of Extended Parallel Server

Salvaging Logical-Log Files

Use **onbar -b -l -s** to salvage the logs.

ON–Bar salvages logical logs automatically before a cold restore unless you specify a physical restore only. ON–Bar salvages the logical logs that are used before it restores the root dbspace. To make sure that no data is lost before you start the cold restore, manually salvage the logical logs in the following situations:

- If you must replace the media that contains the logical logs

  If the media that contains the logical logs is no longer available, the log salvage will fail, resulting in data loss.
- If you plan to specify a physical restore only (**onbar -r -p**)

For more information, see "Salvaging Logical Logs" on page 7-16 and "Performing a Cold Restore" on page 7-16.

Understanding ON-Bar Backup Processes

This section explains how ON–Bar performs backup operations on the database server. You can perform a backup when the database server is in online or quiescent mode. The original ON–Bar process is called the *driver*, and each new ON–Bar process that it creates is called an **onbar_d** *child* process.

Backup Sequence on Dynamic Server

Figure 5-2 on page 5-26 describes the ON–Bar backup sequence. When you issue a backup command, the **onbar-driver** builds a list of storage spaces and creates a backup session.

In a parallel backup (if BAR_MAX_BACKUP is not set to 1), the **onbar-driver** starts one or more **onbar_d** child processes and assigns backup tasks to them. Each **onbar_d** child process backs up one storage space. Each onbar_d child disappears when the backup of its storage space is done. The **onbar-driver** keeps creating new children until all the storage spaces are backed up. Then the **onbar-driver** backs up the logical logs.

The **onbar_driver** backs up dbspaces by the number of used pages. The dbspace with the most used pages is backed up first; the dbspace with the fewest is backed up last.

If you specify a whole-system backup or set BAR_MAX_BACKUP to 1, the **onbar_driver** backs up the storage spaces and logical logs serially. No **onbar_d** child processes are created.

When the backup is complete, the **onbar-driver** determines whether an error occurred and returns a status in the ON–Bar activity log. After each object is backed up, information about it is added to the emergency boot file on the database server and to the **sysutils** database.

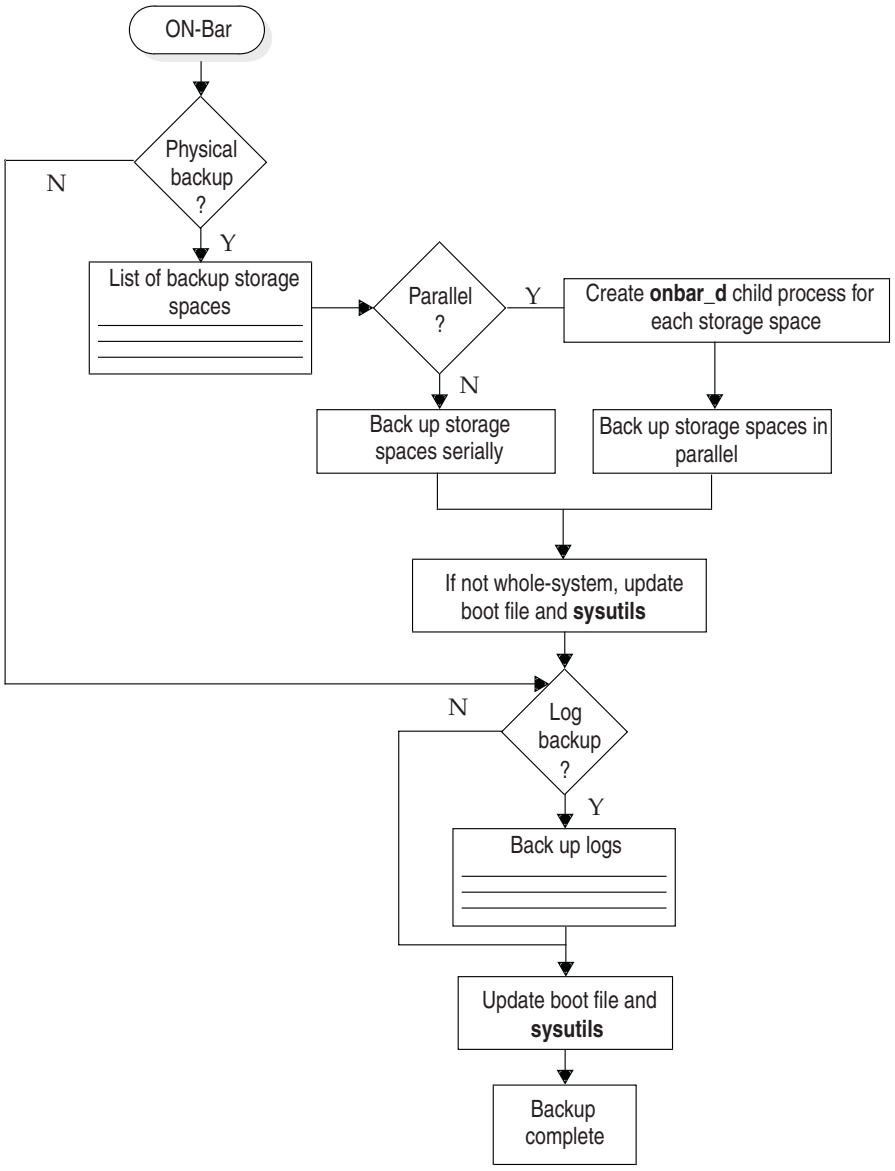


Figure 5-2. ON-Bar Backup Process on Dynamic Server

Backup Sequence on Extended Parallel Server

Figure 5-3 on page 5-27 describes the ON–Bar backup sequence. The **onbar-driver** builds and sends a list of storage spaces to the Backup Scheduler. The Backup Scheduler creates a backup session. It might start one or more **onbar-worker** processes. It then assigns backup tasks to the **onbar-worker** processes.

Once all the storage spaces are backed up, the **onbar-driver** sends a list of logstreams (logical-log data) to the Backup Scheduler that assigns the tasks to **onbar-worker** processes. Each **onbar-worker** process is associated with a coserver and a storage-manager instance. Once an **onbar-worker** process starts, it might be active after the backup or restore session is completed. An **onbar-worker** can process parts of several backup or restore sessions in its lifetime.

Each **onbar-worker** transfers data between the database server and the storage manager until the backup request is fulfilled. When an **onbar-worker** completes its task, it waits for the next task from the Backup Scheduler. If no new task is assigned in a user-specified amount of time, the **onbar-worker** shuts down. You can set the number of minutes that the **onbar-worker** processes wait in BAR_IDLE_TIMEOUT in the ONCONFIG file.

If the Backup Scheduler has new tasks to assign and not enough **onbar-worker** processes are running to complete the task, it calls the **start_worker** script to start one or more new **onbar-worker** processes.

It is recommended that you start **onbar-worker** processes automatically, but if you want to start them manually, see "Using start_worker.sh to Start onbar_worker Processes Manually (XPS)" on page 9-4. If you have set BAR_WORKER_MAX = `0`, you must start a new **onbar-worker** manually.

After each object is backed up, ON–Bar updates the emergency backup boot file on the coserver that is local to the **onbar-worker** and the **sysutils** database. The emergency backup boot file is on the coserver of the **onbar-worker** that backed it up.

To monitor the status of backups, restores, and **onbar-worker** activities, use the **onstat -g bus** or **onstat -g bus_sm** options or check the Backup Scheduler tables in the **sysmaster** database. For more information, see "Monitoring the Backup Scheduler Status (XPS)" on page 9-11 and "Backup Scheduler SMI Tables (XPS)" on page 11-8.

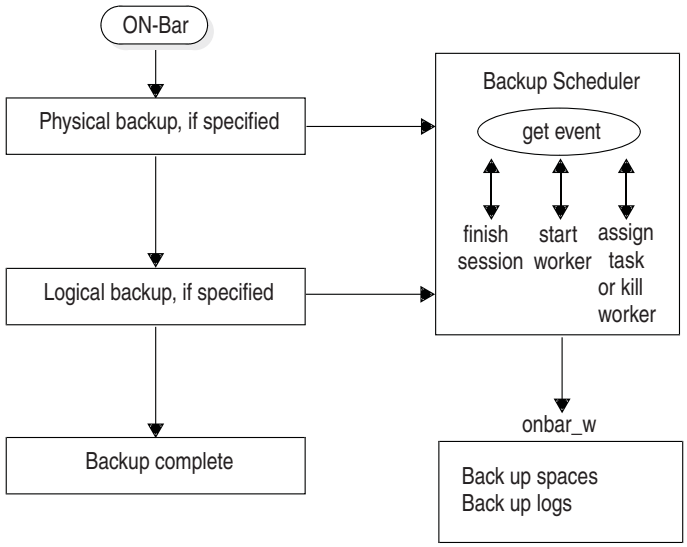


Figure 5-3. ON-Bar Backup Process on Extended Parallel Server

Chapter 6. Verifying Backups

In This Chapter 6-1
archecker Overview 6-1
Verification Process with archecker 6-2
Verifying Backups with archecker 6-3
Estimating the Amount of Temporary Space for archecker 6-4
Verifying Backups 6-5
Performing Verification Only 6-5
Performing a Point-in-Time Verification 6-5
Verifying a Whole-System Backup 6-5
Verifying Blobspaces (IDS) 6-5
Verifying Sbspaces (IDS) 6-6
Interpreting Verification Messages 6-6
Sample Verification Message in the ON-Bar Activity Log 6-6
Sample Verification Message in the archecker Message Log 6-6
Verification Failures 6-6
Backups with Corrupt Pages 6-6
Backups with Corrupt Control Information 6-7
Backups with Missing Data 6-7
Backups of Inconsistent Database Server Data 6-7
Fixing Backup Verification Problems 6-7

In This Chapter

This chapter describes the **archecker** utility for checking the validity and completeness of backups. To ensure that you can restore a backup safely, issue the **onbar -v** command, which calls the **archecker** utility.

archecker Overview

You access the **archecker** utility when you use the **onbar -v** command. You can use **archecker** with the database server in any mode. The **archecker** utility verifies that all pages required to restore a backup exist on the media in the correct form. After you successfully verify a backup, you can restore it safely. If **archecker** shows problems with the backup, contact Technical Support.

The **archecker** utility verifies standard and whole-system backups. The **archecker** utility cannot verify logical-log backups.

Figure 6-1 shows how ON–Bar and **archecker** verify a backup. The **archecker** utility verifies level-0 backups on all database servers. The following steps correspond to the circled numbers in Figure 6-1.

© Copyright IBM Corp. 1996, 2008

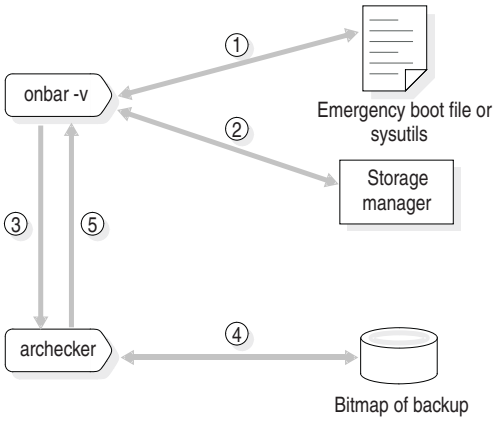


Figure 6-1. How ON-Bar Verifies a Backup

Verification Process with archecker

When the user issues an **onbar -v** command, the following sequence of actions occurs:

1. ON–Bar uses the emergency boot file if the database server is offline or in microkernel mode or the **sysutils** database if the database server is online or quiescent to determine which backup to verify.
2. ON–Bar requests and retrieves the backup data from the storage manager.
3. ON–Bar forwards the backup data to **archecker**.
4. The **archecker** utility scans the backup data and creates a bitmap of the pages. During the scan phase, **archecker** verifies the following types of problems:
   - Backups with corrupt pages
   - Backups with corrupt control information
   - Backups with missing pages that have been added since the last level-0 backup
   - Retrieval of the wrong backup objects

     An example of retrieving the wrong backup object is if ON–Bar requests the **rootdbs** backup from last Wednesday but the storage manager retrieves the **rootdbs** backup from last Tuesday.
5. After it completes the scan, **archecker** uses this bitmap to verify the backup and records the status in the **archecker** message log. ON–Bar also records this status in the ON–Bar activity log.
6. When a backup is verified, ON–Bar inserts a row into the emergency boot file with the backup copy ID and the verification date, and updates the **ins_verify** and **ins_verify_date** rows of the **bar_instance** table in the **sysutils** database. For more information, see “bar_instance” on page 11-2.

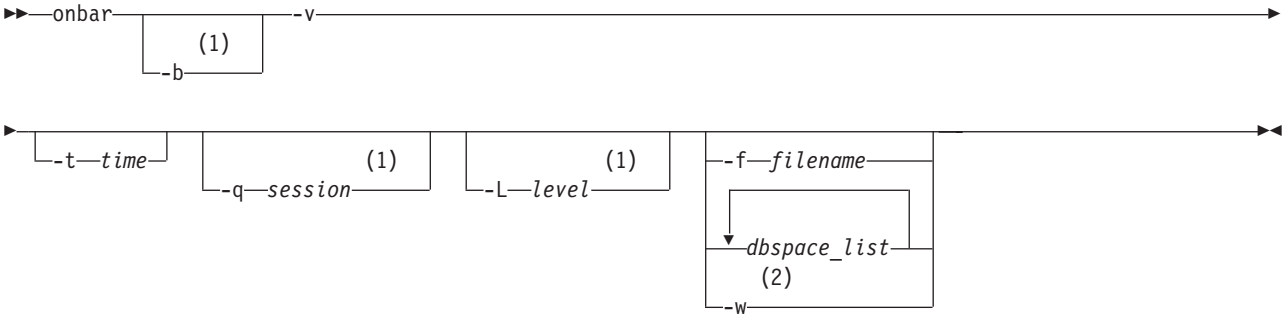
During the verification phase, **archecker** verifies that all the pages for each table are present and checks the partition pages, the reserved pages, the chunk-free list, blobspaces, sbspaces, and extents. The **archecker** utility also checks the free and used counts, verifies that the page stamps match and that no overlap exists in the extents.

Archecker writes temporary files in the directory that the AC_STORAGE parameter specifies. For information, see "AC_STORAGE" on page 10-3.

Verifying Backups with archecker

This diagram shows the **onbar -v** syntax.

Verifying Backups with archecker



Notes:

1 Extended Parallel Server Only

2 Dynamic Server Only

Element	Purpose	Key Considerations
onbar -v	Verifies a backup If verification is successful, you can restore the storage spaces safely.	Specify **onbar -v** to verify the backup. You can perform a point-in-time verification. You cannot verify the logical logs. You must specify the **-b** or **-v** parameter first. You can verify a whole-system or physical-only backup (IDS).
-b	Optionally back ups the data before verifying the backup (XPS only).	onbar -b -v performs a level-0 backup, and then immediately verifies the backup. The following command performs a level-2 backup of the dbspaces alpha, beta, and gamma, and then immediately verifies the level-0, level-1, and level-2 backups of those storage spaces: `onbar -b -v -L 2 alpha beta gamma`
dbspace_list	Names a list of storage spaces to be backed up or verified	If you enter more than one storage-space name, use a space to separate the names. On XPS, if you enter a dbslice name, it verifies all the dbspaces in that dbslice.
-f *filename*	Verifies the storage spaces that are listed in the text file whose path name *filename* provides Use this option to avoid entering a long list of storage spaces every time that you verify them.	You can use any valid UNIX or Windows path name and file name. For the format of this file, see Figure 5-1 on page 5-13. The file can list multiple storage spaces per line.

Element	Purpose	Key Considerations
-L *level*	Specifies the level of backup to verify (XPS): • 0 for a complete backup • 1 for changes since the last level-0 backup • 2 for changes since the last level-1 backup	The **archecker** utility fully verifies level-0 backups on all database servers and performs limited verification of level-1 and level-2 backups. If you do not specify a level, archecker verifies the level-0 backup.
-q *session*	Assigns a name to the verify session (XPS)	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 126 characters. This name appears in the **onstat** utility so that you can follow the progress of the verify session.
-t *time*	Specifies the date and time to which dbspaces are verified	How you enter the time depends on your current GLS locale convention. If the **GL_DATETIME** environment variable is set, you must specify the date and time according to that variable. If the GLS locale is not set, use ANSI-style date format: YYYY-MM-DD HH:MM:SS.
-w	Verifies a whole-system backup	For IDS only.

Estimating the Amount of Temporary Space for archecker

The **archecker** utility requires about 15 megabytes of temporary space for a medium-size system (40-50 gigabytes) and 25 megabytes for a large system. This temporary space is stored on the file system in the directory that the AC_STORAGE parameter specifies, not in the dbspaces. The temporary files contain bitmap information about the backup and copies of partition pages, free pages in a chunk, reserved pages, and optionally, free pages in a blobspace and debugging information. The **archecker** utility must have permissions to the temporary directory.

If the backup is verified successfully, these files are deleted. If the backup fails verification, these files remain. Copy them to another location so that Technical Support can review them.

If your database server contains only dbspaces, use the following formula to estimate the amount of temporary space in kilobytes for the **archecker** temporary files:

```
space = (130 KB * number_of_chunks) + (pagesize * number_of_tables) +
(.05 KB * number_of_logs)
```

Dynamic Server

If your database server contains blobspaces or sbspaces, use the following formula to estimate the amount of temporary space for the **archecker** temporary files:

```
space = (130 KB * number_of_chunks) + (pagesize * number_of_tables) +
(.05 KB * number_of_logs) + (pagesize * (num_of_blobpages/252))
```

End of Dynamic Server

number_of_chunks
The maximum number of chunks that you estimate for the database server.

pagesize The system page size in kilobytes.

number_of_tables
The maximum number of tables that you estimate for the database server.

number_of_logs The number of logical logs on the database server.

num_of_blobpages
The number of blobpages in the blobspaces or the number of sbspaces. (If your database server contains sbspaces, substitute *num_of_blobpages* with the number of sbspaces.) (IDS only)

For example, you would need 12.9 megabytes of temporary disk space on a 50-gigabyte system with a page size of 2 kilobytes. This system does not contain any blobspaces, as the following statement shows:

```
13,252 KB = (130 KB * 25 chunks) + (2 KB * 5000 tables) +
            (.05 KB * 50 logs) + (2 KB * 0)
```

To convert kilobytes to megabytes, divide the result by 1024:

```
12.9 MB = 13,252/1024
```

Verifying Backups

The following examples show how to verify an existing backup and how to verify immediately after backing up.

Performing Verification Only

To verify a backup of all storage spaces, use the **onbar -v** command. The logical logs are not verified.

To verify the backed-up storage spaces listed in the file **bkup1**, use the following command:

```
onbar -v -f /usr/backups/bkup1
```

Performing a Point-in-Time Verification

To perform a point-in-time verification of a backup, use the following command with the *datetime* value in quotes:

```
onbar -v -t "2001-12-10 10:20:50"
```

Verifying a Whole-System Backup

To verify a whole-system backup, use the following command:

For Dynamic Server:

```
onbar -v -w
```

During a verification with archecker, **-w** specifies to verify a whole system backup

Verifying Blobspaces (IDS)

The **onbar -v** command cannot verify the links between data rows and simple large objects in a blobspace. Use the **oncheck -cD** command instead to verify the links in a blobspace. For information on **oncheck**, see the *IBM Informix Administrator's Reference*.

Verifying Sbspaces (IDS)

The **onbar -v** command verifies only the smart-large-object extents in an sbspace. For a complete check, use the **oncheck -cS** command. For information on **oncheck**, see the *IBM Informix Administrator's Reference*.

Interpreting Verification Messages

When you verify a backup, ON–Bar writes summary messages to the **bar_act.log** that report which storage spaces were verified and whether the verification succeeded or failed. The **archecker** utility writes detailed messages to the **ac_msg.log**. Technical Support uses the **ac_msg.log** to diagnose problems with backups and restores.

Sample Verification Message in the ON-Bar Activity Log

The level-0 backup of dbspace **dbs2.2** passed verification, as follows:

```
Begin backup verification of level0 for dbs2.2 (Storage Manager Copy ID:##)
Completed level-0 backup verification successfully.
```

The level-0 backup of **rootdbs** failed verification, as follows:

```
Begin backup verification of level0 for rootdbs (Storage Manager Copy ID:##).
ERROR: Unable to close the physical check: error_message.
```

Sample Verification Message in the archecker Message Log

More detailed information is available in the **archecker** message log, as follows:

```
STATUS: Scan PASSED
STATUS: Control page checks PASSED
STATUS: Starting checks of dbspace dbs2.2.
STATUS: Checking dbs2.2:TBLSpace
.
.
STATUS: Tables/Fragments Validated: 1
Archive Validation Passed
```

Verification Failures

If a backup fails verification, do not attempt to restore it. The results are unpredictable and range from corruption of the database server to a failed restore because ON–Bar cannot find the backup object on the storage manager. In fact, the restore might appear to be successful but it hides the real problem with the data or media.

The three different types of corrupt backups are as follows:

- Backups with corrupt pages
- Backups with corrupt control information
- Backups with missing data

Backups with Corrupt Pages

If the pages are corrupt, the problem is with the databases rather than with the backup or the media.

For Dynamic Server, run **oncheck -cd** on any tables that produce errors and then redo the backup and verification. To check extents and reserved pages, run **oncheck -ce** and **oncheck -cr**.

For Extended Parallel Server, run **onutil CHECK DATA** on any tables that produce errors and then redo the backup and verification. To check extents and reserved pages, run **onutil CHECK SPACE** and **onutil CHECK RESERVED**.

Backups with Corrupt Control Information

In this case, all the data is correct, but some of the backup control information is incorrect, which could cause problems with the restore. Ask Technical Support for assistance.

Backups with Missing Data

When a backup is missing data, it might not be recoverable. After a data loss, try to restore from an older backup. Then restore the current logical logs.

Backups of Inconsistent Database Server Data

There are cases where **archecker** returns "success" to ON–Bar but shows "failure" in the **archecker** message logs. This situation occurs when **archecker** verifies that ON–Bar backed up the data correctly, but the database server data was invalid or inconsistent when it was backed up.

Fixing Backup Verification Problems

Follow these steps when a backup fails verification. The first procedure diagnoses why a backup failed verification; the second procedure verifies an expired backup; and the third procedure verifies a backup with missing data.

To diagnose why a backup failed verification:

1. Verify that the **AC_CONFIG** environment variable and the contents of the **archecker** configuration file are set correctly. If these variables are set incorrectly, the ON–Bar activity log displays a message.
2. Immediately redo the backup onto different media.

   Do *not* reuse the original backup media because it might be bad.
3. Do not use any backups based on this backup. If the level-0 backup is bad, do not use the corresponding level-1 and level-2 backups.
4. Verify this new backup. If verification succeeds, you will be able to restore the storage spaces with confidence.
5. Use your storage manager to expire the backup that failed verification and then run the **onsmsync** utility without arguments to remove the bad backup from the **sysutils** and emergency boot files.

   For more information on expiring data from the storage manager, see your storage-manager documentation or the *IBM Informix Storage Manager Administrator's Guide*. For more information, see "Using the onsmsync Utility" on page 9-6.
6. If verification fails again, call Technical Support and provide them with the following information:
   - Your backup tool name (ON–Bar)
   - The database server **online.log**
   - The archecker message log
   - The AC_STORAGE directory that contains the bitmap of the backup and copies of important backed-up pages

   If only part of the backup is corrupt, Technical Support can help you determine which portion of the backup can be restored in an emergency.
7. Technical Support might advise you to run **oncheck** options against a set of tables. (See "Backups with Corrupt Pages" on page 6-6.)

To verify an expired backup:

1. Check the status of the backup save set on the storage manager. If the storage manager has expired the backup save set, the **archecker** utility cannot verify it.

2. Use the storage-manager commands for activating the expired backup save set. See your storage-manager documentation or the *IBM Informix Storage Manager Administrator's Guide*.
3. Retry the backup verification: **onbar -v**.

To restore when a backup is missing data:

1. Choose the date and time of an older backup than the one that just failed. To perform a point-in-time verification, use the following command:

   ```
   onbar -v -t datetime dbspace1
   ```

2. If the older backup passes verification, perform a point-in-time physical restore using the same datetime value, then perform a log restore, as follows:

   ```
   onbar -r -p -t datetime dbspace1
   onbar -r -l
   ```

3. To prevent ON-Bar from using a backup that failed verification as part of a restore, expire the bad backup at your storage manager and then run the **onsmsync** utility without arguments. The **onsmsync** utility removes backups that are no longer held by the storage manager from the emergency boot file and the **sysutils** database, thereby preventing ON-Bar from attempting to use such backups.

Chapter 7. Restoring Data with ON-Bar

In This Chapter . . . 7-2
ON-Bar Restore Types . . . 7-2
Warm Restore . . . 7-2
Cold Restore . . . 7-2
Mixed Restore . . . 7-3
Parallel Restore . . . 7-4
Whole-System Restore (IDS) . . . 7-4
Point-in-Time Restore . . . 7-4
Imported Restore . . . 7-4
Rename Chunks Restore (IDS) . . . 7-5
Restartable Restore (IDS) . . . 7-5
Continuous Log Restore . . . 7-5
Pre-Recovery Checklist . . . 7-6
Monitoring Restores . . . 7-7
Ensuring That Storage Devices Are Available . . . 7-7
Restoring Save Sets with ISM . . . 7-8
Performing a Complete Restore . . . 7-8
Performing a Physical-Only or Logical-Only Restore . . . 7-10
Using IBM Informix Server Administrator to Restore Data . . . 7-13
Examples of ON-Bar Restore Commands . . . 7-13
Performing a Restore . . . 7-13
Restoring Specific Storage Spaces . . . 7-14
Performing a Logical Restore . . . 7-14
Skipping Logical Replay (XPS) . . . 7-14
Restoring Data on Extended Parallel Server . . . 7-15
Performing a Physical Restore Followed By a Logical Restore . . . 7-15
Salvaging Logical Logs . . . 7-16
Performing a Cold Restore . . . 7-16
Performing a Whole-System Restore (IDS) . . . 7-17
Restoring Data Using a Mixed Restore . . . 7-18
Restoring Data to a Point in Time . . . 7-20
Restoring from an Older Backup . . . 7-22
Performing a Point-in-Log Restore (IDS) . . . 7-22
Restoring Online Storage Spaces . . . 7-22
Recreating Chunk Files During a Restore . . . 7-22
Restoring a Dropped Storage Space . . . 7-23
Deferring Index Rebuild After Logical Restore (XPS) . . . 7-24
Restoring Data when Reinitializing the Database Server . . . 7-24
Configuring Continuous Log Restore using ON-Bar . . . 7-25
Renaming Chunks During a Restore (IDS) . . . 7-26
Key Considerations . . . 7-26
New-Chunk Requirements . . . 7-26
Syntax . . . 7-27
Examples of Renaming Chunks During a Restore . . . 7-28
Renaming Chunks with Command Line Options . . . 7-28
Renaming Chunks with a File . . . 7-28
Renaming Chunks While Specifying Other Options . . . 7-28
Renaming a Chunk to a Nonexistent Device . . . 7-28
Transferring Data with the Imported Restore . . . 7-29
Importing a Restore . . . 7-30
Initializing High-Availability Data Replication with ON-Bar (IDS) . . . 7-32
Restoring Nonlogging Databases and Tables . . . 7-33
Restoring Table Types . . . 7-33
Restoring Tables with Large Objects . . . 7-34
Using Restartable Restore to Recover Data (IDS) . . . 7-35

© Copyright IBM Corp. 1996, 2008

Restartable Restore Example 7-35
Restarting a Restore 7-35
Interaction Between Restartable Restore and BAR_RETRY Value 7-36
Restarting a Logical Restore 7-37
Resolving a Failed Restore 7-38
Understanding ON-Bar Restore Processes 7-40
Warm-Restore Sequence on Dynamic Server 7-40
Cold-Restore Sequence on Dynamic Server 7-41
Warm-Restore Sequence on Extended Parallel Server 7-42
Cold-Restore Sequence on Extended Parallel Server 7-43

In This Chapter

This chapter describes the different types of restores that ON–Bar performs.

ON-Bar Restore Types

You can restore storage spaces stored in both raw and cooked files. If your system contains primary and mirror storage spaces, ON–Bar writes to both the primary and mirror chunks at the same time during the restore, except for an external restore. Mirroring is a strategy that pairs a *primary* chunk of one storage space with an equal-sized *mirror chunk*.

Important: You cannot specify temporary dbspaces in a warm or cold restore. When you restore the critical dbspaces (for example, the root dbspace), the database server recreates the temporary dbspaces, but they are empty.

Warm Restore

The database server is in online, quiescent, or fast-recovery mode in a warm restore. Unless your database server has failed, you can restore noncritical storage spaces in a warm restore in the following circumstances:

- The storage space is online, but one of its chunks is offline, recovering, or inconsistent.
- The storage space is offline or down.

If the database server goes offline but the critical dbspaces are all right, bring the database server online and perform a warm restore. If the database server goes offline and a critical dbspace is down, perform a cold restore. For details, see "Cold Restore" on page 7-2.

Dynamic Server

A warm restore can be performed after a dbspace has been renamed and a level 0 archive of the rootdbs and renamed dbspace is taken.

End of Dynamic Server

Note: Warm restores are not supported for Enterprise Replication servers.

Cold Restore

If a critical dbspace is damaged because of a disk failure or corrupted data, the database server goes offline automatically. If a critical dbspace goes down, you must perform a *cold restore* of all critical dbspaces.

The database server must be offline on Dynamic Server or in microkernel mode on Extended Parallel Server for a cold restore. You can perform a cold restore of all storage spaces regardless of whether they were online or offline when the database server went down.

Perform a cold restore when the database server fails or you need to perform one of the following tasks:

- Point in time restore
- Point in log restore
- Whole system restore (IDS only)
- Imported restore
- Renaming chunks (IDS only)

A cold restore starts by physically restoring all critical storage spaces, then the noncritical storage spaces, and finally the logical logs. The database server goes into recovery mode after the reserved pages of the root dbspace are restored. When the logical restore is complete, the database server goes into quiescent mode. Use the **onmode** command to bring the database server online. For more information, see "Performing a Cold Restore" on page 7-16.

Tip: If you mirror the critical dbspaces, you are less likely to have to perform a cold restore after a disk failure because the database server can use the mirrored storage space. If you mirror the logical-log spaces, you are more likely to be able to salvage logical-log data if one or more disks fail.

Dynamic Server

A cold restore can be performed after a dbspace has been renamed and a level-0 backup of the rootdbs and renamed dbspace is performed.

End of Dynamic Server

Note: Cold restores are required for Enterprise Replication servers before resuming replication, warm restores are not supported.

Mixed Restore

A *mixed restore* is a cold restore of an initial group of storage spaces followed by one or more warm restores of the remaining storage spaces. The initial set of storage spaces you restore in the cold restore must include all critical storage spaces in the server. To the extent that you do not restore all storage spaces during the initial cold restore and avoid the time necessary to restore them, you can bring the server online faster than if you were to perform a cold restore of the entire server. You can then restore the remaining storage spaces in one or more warm restores.

The storage spaces that you do not restore during the cold restore are not available until after you restore them during a warm restore, although they might not have been damaged by the failure. While a mixed restore makes the critical data available sooner, the complete restore takes longer because the logical logs are restored and replayed several times, once for the initial cold restore and once for each subsequent warm restore.

Parallel Restore

If you perform a restore using the **onbar -r** command, ON–Bar restores the storage spaces in parallel and replays the logical logs once. dbspaces are restored by number of used pages.

Dynamic Server

If BAR_MAX_BACKUP is set to 1, ON–Bar restores the storage spaces serially. If you did not perform a whole-system backup, you must use the **onbar -r** command to restore the data.

End of Dynamic Server

Extended Parallel Server

If BAR_WORKER_MAX is set to 1, ON–Bar restores the storage spaces serially.

End of Extended Parallel Server

Whole-System Restore (IDS)

Whole-system restores are always serial. If you backed up the whole system (**onbar -b -w**), you can restore the whole system (**onbar -r -w** or **onbar -r -p -w**) with or without restoring the logical logs. Choose a whole-system restore for small systems or when you do not need to restore logs. You can perform a whole-system point-in-time restore.

Point-in-Time Restore

A point-in-time restore is a cold restore that you can use to undo mistakes that might otherwise not be fixable. An example of such a mistake is dropping a table by mistake. A full restore restores the table during the physical restore but drops it again during the logical restore. A point-in-time restore lets you restore the data to the moment just before the table was dropped.

When you restore the database server to a specific time, any transactions that were uncommitted at the specified point in time are lost. Also, all transactions after the point-in-time restore are lost. For information on how to restore to a specific time, see "Restoring Data to a Point in Time" on page 7-20.

Imported Restore

ON–Bar allows you to restore objects to a different database server instance than the one from which the data was backed up. This type of restore is called an *imported restore*.

You can perform imported restores using whole-system, serial, or parallel backups. You must use compatible versions of XBSA and storage managers for both operations. If you perform a parallel imported restore, it must include all storage spaces, logical logs, and administrative files from the source database server to synchronize the instance. For more information, see "Transferring Data with the Imported Restore" on page 7-29.

You cannot use a backup from one database server version to restore on a different version.

Rename Chunks Restore (IDS)

ON–Bar allows you to rename chunks by specifying new chunks paths and offsets during a cold restore. This option is useful if you need to restore storage spaces to a different disk from the one on which the backup was made. You can rename any type of chunk, including critical chunks and mirror chunks.

The ON-Bar rename chunk restore only works for cold restores. The critical dbspaces (the rootdbs and any dbspace containing logical or physical logs) must be restored during a cold restore. If you do not specify the list of dbspaces to be restored, then the server will restore the critical dbspaces and all the other dbspaces. But if you specify the list of dbspaces to be restored, then the critical dbspaces must be included in the list.

For more information, see "Renaming Chunks During a Restore (IDS)" on page 7-26.

You can rename chunks during an external cold restore. See "Renaming Chunks" on page 8-11 for more information.

Restartable Restore (IDS)

If a failure occurs with the database server, media, or ON–Bar during a restore, you can restart the restore from the place that it failed. You do not have to restart the restore from the beginning. The RESTARTABLE_RESTORE parameter controls whether ON–Bar is able to keep track of the storage spaces and logical logs that were restored successfully.

You can restart the following types of restores:

- Whole system
- Point in time
- Storage spaces
- Logical part of a cold restore

For more details, see "Using Restartable Restore to Recover Data (IDS)" on page 7-35 and "RESTARTABLE_RESTORE (IDS)" on page 10-21.

Continuous Log Restore

Normal log restore restores all of the available log file backups and applies the log records. After the last available log is restored and applied, the log restore finishes. Transactions that are still open are rolled back in the transaction cleanup phase, then the server is brought into quiescent mode. After the server is quiesced, no more logical logs can be restored.

With continuous log restore, instead of transaction cleanup the server is put into log restore suspended state after the last available log is restored. The restore client (**ontape** or ON-Bar) exits and returns control to you. With the server in this state, you can start another logical restore after additional logical logs become available. As long as you start each log restore as a continuous log restore, you can continue this cycle indefinitely.

One use of continuous log restore is to keep a second system available in case the primary system fails. You can restore logical logs backed up on the primary system on the secondary system as they become available. If the primary system fails, you

can restore remaining available logical logs on the secondary system and bring that secondary system online as the new primary system.

Continuous log restore requires much less network bandwidth than High-Availability Data Replication (HDR) and enterprise data replication (ER). Continuous log restore is more flexible than HDR and ER because you can start continuous log restore at any time. As a result, continuous log restore is more robust than HDR or ER in unpredictable circumstances, such as intermittent network availability.

For more information, see "Configuring Continuous Log Restore using ON-Bar" on page 7-25 and "Configuring Continuous Log Restore Using ontape" on page 15-10.

Pre-Recovery Checklist

Use this checklist to help you decide if a restore is required.

- Determine if you need to restore. If one or more of these problems is true, you need to do a recover to fix the problem:
  - Has data been lost or corrupted?
  - Does a committed transaction error need to be undone?
  - Is the database server down or has a disk failed?
  - Is a storage space or chunk down or inconsistent?
- Review the following files and outputs to obtain information about your system and to determine the problem:
  - The **onstat -d** and **onstat -l** outputs
  - The database server message log
  - The ON–Bar activity log
  - The storage-manager logs
  - The **oncheck** output (Dynamic Server) or **onutil** CHECK output (Extended Parallel Server)
  - The **oncfg** files
  - The physical data layout (disks)
  - The database schema (**dbschema** command)
  - The **af*** files (assertion failures), if any
  - The core dump files, if any

    Extended Parallel Server

  - The **xcfg** file

    End of Extended Parallel Server

  - The **ism_chk.pl** report

    The **ism_chk.pl** report is useful when you investigate backup or restore problems. For details, see the *IBM Informix Storage Manager Administrator's Guide*.
- Estimate how long the restore will take.
- Determine whether a warm or cold restore is needed. See "Warm, Cold, and Mixed Restores" on page 1-4 for more information.
- If you need to take the database server offline for the restore, ask your client users to log off the system.

- If you suspect a problem with the storage manager or the XBSA connection, the operating system, or the storage media, contact your vendor to resolve the problem before doing a restore.

Monitoring Restores

To determine the state of each storage space and its chunks, or the status of the restore, examine the output of the **onstat -d** utility. The **onstat -d** utility works only with the database server online. For more information on **onstat -d**, see "onstat -d" on page B-1.

The following table describes the information in the second position of the **flags** column in the first (storage spaces) and second (chunks) sections of the **onstat -d** output and the actions required to solve the problems.

Extended Parallel Server

On Extended Parallel Server, you can use the **xctl onstat -d** utility to check the storage spaces on all coservers.

End of Extended Parallel Server

onstat -d Flag	Storage Space or Chunk State	Action Required
(No flag)	Storage space no longer exists.	Perform a point-in-time cold restore to a time before the storage space was dropped.
D	Chunk is down or storage space is disabled.	Perform a warm restore of the affected storage space.
I	Chunk has been physically restored, but needs a logical restore.	Perform a logical restore.
L	Storage space is being logically restored.	Retry the logical restore.
N	Chunk is renamed and either down or inconsistent. (IDS)	Perform a warm restore of the chunk when the physical device is available.
O	Chunk is online.	No action required.
P	Storage space is physically restored.	Perform a logical restore, if one is not already in progress.
R	Storage space is being restored.	Perform a physical or logical restore.
X	Storage space or chunk is newly mirrored.	No action required.

Ensuring That Storage Devices Are Available

Verify that storage devices and files are available before you begin a restore. For example, when you drop a dbspace or mirroring for a dbspace after your level-0 backup, you must ensure that the dbspace or mirror chunk device is available to the database server when you begin the restore. If the storage device is not available, the database server cannot write to the chunk and the restore fails.

When you add a chunk after your last backup, you must ensure that the chunk device is available to the database server when it rolls forward the logical log.

Restoring Save Sets with ISM

If you are using ISM, you can restore data from save sets on the storage volume. When the ISM server receives a restore request, the **ism_watch** command prompts you to mount the required storage volume on the storage device. When you mount the volume, the restore will resume.

You can set the retention period for either a save set or volume. Unless all the save sets on the volume have expired, you can use ON–Bar to restore it.

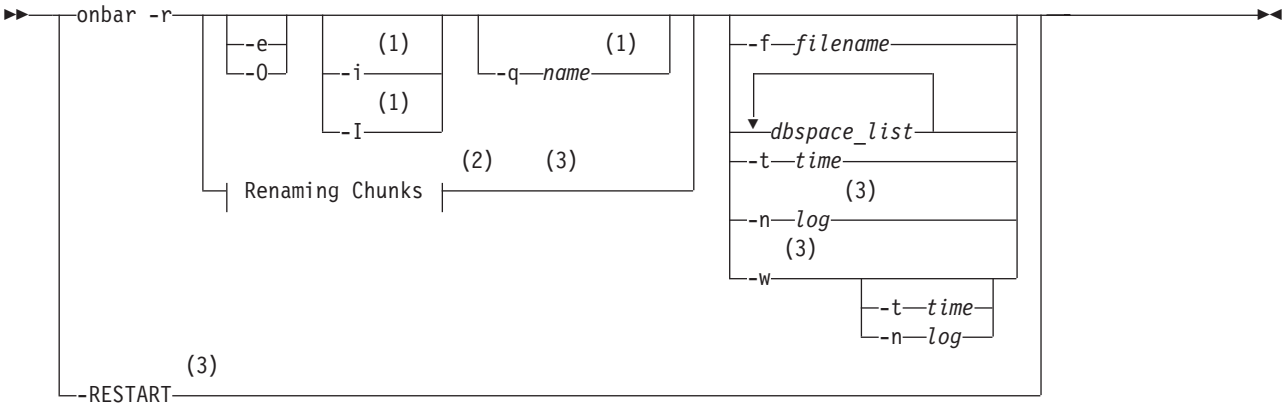
After the retention period for a save set expires, ON–Bar can no longer restore it. To recreate an expired save set, use the **ism_catalog -recreate from** command.

If you set the retention period for a volume, ISM retains the save sets until all the save sets on that volume have expired. To recover an expired volume, use the **ism_catalog -recover from** command. For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

Performing a Complete Restore

This diagram shows the syntax for the **onbar -r** commands.

Performing a Complete Restore



Notes:

1 Extended Parallel Server Only

2 See 7-27

3 Dynamic Server Only

Element	Purpose	Key Considerations
onbar -r	Specifies a restore	In a cold restore, the **-r** option restores all storage spaces, salvages and restores the logical logs. In a warm restore, the **-r** option restores all offline storage spaces and restores the logical logs. You must specify the **-r** parameter first.
dbspace_list	Names one or more dbspaces, blobspaces (IDS), sbspaces (IDS), or dbslices (XPS) to be restored	ON–Bar restores only the storage spaces listed. If it is a cold restore, you must list all the critical dbspaces. If you enter more than one storage-space name, use a space to separate the names.

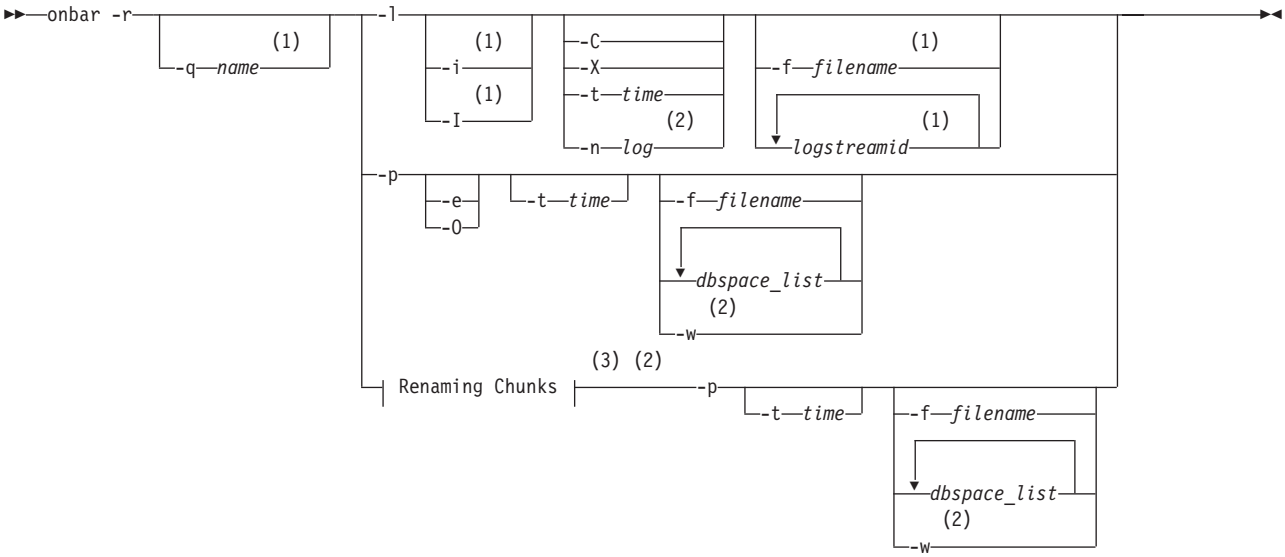
Element	Purpose	Key Considerations
-e	Specifies an external restore	After you externally restore the storage spaces with a third-party utility, run **onbar -r -e** to mark the storage spaces as physically restored, restore the logical logs, and bring the storage spaces back online. For details, see "Using External Restore Commands" on page 8-11.
-f *filename*	Restores the storage spaces that are listed in the text file whose path name *filename* provides Use this option to avoid entering a long list of storage spaces every time that you use this option.	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile2**), and absolute (**/usr/informix/backup_lists/listfile3** or **c:\informix\backup_lists\listfile3**) filenames. This file can list multiple storage spaces per line.
-i (XPS)	Queries for indexes to be rebuilt and trigger index rebuild if necessary	This option triggers the rebuild of any indexes that can be rebuilt. It can be used in conjunction with the **-r** option only, and when used, no additional options can be included. The **onbar -r -i** command only triggers index rebuild and does not actually restore any storage spaces. You cannot limit the action of **onbar -r -i** to a subset of storage spaces and you cannot supply a list of dbspaces or dbslices.
-I (XPS)	Does not query for indexes to be rebuilt and does not trigger index rebuild.	This option is used to control index building. When used, ON-Bar does not query for indexes that need to be built and does not rebuild them. When logical log replay encounters a CREATE INDEX log record, by default ON-Bar schedules each index for rebuild at the end of the logical roll forward. Because rebuilding an index can take a long time and requires shared locks on all table fragments on which the index is defined, consider turning off the automatic rebuilding of indexes with this option. At a later time, use the **onbar -r -i** command to rebuild indexes, or use SQL commands to drop and redefine such indexes manually.
-n *log*	Indicates the *uniqid* of the log to restore in a cold restore To find the uniqid number, use the **onstat -l** command (IDS).	A point-in-log restore is a special kind of point-in-time restore. You must restore all storage spaces in a point-in-log restore so that the data is consistent. If any logical logs exist after this one, ON–Bar does not restore them and their data is lost.

Element	Purpose	Key Considerations
-O	Overrides internal error checks. Allows the restore of online storage spaces. Forces the recreation of chunk files that no longer exist.	Used to override internal error checks to perform the following tasks: • Force the restore of online storage spaces. If a storage space in the list of storage spaces to restore is online, the **-O** option allows ON-Bar to bring the storage space offline and then restore it. If this operation succeeds, ON-Bar completes with an exit code of 177. • Force the creation of nonexistent chunk files. If a chunk file for a storage space being restored no longer exists, the **-O** option allows ON-Bar to recreate it. The newly created chunk file is cooked disk space, not raw disk space. If ON-Bar successfully recreates the missing chunk file, ON-Bar completes with an exit code of 179. • Force a cold restore to proceed if a critical storage space is missing. In a cold restore, ON-Bar checks whether every critical space is being restored. This check occasionally causes false warnings. Use the **-O** option to override this check. If the warning was valid, the restore fails. If the warning was false and ON-Bar successfully restores the server, ON-Bar completes with an exit code of 115.
-q ***name***	Assigns a name to the restore This name appears in the **onstat** utility so that you can follow the progress of the backup session (XPS).	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 128 characters.
-RESTART	Restarts a restore after a database server or ON–Bar failure (IDS)	For the restore to be restartable, the RESTARTABLE_RESTORE configuration parameter must be set to ON when the restore failure occurs. If RESTARTABLE_RESTORE is set to OFF, the **-RESTART** option does not work. For more information, see "Using Restartable Restore to Recover Data (IDS)" on page 7-35.
-t ***time***	Specifies the time of the last transaction to be restored from the logical logs in a cold restore	You must specify the **onbar -r -t** option (point-in-time) in a cold restore only and must restore all storage spaces to the same time. For more information, see "Restoring Data to a Point in Time" on page 7-20.
-w	Performs a whole-system restore of all storage spaces and logical logs from the last whole-system backup (IDS)	You must specify the **-w** option in a cold restore. If you specify **onbar -r -w** without a whole-system backup, return code 147 appears because ON–Bar cannot find any storage spaces backed up as part of a whole-system backup.

Performing a Physical-Only or Logical-Only Restore

This diagram shows the syntax for a physical-only or logical-only restore.

Performing a Physical-Only or Logical-Only Restore



Notes:

1 Extended Parallel Server Only

2 Dynamic Server Only

3 See 7-27

Element	Purpose	Key Considerations
onbar -r	If specified with the **-p** option, restores the storage spaces only If specified with the **-l** option, restores the logical logs only.	You must specify the **-r** parameter first.
-C	Restores logs from the current logical log tape without sending prompts to mount the tape.	The server is placed in suspend log restore state, and the command exits after the last applicable log is restored. The server sends a prompt if a log spans tapes. The configuration parameter RESTARTABLE_RESTORE does not impact continuous log restoration.
dbspace_list	Names one or more dbspaces, blobspaces (IDS), sbspaces (IDS), or dbslices (XPS) to be restored	ON–Bar restores only the storage spaces listed. If it is a cold restore, you must list all the critical dbspaces. If you enter more than one storage-space name, use a space to separate the names.
-e	Specifies an external restore	After you externally restore the storage spaces with a third-party utility, run **onbar -r -e** to mark the storage spaces as physically restored, restore the logical logs, and bring the storage spaces back online. For details, see "Using External Restore Commands" on page 8-11.
-f *filename*	Restores the storage spaces that are listed in the text file whose path name *filename* provides Use this option to avoid entering a long list of storage spaces every time that you use this option.	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile2**), and absolute (**/usr/informix/backup_lists/listfile3** or **c:\informix\backup_lists\listfile3**) filenames. This file can list multiple storage spaces per line.

Element	Purpose	Key Considerations
-i (XPS)	Queries for indexes to be rebuilt and trigger index rebuild if necessary	This option triggers the rebuild of any indexes that can be rebuilt. It can be used in conjunction with the **-r** option only, and when used, no additional options can be included. The **onbar -r -i** command only triggers index rebuild and does not actually restore any storage spaces. You cannot limit the action of **onbar -r -i** to a subset of storage spaces and you cannot supply a list of dbspaces or dbslices.
-I (XPS)	Does not query for indexes to be rebuilt and do not trigger index rebuild.	This option is used to control index building. When used, ON-Bar does not query for indexes that need to be built and does not rebuild them. When logical log replay encounters a CREATE INDEX log record, by default ON-Bar schedules each such index for rebuild at the end of the logical roll forward. Since rebuilding an index can require a significant amount of time and requires shared locks to be held on all table fragments on which the index is defined, you might want to turn off the automatic rebuilding of indexes by using this option. At a later time you can use the **onbar -r -i** command to rebuild indexes, or you can use SQL commands to drop and redefine such indexes manually.
-l	Specifies a logical restore only Restores and rolls forward the logical logs.	The logical restore applies only to those storage spaces that have already been physically restored.
logstreamid	Uniquely identifies logical-log records that a given coserver generates (XPS)	If you supply more than one *logstreamid*, separate each item in the list with a space. A logstream is a coserver ID.
-n *log*	Indicates the *uniqid* of the last log to restore in a cold restore To find the uniqid number, use the **onstat -l** command (IDS).	A point-in-log restore is a special kind of point-in-time restore. You must restore all storage spaces in a point-in-log restore so that the data is consistent. If any logical logs exist after this one, ON–Bar does not restore them and their data is lost.
-O	Overrides internal error checks. Allows the restore of online storage spaces. Forces the recreation of chunk files that no longer exist.	Used to override internal error checks to perform the following tasks: • Force the restore of online storage spaces. If a storage space in the list of storage spaces to restore is online, the **-O** option allows ON-Bar to bring the storage space offline and then restore it. If this operation succeeds, ON-Bar completes with an exit code of 177. • Force the creation of nonexistent chunk files. If a chunk file for a storage space being restored no longer exists, the **-O** option allows ON-Bar to recreate it. The newly created chunk file is cooked disk space, not raw disk space. If ON-Bar successfully recreates the missing chunk file, ON-Bar completes with an exit code of 179. • Force a cold restore to proceed if a critical storage space is missing. In a cold restore, ON-Bar checks whether every critical space is being restored. This check occasionally causes spurious warnings. Use the **-O** option to override this check. If the warning was valid, the restore will fail. If the warning was spurious and ON-Bar successfully restores the server, ON-Bar completes with an exit code of 115.

Element	Purpose	Key Considerations
-p	Specifies a physical restore only	You must follow a physical restore with a logical restore before data is accessible unless you use a whole-system restore. This option turns off automatic log salvage before a cold restore.
-q *name*	Allows you to assign a name to the restore This name appears in the **onstat** utility so that you can follow the progress of the backup session (XPS).	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 128 characters.
-t *time*	Specifies the time of the last transaction to be restored from the logical logs in a cold restore	You can specify the **onbar -r -p -t** command in a warm or cold restore to restore specific storage spaces from an old physical backup. You must then use **onbar -r -l** to finish the logical restore. For more information, see "Restoring from an Older Backup" on page 7-22.
-w	Performs a whole-system restore (IDS)	To restore the storage spaces only, specify the **-w -p** option in a cold restore.
-X	Quiesces a server in logical restore suspend state without restoring additional logs.	

Using IBM Informix Server Administrator to Restore Data

You can use IBMInformix Server Administrator (ISA) to perform backups and restores with ON–Bar. For more information, see the ISA online help.

Examples of ON-Bar Restore Commands

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

For an example of renaming chunks during a cold restore, see "Renaming Chunks During a Restore (IDS)" on page 7-26.

Performing a Restore

To perform a complete cold or warm restore, use the **onbar -r** command. ON–Bar restores the storage spaces in parallel. To speed up restores, you can add additional CPU virtual processors. To perform a restore, use the following command:

```
onbar -r
```

In a warm restore, the **-r** option restores all down storage spaces and logical logs, and skips online storage spaces. A *down* storage space means it is offline or a chunk in it is inconsistent.

You cannot perform more than one warm restore simultaneously. If you need to restore multiple storage spaces, specify the set of storage spaces to restore to ON-Bar (see "Restoring Specific Storage Spaces" on page 7-14) or allow ON-Bar to restore all down storage spaces by not explicitly specifying any spaces.

In a cold restore, the **-r** option automatically salvages the logical logs, and restores all storage spaces and appropriate logical logs.

Tip: For faster performance in a restore, assign separate storage devices for backing up storage spaces and logical logs. If physical and logical backups are mixed together on the storage media, it takes longer to scan the media during a restore.

Restoring Specific Storage Spaces

To restore particular storage spaces (for example, a dbspaces named fin_dbspace1 and fin_dbspace2), use the -r option, as the following example shows:

```
onbar -r fin_dbspace1 fin_dbspace2
```

You can also specify the storage spaces to restore by listing them in a text file and passing the path name of the file to ON-Bar with the **-f** option.

If any of the dbspaces that you request to restore are online, they are skipped in the restore. ON–Bar writes a message to the activity log about the skipped dbspaces.

Performing a Logical Restore

To perform a logical restore, use the following command:

```
onbar -r -l
```

Warning: Because the logical-log files are replayed using temporary space during a warm restore, make sure that you have enough temporary space for the logical restore.

The minimum amount of temporary space that the database server needs is equal to:

Dynamic Server

- The total logical-log space for the database server instance, or the number of log files to be replayed, whichever is smaller.

  Tip: To improve performance, replay logical-log transactions in parallel during a warm restore. Use the ON_RECVRY_THREADS configuration parameter to set the number of parallel threads. To replay logical-log transactions in parallel during a cold restore, use the OFF_RECVRY_THREADS configuration parameter. For more information, see your *IBM Informix Performance Guide*.

End of Dynamic Server

Extended Parallel Server

- The total logical-log space for the coservers on which storage spaces are being restored, or the number of log files to be replayed, whichever is smaller. Each coserver must have enough temporary space for all its temporary log files.

End of Extended Parallel Server

Skipping Logical Replay (XPS)

The database server automatically skips logical replay when it is not necessary during a warm restore. For example, if you lose a dbspace that contains a large table that has not changed since its last backup, you can quickly restore it without replaying the logical logs. If *all* dbspaces being restored on a coserver meet the following criteria, logical replay is skipped for the warm restore:

- A backup or set of incremental backups exists for the dbspaces.
- No logging activity has occurred for the dbspaces since the last backup. Read-only (static) dbspaces will always satisfy this condition, unless they were changed to read-only after the last backup was taken.

If dbspaces are being restored on multiple coservers, logical replay is skipped on those coservers where no dbspaces need it and performed on those coservers where at least one dbspace needs it.

In the following **onstat -d** example, the **S** flag shows that dbspace **dbnoreplay** is a candidate for skipping logical replay. This **S** flag disappears the first time that a logging operation occurs in that dbspace.

```
Dbspaces
address number flags fchunk nchunks flags owner    name
a66c140 1      1     1      1       N     informix rootdbs
a68bea8 2      20001 2      1       N  S  informix dbnoreplay
```

The database server always replays the logical logs during a cold restore.

Restoring Data on Extended Parallel Server

To restore all dbspaces in a dbslice named **fin_slice**, use the following command:

```
onbar -r fin_slice
```

If you have a mixture of dbspaces to restore, some that require logical replay and some that do not, follow these steps:

1. Restore the dbspaces that do not require logical replay first.
2. Restore the dbspaces that require logical replay.

You can use the dbspaces restored in the first pass sooner but the total restore time might be longer. This method enables you to quickly restore tables in a data warehouse. For more information, see "Skipping Logical Replay (XPS)" on page 7-14.

Performing a Physical Restore Followed By a Logical Restore

In certain situations, you might want to perform a restore in stages. If multiple devices are available for the restore, you can restore multiple storage spaces separately or concurrently, and then perform a single logical restore.

To perform a warm restore in stages:

1. Perform a physical restore:
   ```
   onbar -r -p
   ```
2. Back up the logical logs:
   ```
   onbar -b -l
   ```
3. Perform a logical restore:
   ```
   onbar -r -l
   ```

To perform a cold restore in stages:

1. Optionally, salvage the logical logs manually:
   ```
   onbar -b -l -s
   ```
   To perform a cold restore without salvaging the logical logs, skip this step.
2. Perform a physical restore:
   ```
   onbar -r -p
   ```
3. Perform a logical restore:

```
onbar -r -l
```

4. Synchronize the **sysutils** database and emergency boot files:

   ```
   onsmsync
   ```

   You must run **onsmsync** (without arguments) after performing a cold restore that salvages logs.

For information on what actions to take when an error occurs during a physical or logical restore, see "Resolving a Failed Restore" on page 7-38.

Salvaging Logical Logs

Decide whether you want to salvage the logical logs before you perform a cold restore. If not, the data in the logical logs that has not been backed up is lost. If a disk is damaged, salvage the logs if they are still accessible before you replace the disk. For more information, see "Performing a Cold Restore" on page 7-16.

The **onbar -r** command automatically salvages the logical logs. Use the **onbar -r -p** and **onbar -r -l** commands if you want to skip log salvage.

Dynamic Server

If you set the LTAPEDEV configuration parameter to **/dev/null** on UNIX or to NUL or **\dev\nul** on Windows, the logical logs are *not* salvaged in any ON–Bar restore (**onbar -r** or **onbar -r -w**, for example).

End of Dynamic Server

Avoid salvaging the logical logs in the following situations:

- When you perform an imported restore

  Salvage the logical logs on the source database server but not on the target database server. For more information, see "Transferring Data with the Imported Restore" on page 7-29.
- If you reinitialize the database server (**oninit -i**) before you perform a cold restore

  Reinitialization creates new logical logs that do not contain the data that you want to restore.
- If you install a new disk for the dbspace that contains the logical logs

  Salvage the logs from the old disk, but not from the new disk.

Performing a Cold Restore

If a critical storage space is damaged because of a disk failure or corrupted data, you must perform a cold restore. If a disk fails, you need to replace it before you can perform a cold restore to recover data.

Extended Parallel Server

On Extended Parallel Server, if a critical dbspace on any of the coservers goes down, you must perform a cold restore on all coservers.

End of Extended Parallel Server

Warning: Back up all storage spaces before you perform a cold restore. If you do not and you try to perform a cold restore, data in the storage spaces that

were not backed up will be lost. The storage space is marked as offline after the cold restore. Drop the storage space so that you can reuse its disk space.

To perform a cold restore with automatic log salvage:

1. Copy the administrative files to a safe place: ONCONFIG, **sqlhosts** (UNIX only), emergency boot files, **oncfg** files, and **xcfg** files (XPS only).
2. Take the database server offline.

   Dynamic Server

   Use the following command:

   ```
   onmode -ky
   ```

   End of Dynamic Server

   Extended Parallel Server

   Take the database server offline and then bring it to microkernel mode:

   ```
   xctl onmode -ky
   xctl -C oninit -m
   ```

   End of Extended Parallel Server

3. If the disk that contains the logical-log files needs to be replaced or repaired, use the following command to salvage logical-log files on the damaged disk:

   ```
   onbar -b -l -s
   ```

4. Then repair or replace the disk.
5. If the files in **INFORMIXDIR** were destroyed, copy the previously saved administrative files to their original locations.

   However, if you did the cold restore because a critical dbspace was lost, you do not need to copy the administrative files. For more information, see "Administrative Files to Back Up" on page 5-4.
6. To restore the critical and noncritical storage spaces, use the following command:

   ```
   onbar -r
   ```

   When the restore is complete, the database server is in quiescent mode.
7. To bring the database server online, use the following command:

   ```
   onmode -m
   ```

Performing a Whole-System Restore (IDS)

A whole-system restore must be a cold restore and it must restore all storage spaces. A whole-system restore does not require you to restore the logical logs.

A whole-system restore requires a whole-system backup. However, you can perform a plain restore of a whole-system backup. If you use **onbar -b -w** to back up the whole system, you can restore with any of the following commands:

```
onbar -r -w         # whole-system restore (salvages logs automatically)
onbar -r -p -w      # physical-only whole-system restore (no log salvage)
onbar -r            # parallel restore of the whole-system backup
onbar -r dbspaces   # restore dbspaces from a whole-system backup
onbar -r -t time    # point-in-time restore
onbar -r -t time -w # whole-system point-in-time restore
```

If you use **onbar -r -p -w**, the database server is in fast recovery mode when the restore completes. Perform either a logical restore (**onbar -r -l**) or use **onmode -m** to bring the database server online. For more information, see "Performing a Whole-System Backup (IDS)" on page 5-15.

Considerations When LTAPEDEV is Set to Null: A whole-system backup with LTAPEDEV set to **/dev/null** on UNIX or to **\dev\nul** on Windows does not back up the logical logs.

To restore the data from a whole-system backup when LTAPEDEV is null:

1. Upon restore, you must use the **onbar -r -w -p** command.

   When the physical-only whole-system restore completes, the database server is in fast recovery mode.
2. If the database server is offline, use the **onmode -sy** command to perform fast recovery.

   If the database server is online, use the **onmode -m** command to perform fast recovery.

Using the -O Option in a Whole-System Restore: Use the **-O** option with a whole-system restore only to recreate missing chunk files. You cannot use the **onbar -r -w -O** command when the database server is online because the root dbspace cannot be taken offline during the whole-system restore.

Restoring Data Using a Mixed Restore

You can use mixed restore to reduce the time until urgent data becomes online and available when you need to restore the server. Urgent data is data that you deem as critical to your business operation, and should not be confused with critical dbspaces in the Informix server (the root dbspace and any dbspaces containing the physical or logical logs).

In a mixed restore you perform a cold restore on only the critical dbspaces and dbspaces containing your urgent data first. Because you do not restore all dbspaces in the system and you save the time necessary to restore those dbspaces, you can bring the server online faster. You then restore the remaining dbspaces in one or more warm restores.

Important: You should run the **onsmsync** utility, without arguments, after the initial cold restore but before any warm restores.

For example, consider a database server with four dbspaces in addition to the root dbspace: **logdbs**, **dbs_1**, **dbs_2**, and **dbs_3**. Suppose the logical logs are stored in **logdbs** and the physical log is in the root dbspace. The critical dbspaces that must be restored during the initial cold restore are **rootdbs** and **logdbs**:

```
onbar -r rootdbs logdbs dbs_1
```

When the cold restore completes, you can bring the server online and any data stored in **rootdbs**, **logdbs**, and **dbs_1** becomes accessible.

Next, run the **onsmsync** utility without arguments:

```
onsmsync
```

You can then perform a warm restore of **dbs_2**:

```
onbar -r dbs_2
```

Finally, you can perform a warm restore of all remaining dbspaces (for this example, only **dbs_3**):

```
onbar -r
```

Instead of performing two warm restores, you could have issued the **onbar -r** command, without specifying a list of dbspaces, immediately after the initial cold restore. This command would have restored all dbspaces remaining to be restored: **dbs_2** and **dbs_3**. Conversely, in a larger system with dozens of dbspaces, you could divide the warm restore portion of the mixed restore into several warm restores, each restoring only a small subset of the dbspaces remaining to be restored in the system.

Tip: If you do not run **onsmsync** after the cold part of the mixed restore, ON-Bar automatically runs **onsmsync**. You should run **onsmsync** as a separate step so that you can address any errors that might occur. If you allow ON-Bar to run **onsmsync** and **onsmsync** fails, the restore proceeds but might fail.

Tip: You can perform both the initial cold restore and each subsequent warm restore in stages, as described in the section "Performing a Physical Restore Followed By a Logical Restore" on page 7-15.

Strategies for Using Mixed Restore: To successfully implement a mixed-restore strategy, you should carefully select the set of dbspaces in which you place your databases and database objects at the time you create them. ON-Bar backs up and restores physical, not logical, entities. Thus, ON-Bar cannot restore a particular database or a particular set of tables. Instead, ON-Bar restores a particular set of storage spaces. It is up to you to track what is stored in those storage spaces.

For example, consider a database with the catalogs in the dbspace **cat_dbs**:

```
create database mydb in cat_dbs with log;
```

A table in this database is fragmented among the dbspaces **tab_dbs_1** and **tab_dbs_2**:

```
create table mytab (i integer, c char(20))
fragment by round robin in tab_dbs_1, tab_dbs_2;
```

An index for the table is stored in the dbspace **idx_dbs**:

```
create index myidx on mytab(i) in idx_dbs;
```

If you need to restore the server, you cannot access all of the data in the example database until you have restored the dbspaces containing the database catalogs, table data, and index: in this case, the dbspaces **cat_dbsp**, **tab_dbs_1**, **tab_dbs_2**, and **idx_dbsp**.

To simplify the management and tracking of your data, it is recommended that you divide your set of dbspaces into subsets in which you store data of a particular urgency. When you create your database objects, place them in dbspaces appropriate to their urgency. For example, if you have data with three levels of urgency, you might want to place all the objects (database catalogs, tables, and indexes) associated with your most urgent data in a particular set of dbspaces or dbslices: for example, **urgent_dbs_1**, **urgent_dbs_2**, ...**urgent_dbs_*n***. You would place all the objects associated with less urgent data in a different set of dbspaces (or dbslices): for example, **less_urgent_dbs_1**, **less_urgent_dbs_2**, ... **less_urgent_dbs_*k***. Lastly, you would place your remaining data in a different set of dbspaces (or dbslices): for example, **non_urgent_dbs_1**, **non_urgent_dbs_2**, ... **non_urgent_dbs_*r***.

If you need to restore the server, you would first perform a cold restore of all critical dbspaces and dbspaces containing urgent data, **urgent_dbs_1** through **urgent_dbs_*n***. For example, assume logical logs are distributed among two dbspaces, **logdbsp_1** and **logdbsp_2**, and the physical log is in **rootdbs**. The critical **dbspaces** are therefore **rootdbs**, **logdbsp_1**, and **logdbsp_2**.

You would perform the initial cold restore by issuing the following ON-Bar command:

```
onbar -r rootdbs logdbsp_1 logdbsp_2 urgent_dbs_1 ...  urgent_dbs_2
```

At this point you can bring the server online and all business-urgent data is available.

Next, perform a warm restore for the less-urgent data:

```
onsmsync
onbar -r less_urgent_dbs_1 less_urgent_dbs_2 .....  less_urgent_dbs_k
```

Finally, you can perform a warm restore for the rest of the server by issuing the following command:

```
onbar -r non_urgent_dbs_1 non_urgent_dbs_2 ... non_urgent_dbs_r
```

Alternatively, you can use the following command to restore all storage spaces:

```
onbar -r
```

Restoring Data to a Point in Time

A point-in-time restore enables you to restore the database server to the state it was in at a particular point in time. A point-in-time restore is typically used to recover from a mistake. For example, if you accidentally dropped a database, you can restore the server to a point in time just before you dropped the database.

A point-in-time restore is specified by including the **-t** option in the ON-Bar command; for example, **onbar -r -t *time***. If you use the **onbar -r -t *time*** command, you must restore all storage spaces to the same point in time.

Extended Parallel Server

Tip: In a multiple coserver configuration, a point-in-time restore automatically restores all coservers to the same point in time. You cannot restore individual coservers.

End of Extended Parallel Server

Important: To determine the appropriate date and time for the point-in-time restore, use the **onlog** utility that the *IBM Informix Administrator's Reference* describes. The **onlog** output displays the date and time of the committed transactions in the logical log. All data transactions that occur after *time* or *last_log* are lost.

Performing a Cold Point-in-Time Restore:

To restore database server data to its state at a specific date and time, enter a command using the date and time format for your GLS locale, as this example shows:

```
onbar -r -t "2004-05-10 11:35:57"
```

In this example, the restore replays transactions that committed on or before the specified time, including any transactions with a commit time of 11:35:57. Transactions in progress but not committed by 11:35:57 are rolled back.

Quotation marks are recommended around the date and time. The format for the English locale is **yyyy-mm-dd hh:mm:ss**. If the **GL_DATETIME** environment variable is set, you must specify the date and time according to that variable. For an example of using a point-in-time restore in a non-English locale, see "Point-in-Time Restore Example" on page D-2.

Dynamic Server

You can also perform a whole-system, point-in-time restore.

End of Dynamic Server

Performing a Point-in-Time Cold Restore in Stages: You can perform a point-in-time cold restore in stages, similar to an ordinary restore in stages as described in the section "Performing a Physical Restore Followed By a Logical Restore" on page 7-15. Use the **-t** *time* option for both the physical and logical restore steps:

```
onbar -r -p -t "2004-05-10 11:35:57"
onbar -r -l -t "2004-05-10 11:35:57"
```

Important: The point-in-time values for the **-t** option must be identical for both the physical and logical restore.

Performing a Point-in-Time Mixed Restore: You can perform a point-in-time mixed restore. Supply a point-in-time value to the initial cold restore by using the **-t** *time* option, then restore the remaining storage spaces in one or more warm restores. Do not include the **-t** *time* option in the warm restores.

The following example performs a cold restore for a subset of the storage spaces (including all critical dbspaces) in the initial cold restore, and then performs a warm restore for **dbspace_2** and **dbspace_3**, followed by a warm restore of **dbspace_4** and **dbspace_5**, and finally performs a warm restore of all remaining storage spaces:

```
onbar -r -t "2004-05-10 11:35:57" rootdbs logspace_1 dbspace_1
onsmsync
onbar -r dbspace_2 dbspace_3
onbar -r dbspace_4 dbspace_5
onbar -r
```

Tip: You can perform the cold part of the mixed restore in stages, as described in the section "Performing a Point-in-Time Cold Restore in Stages." Supply a list of dbspaces or dbslices to the physical restore. For example:

```
onbar -r -p -t "2004-05-10 11:35:57" rootdbs_1 dbslice_1
onbar -r -l -t "2004-05-10 11:35:57"
```

Performing a Point-In-Time Restore with Multiple Timelines : When you perform more than one point-in-time restore, you create multiple timelines. You can specify any time in any timeline with the **onbar -r -t** *time* command.

Performing a Point-in-Time Warm Restore: You cannot perform a point-in-time warm restore. A warm restore of a dbspace will always roll forward the dbspace to the latest time available in the logical logs.

Restoring from an Older Backup

By default, ON–Bar restores the latest backup. If you do not want to restore this backup (for example, when backup verification failed or the backup media was lost), you can restore from an older backup.

To restore from an older backup using a physical point-in-time restore:

1. Find the time of the older backup in the message log or ON–Bar activity log or from the storage manager.
2. To restore all or specific storage spaces, issue the following commands:

```
onbar -r -p -t time [dbspaces_from_older_backup]
onbar -r -p [remaining_dbspaces]
onbar -r -l
```

To restore from an older backup by expiring the bad backup:

1. Expire the bad backup in the storage manager.
2. Run **onsmsync** without arguments.
3. To restore the data, issue the following command:

```
onbar -r
```

Performing a Point-in-Log Restore (IDS)

A point-in-log restore is similar to a point-in-time restore. The point-in-log restore stops at the time of the last committed transaction listed in the logical log. You must use point-in-log restore in a cold restore only and you must restore all storage spaces. To perform a point-in-log restore, use the following command:

```
onbar -r -n log_id
```

You can specify any log ID from any timeline to restore to a specific logical log. If the specific logical log applies to more than one timeline, then ON-Bar uses the latest one.

Restoring Online Storage Spaces

Use the following command to force a restore of online storage spaces (except critical dbspaces) in a warm restore:

```
onbar -r -O dbsp1 dbsp2
```

The database server automatically shuts down each storage space before it starts to restore it. Taking the storage space offline ensures that users do not try to update its tables during the restore process.

Dynamic Server

For special considerations on using the **-O** option, see "Using the -O Option in a Whole-System Restore" on page 7-18.

End of Dynamic Server

Recreating Chunk Files During a Restore

If the disk or file system fails, one or more chunk files could be missing from the dbspace. If you use the **-O** option in a warm or cold restore, ON–Bar recreates the missing chunk files, including any necessary directories, before restoring the dbspace as long as enough space exists on the file system. The newly created chunk files are cooked files and are owned by group **informix** on UNIX or group **Informix-Admin** on Windows.

To restore when using cooked chunks:

1. Install the new disk.
2. On UNIX, mount the device as a file system.

   On Windows, format the disk.
3. Allocate disk space for the chunk file.

   For instructions, see the chapter on managing data in the *IBM Informix Administrator's Guide*.
4. Issue the following command to recreate the chunk files and restore the dbspace:

   ```
   onbar -r -O crashedspace
   ```

Important: ON–Bar does not recreate chunk files during a logical restore if the logical logs contain chunk-creation records.

To restore when using raw chunks:

1. Install the new disk.

UNIX Only

2. If you use symbolic links to raw devices, create new links for the down chunks that point to the newly installed disk.

   ON–Bar restores the chunk file to where the symbolic link points.

End of UNIX Only

3. Issue the following command to restore the dbspace:

   ```
   onbar -r crashedspace
   ```

Restoring a Dropped Storage Space

If you accidentally drop a storage space, you can use a point-in-time restore or a point-in-log restore to recover it.

To restore a dropped storage space using separate physical and logical restores:

1. In this example, the database server has **dbspace1**, which was dropped accidentally, and **dbspace2**. Use **onlog** or the database server message log to obtain a time before **dbspace1** was dropped.
2. Shut down the database server.
3. Perform a physical-only restore of all storage spaces:

   ```
   onbar -r -p -t time rootdbs dbspace1 dbspace2
   ```
4. To restore the dropped storage space and prevent the logical log from replaying the drop, enter one of the following commands:
   a. If you use the point-in-log command, specify the *uniqid* of the log before the log that contains the drop command:

      ```
      onbar -r -l -n uniqid
      ```
   b. If you use the logical point-in-time command, use the same time as in step 3:

      ```
      onbar -r -l -t time
      ```

To restore a dropped storage space when the chunk files were also deleted:

1. Use the **onlog** utility to find the logical-log file that contains the dropped transaction for the storage space.

2. To restore a dropped storage space when the chunk files were deleted, enter the following command:

   ```
   onbar -r -t time -O
   ```

   The point-in-time restore restores the dropped storage space and automatically recreates the chunk files.

Warning: You must restore the data to a point in time before the storage space was dropped in both the physical and logical restores.

Deferring Index Rebuild After Logical Restore (XPS)

When logical log replay encounters a CREATE INDEX log record, by default ON-Bar schedules each such index for rebuild at the end of logical roll forward. Since rebuilding an index can require a significant amount of time and requires share locks to be held on all table fragments on which the index is defined, you might want to turn off the automatic rebuilding of indexes.

To prevent ON-Bar from rebuilding indexes at the end of logical roll forward, use the **-I** option in the restore command.

For example, the following command restores the entire system, without rebuilding indexes:

```
onbar -r -I
```

When you use the **-I** option with ON-Bar, any indexes that did not already exist at the time the backup was performed will be unavailable or might be unusable after the restore.

You can recreate the down indexes in one of the following ways:

- Drop and redefine them manually using the DROP INDEX and CREATE INDEX SQL commands.
- Use the **onbar -r -i** command. The **onbar -r -i** command only triggers index rebuild among storage spaces that are already online; it does not restore any storage spaces.

The **-I** and **-i** options are particularly useful if you have created many indexes since your last backup and have performed a mixed restore on the system. In this case, you might want to use the **-I** option in each phase of the mixed restore. Then, after you have restored the entire system, run the **onbar -r -i** command to recreate the down indexes.

Restoring Data when Reinitializing the Database Server

Any backups that you performed before reinitializing the database server are unusable. During initialization, ON–Bar saves the emergency boot file elsewhere and starts a new, empty emergency boot file. Do not use the copy of the emergency boot file unless you want to restore the previous database server instance.

To reinitialize the database server after a failure when you do not need the old data:

1. Do not copy the old emergency boot file into the database server directory (**$INFORMIXDIR/etc** on UNIX or **%INFORMIXDIR%\etc** on Windows).
2. To perform a complete backup, use **ON-Bar -b**.

To reinitialize the database server and restore the old data:

1. Before you reinitialize the database server, copy the administrative files (emergency boot, **oncfg**, and ONCONFIG files) to a different directory, if possible.
2. Reinitialize the database server.
3. Recopy the administrative files into the database server directory because you need the information in the old emergency boot file.

   If the administrative files are unavailable, copy them from the last backup into the database server directory.
4. Perform a restore. Do not salvage the logical logs.
5. Any changes made after reinitialization are lost.
6. Verify that you restored the correct instance of the critical and noncritical storage spaces.

Configuring Continuous Log Restore using ON-Bar

Use continuous log restore to keep a second system (hot backup) available to replace the primary system if the primary system fails. For more information, see "Continuous Log Restore" on page 7-5.

Prerequisites:

The version of Dynamic Server must be identical on both the primary and secondary systems.

Procedure:

1. On the primary system, perform a level-0 backup with the following command:

   ```
   onbar -b -L 0
   ```
2. Import the backup objects that were created to the storage manager of the secondary server. For more information about importing backup objects, see "Importing a Restore" on page 7-30.
3. On the secondary system, perform a physical restore with the following command:

   ```
   onbar -r -p
   ```

   After the physical restore completes on the secondary system, the database server waits in fast recovery mode to restore logical logs.
4. On the primary system, back up logical logs with the following command:

   ```
   onbar -b -l
   ```
5. Transfer the backed up logical logs to the secondary system and restore them with the following command:

   ```
   onbar -r -l -C
   ```
6. Repeat steps 4 and 5 for all logical logs that are available to back up and restore.
7. If you are doing continuous log restore on a secondary system as an emergency standby, run the following commands to complete restoring logical logs and quiesce the server.

   If logical logs are available to restore:

   ```
   onbar -r -l
   ```

   After all available logical logs are restored:

   ```
   onbar -r -l -X
   ```

Renaming Chunks During a Restore (IDS)

You can rename chunks by specifying new chunks paths and offsets during a cold restore with ON–Bar. This option is useful if you need to restore storage spaces to a different disk from the one on which the backup was made. You can rename any type of chunk, including critical chunks and mirror chunks.

Tip: If you use symbolic links to chunk names, you might not need to rename chunks; you need only edit the symbolic name definitions. For more information, see the *IBM Informix Administrator's Guide*.

Key Considerations

During a cold restore, ON–Bar performs the following validations to rename chunks:

1. It validates that the old chunk path names and offsets exist in the archive reserved pages.
2. It validates that the new chunk path names and offsets do not overlap each other or existing chunks.
3. If renaming the primary root or mirror root chunk, it updates the ONCONFIG file parameters ROOTPATH and ROOTOFFSET, or MIRRORPATH, and MIRROROFFSET. The old version of the ONCONFIG file is saved as **$ONCONFIG.*localtime***.
4. It restores the data from the old chunks to the new chunks (if the new chunks exist).
5. It writes the rename information for each chunk to the online log.

If either of the validation steps fail, the renaming process stops and ON–Bar writes an error message to the ON–Bar activity log.

Warning: Perform a level-0 archive after you rename chunks; otherwise you will have to restore the renamed chunk to its original path name and then rename the chunk again.

Important: If you add a chunk after performing a level-0 archive, that chunk cannot be renamed during a restore. Also, you cannot safely specify that chunk as a new path in the mapping list.

Important: Renaming chunks for database servers participating in HDR involves a significant amount of time offline for both database servers. For more information, see the *IBM Informix Administrator's Guide*.

New-Chunk Requirements

To rename a chunk, follow these guidelines for new chunks:

- The new chunk does not need to exist

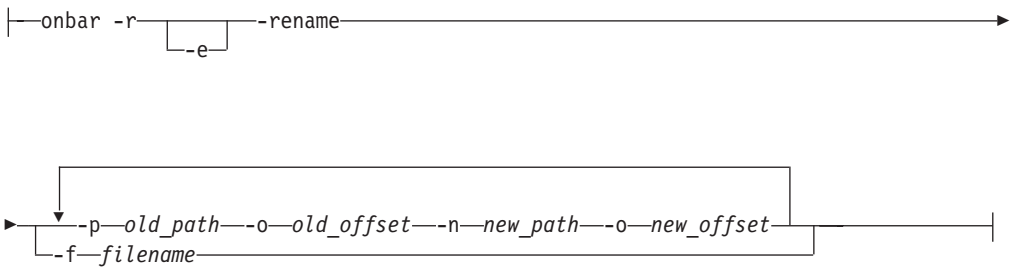
  You can install the new chunk later and perform a warm restore of a storage space containing it. If you specify a nonexistent chunk, ON–Bar records the rename information in the chunk reserved pages, but does not restore the data. The renamed (but not restored) chunks have a status of offline, designated by `N`, in the **onstat -d** chunk status command output.
- New chunks must have the proper permissions.

  Rename operations fail unless the chunks have the proper permissions. For more information, see the *IBM Informix Administrator's Guide*.

Syntax

This diagram shows the ON–Bar syntax for renaming chunks during a cold restore.

Renaming Chunks:



Element	Purpose	Key Considerations
onbar -r	Specifies a restore	You must specify the **-r** parameter first.
-rename	Renames one or more chunks during a cold restore	
-e	Specifies an external restore.	You can rename chunks during an external cold restore.
-f *filename*	Specifies a file containing the names and offsets of chunks to be renamed and their new locations Use to rename a large number of chunks at one time	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile2**), and absolute (**/usr/informix/backup_lists/listfile3** or **c:\informix\backup_lists\listfile3**) filenames. In the file, list the old chunk path name, the old offset, the new chunk path name, and the new offset. Put a blank space or a tab between each item. Put information for each chunk on a separate line. Blank lines are ignored. Begin comment lines with a # symbol.
-p *old_path* **-o** *old_offset* **-n** *new_path* **-o** *new_offset*	Specifies the chunk to be renamed and its new location Use to rename one or more chunks at one time	The variables for this element are: • *old_path* is the current path and filename of the chunk • *old_offset* is the current offset of the chunk, in kilobytes • *new_path* is the new path and filename of the chunk • *new_offset* is the new offset of the chunk

You can use the following options after the **-rename** command:

- **-e**
- **-f** *filename*
- *dbspace_list*
- **-t** *time*
- **-n** *log*
- **-w**
- **-p**

For more information on these options, see "Performing a Complete Restore" on page 7-8.

Examples of Renaming Chunks During a Restore

To rename a chunk, provide the old chunk location and the new chunk location, either at the command line or in a file.

The following table lists example values for two chunks that are used in the examples in this section.

Element	Value for First Chunk	Value for Second Chunk
old path	/chunk1	/chunk2
old offset	0	10000
new path	/chunk1N	/chunk2N
new offset	20000	0

Renaming Chunks with Command Line Options

To rename the chunks by supplying information on the command line, use this command:

```
onbar -r -rename -p /chunk1 -o 0 -n /chunk1N -o 20000
         -rename -p /chunk2 -o 10000 -n /chunk2N -o 0
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming Chunks with a File

To rename the chunks by supplying a file named **listfile**, use this command:

```
onbar -r -rename -f listfile
```

The contents of the **listfile** file are:

```
/chunk1 0 /chunk1N 20000
/chunk2 10000 /chunk2N 0
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming Chunks While Specifying Other Options

To rename the chunks using command-line options while performing a physical restore on **dbspace1** and **dbspace2**, where **rootdbs** is the name of the rootdbs, use the following command:

```
onbar -r -rename -p /chunk1 -o 0 -n /chunk1N -o 20000
         -rename -p /chunk2 -o 10000 -n /chunk2N -o 0
         -p rootdbs dbspace1 dbspace2
```

Alternatively, to rename the chunks using file while performing a physical restore on **dbspace1** and **dbspace2**, use the following command:

```
onbar -r -rename -f listfile -p rootdbs dbspace1 dbspace2
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming a Chunk to a Nonexistent Device

To rename a chunk to a device that does not yet exist, you specify the new path name, but you do not restore its storage spaces until after you install the physical device. This option is useful if you need to rename a chunk and it is convenient to perform a cold restore before you install the new device. When the new chunk device is ready, you can perform a warm restore of a storage space onto it.

You can combine renaming chunks with existing devices and renaming chunks with nonexistent devices in the same rename operation. This example shows how to rename a single chunk to a nonexistent device name.

The following table lists example values for the chunks used in this example.

Storage Space	Old Chunk Path	Old Offset	New Chunk Path	New Offset
sbspace1	/chunk3	0	/chunk3N	0

To rename a chunk to a nonexistent device:

1. Rename the chunk:

   ```
   onbar -r -rename -p /chunk3 -o 0 -n /chunk3N -o 0
   ```

2. When the following prompt appears, enter `y` to continue:

   ```
   The chunk /chunk3N does not exist. If you continue, the
   restore may fail later for the dbspace which contains this chunk.
   Continue without creating this chunk? (y/n)
   ```

   The chunk **/chunk3** is renamed to **/chunk3N**, but the data has not yet been restored to **/chunk3N**.

3. Perform a level-0 archive.
4. Add the physical device for **/chunk3N**.
5. Perform a warm restore of **sbspace1**:

   ```
   onbar -r sbspace1
   ```

6. Perform a level-0 archive.

Transferring Data with the Imported Restore

With the *imported restore* feature, you can transfer all the data from one instance of the database server to the same instance on a foreign host. For example, you can back up data on one computer and restore the data on a different computer. You can perform imported restores using whole-system, parallel, or serial backups.

The imported restore is useful in the following situations:

- Disaster recovery
- Database server upgrade
- Initialization of High-Availability Data Replication (HDR)

When you prepare for an imported restore, consider these points:

- Make sure that your storage manager supports imported restores.
- The whole-system backup must include all storage spaces; logical logs are optional.

  The parallel backup must include all storage spaces and logical logs.
- You can change the database server name in an imported restore.

Important: You cannot use a backup from one database server version to restore on a different version.

For information on importing a restore with ISM, see the *IBM Informix Storage Manager Administrator's Guide*. For information on using HDR, see the *IBM Informix Administrator's Guide*. If you are using a third-party storage manager, use the following procedure for an imported restore.

To set up an imported restore:

1. Install the database server and the storage manager on the target computer. Both computers must have the following:
   - Identical hardware and operating systems
   - Identical database server versions
   - Be on the same LAN or WAN
   - Identical storage-manager versions
   - Compatible XBSA libraries

   The source computer (also called the *primary server*) contains the current instance that you want to replicate. The target computer (also called the *secondary server*) contains the computer where you want to replicate the source instance.
2. Set up the storage manager on the target database server instance.
   a. Define the same type of storage devices as on the source instance.
   b. Label the storage media with the correct pool names.
   c. Mount the storage devices.
   d. Update the **sm_versions** file on the target computer with the storage-manager version.

Importing a Restore

When importing a restore, you must back up your data and migrate the storage manager objects to the target server, do a physcial restore and a logical restore.

Before you back up the data, set the storage-manager environment variables. See "Customizing ON-Bar and Storage-Manager Commands" on page 9-2 for more information.

To back up the data and migrate the storage-manager objects:

1. Perform a level-0 backup (**ON-Bar -b** or **onbar -b -w**) of all storage spaces on the source database server. (Do not perform an incremental backup.)
2. If you are using ISM, follow these steps:
   a. Shut down the storage manager on both computers.
   b. Create a tar file of the storage-manager directories on the source computer.
   c. Copy this tar file and unpack it on the target computer.

   With other storage managers, you might be able to use backup tapes or import the storage-manager directories over the network. For more information, see your storage-manager documentation.
3. Mount the transferred storage volumes.
   - If the backup files are on disk, copy them from the source computer to the target computer.
   - If the backup is on tape, mount the transferred volumes on the storage devices that are attached to the target computer. Both the source and target computers must use the same type of storage devices such as 8mm tape or disk.
   - Some storage managers support remote backups to a backup server. If the backup is on the backup server, retrieve the backup from that backup server.
4. Use storage-manager commands (such as **nsradmin -c**) to add the source host name as a client on the target computer.

To perform the imported restore:

1. Copy the following files from the source computer to the target computer:
   a. Emergency boot file

      Rename the emergency boot file with the target database server number. For example, rename **ixbar.51** to **ixbar.52**. The emergency boot file needs only the entries from the level-0 backup on the source computer.

      On Dynamic Server, the filename is **ixbar.***servernum*. On Extended Parallel Server, the filename is **bixbar.***servernum.coservernum*.

   b. The **oncfg** files: **oncfg_***servername.servernum*

      ON–Bar needs the **oncfg** file to know what dbspaces to retrieve. Rename the **oncfg** file with the target database server name and number. For example, rename **oncfg_bostonserver.51** to **oncfg_chicagoserver.52**. The filename should match the DBSERVERNAME and SERVERNUM on the target computer.

   c. The ONCONFIG file

      In the ONCONFIG file, update the DBSERVERNAME and SERVERNUM parameters with the target database server name and number.

   d. Storage-manager configuration files, if any

      The storage-manager configuration files might need updating.

2. Restore the data in one of the following ways:
   - If you have never run an IDS instance on the target server, use the onbar -r command to restore the data.

     Dynamic Server

     If you are importing a whole-system backup, you can use the **onbar -r -w -p** command to restore the data.

     End of Dynamic Server

   - If you have run an IDS instance on the target server, restore the data in two stages; this avoids salvaging the logs and any potential corruption of the instance.
     a. Use the **ON-Bar -r -p** command to restore the physical data.
     b. Use the **ON-Bar -r -l** command to restore the logical logs.

3. Before you expire objects on the target computer and the storage manager using **onsmsync**, perform one of the following tasks:
   - Manually edit the emergency boot file **viz ixbar.***servernum* in the **$INFORMIXDIR/etc** directory on the target computer to replace the Informix server name that is used on the source computer with the Informix server name of the target computer.
   - Execute the command **onsmsync -b** as user **informix** on the target computer to regenerate the emergency boot file from the sysutils database only, so that the newly regenerated emergency boot file reflects the server name of the target computer.

   Otherwise, **onsmsync** expires the incorrect objects.

Important: Every chunk (including mirrors) must match exactly in size, location, and offset on the source and target computers for the imported restore to complete.

Initializing High-Availability Data Replication with ON-Bar (IDS)

Follow the steps for the imported restore and then start HDR and perform a physical-only restore on the target computer. Also see "Initializing HDR with an External Backup and Restore (IDS)" on page 8-16.

Important: If you use ON–Bar to perform the backup and restore, **ontape** is required on both database servers. You cannot remove **ontape** from database servers participating in HDR.

To perform the imported restore:

1. Follow the steps in "To set up an imported restore" on page 7-30.
2. On the source computer, add entries into your **sqlhosts** file or registry to recognize the target instance.

   Verify that the source and target database servers can communicate over the network. For more information on **sqlhosts**, see the *IBM Informix Administrator's Guide*.
3. Follow the steps in "To back up the data and migrate the storage-manager objects" on page 7-30.
4. Copy the emergency boot files, ONCONFIG file, and storage manager files from the source computer to the target computer.

To initialize High-Availability Data Replication:

1. To start HDR on the source database server, use the following command:

   ```
   onmode -d primary secondary_dbservername
   ```

   You might see the following messages in the database server message log:

   ```
   19:28:15  DR: new type = primary, secondary server name = solo_724
   19:28:15  DR: Trying to connect to secondary server ...
   19:28:18  DR: Primary server connected
   19:28:18  DR: Receive error
   19:28:18  DR: Failure recovery error (2)
   19:28:19  DR: Turned off on primary server
   19:28:20  Checkpoint Completed: duration was 0 seconds.
   19:28:20  DR: Cannot connect to secondary server
   19:28:31  DR: Primary server connected
   19:28:31  DR: Receive error
   19:28:31  DR: Failure recovery error (2)
   19:28:32  DR: Turned off on primary server
   19:28:33  Checkpoint Completed: duration was 0 seconds.
   19:28:33  DR: Cannot connect to secondary server
   ```
2. Perform a physical-only restore on the target computer.

   ```
   onbar -r -p
   ```

   If you performed a whole-system backup (**onbar -b -w**), you could optionally use **onbar -r -w -p** to restore the storage spaces only.
3. Check the database server message log, ON–Bar activity log, and the storage-manager error log to see whether the restore was successful.
4. To start HDR on the target database server, use the following command:

   ```
   onmode -d secondary primary_dbservername
   ```
5. If the logical logs needed to synchronize the two database servers are still present on the source database server, the target server retrieves them from the source database server.
6. While the database servers are synchronizing, the logical logs are transferred automatically from the source to the target server.

7. If the logical logs are not on the source database server, you are prompted to restore the required logical logs. If the target database server requires a log number that no longer exists because it was overwritten, ON–Bar will need to retrieve that logical log from the backup.

The following **online.log** messages might display while the database servers are synchronizing:

```
19:37:10  DR: Server type incompatible
19:37:23  DR: Server type incompatible
19:37:31  DR: new type = secondary, primary server name = bostonserver
19:37:31  DR: Trying to connect to primary server ...
19:37:36  DR: Secondary server connected
19:37:36  DR: Failure recovery from disk in progress ...
19:37:37  Logical Recovery Started.
19:37:37  Start Logical Recovery - Start Log 11, End Log ?
19:37:37  Starting Log Position - 11 0x629c
19:37:44  Checkpoint Completed:  duration was 0 seconds.
19:37:45  Checkpoint Completed:  duration was 0 seconds.

19:37:47  Checkpoint Completed:  duration was 0 seconds.
19:37:48  DR: Secondary server operational
19:37:49  Checkpoint Completed:  duration was 0 seconds.
```

Restoring Nonlogging Databases and Tables

Warning: If you do not use logging for your databases or tables, ON–Bar can only restore the data up to the time it was most recently backed up. Changes made to data since the last standard backup are not restorable. If you do not use logging, you would need to redo lost transactions manually.

Dynamic Server

If logical-log backups are disabled because LTAPEDEV is set to **/dev/null** or **NUL**, you can restore only whole-system backups.

End of Dynamic Server

Extended Parallel Server

If logical-log backups are disabled because LOG_BACKUP_MODE is set to NONE, restores are not possible. Extended Parallel Server does not support nonlogging databases. If a parallel archive is taken of a non logged database, then upon restoring that database inconsistencies can occur. To guarantee consistency a whole system backup is required.

End of Extended Parallel Server

Warning: It is strongly recommended that you do not set LTAPEDEV to **/dev/null** or NUL, or LOG_BACKUP_MODE to NONE.

Restoring Table Types

Table 7-1 discusses restore scenarios for different table types. For more information about the table types, see the *IBM Informix Administrator's Guide* and the *IBM Informix Guide to SQL: Syntax*.

Table 7-1. Restoring Table Types

Table Type	IDS	XPS	Can You Restore This Type of Table?
Standard	X	X	Yes. Warm restore, cold restore, and point-in-time restore work.
Duplicated		X	Yes. Only the primary copy of the table is restored. Following a restore, manually drop all duplicates using the SQL command DROP DUPLICATE OF TABLE. You can then duplicate the table again by using the SQL command CREATE DUPLICATE OF TABLE. For more information, see the *IBM Informix Guide to SQL: Syntax*.
Temp	X	X	No.
Scratch		X	No.
Operational		X	You can restore an operational table if no light appends occurred since the last level-0 backup. If light appends occur to the table since the last backup, the table is not wholly restorable. This sort of problem can also occur if you restore from an older backup. To determine whether your table was restored to its original state, check the message log for the following error message: `Portions of partition` *`partnum`* `of table` *`tablename`* `in database` *`dbname`* `were not logged. This partition cannot be rolled forward.` If you see this message, the table or table fragments were not restored to their original state. If you want full access to whatever remains in this table, you need to alter the table to raw and then to the desired table type. This alter operation removes inconsistencies in the table that resulted from replaying non-logged operations such as light appends.
Raw	X	X	When you restore a raw table, it contains only data that was on disk at the time of the last backup. Because raw tables are not logged, changes that occurred since the last backup cannot be restored.
Static		X	Yes, you can restore the data present at the last dbslice or dbspace backup. Static tables cannot change data. If you alter a static table to another type and update the data, the recoverability of the table depends on each type the table has been since each dbspace was backed up. For example, if you alter a static table to raw, it follows the rules for restoring a raw table.

Restoring Tables with Large Objects

ON–Bar supports table-level restores of smart large objects and binary large objects.

Smart Large Objects

For Dynamic Server, table-level restore also supports smart large objects for physical restore only (restore from level-0 archive).

The storage location of the smart large object columns being restored must be specified with the PUT clause in the CREATE TABLE statement. The restored smart large objects are created with the create-time flags LO_NOLOG and LO_NOKEEP_LASTACCESS_TIME. These flags override the LOG and KEEP ACCESS TIME column attributes if they are specified in the target table for the smart large object column.

BLOBs

Table-level restore supports restoring tablespace BLOBs, but not blobspace BLOBs. If you attempt to restore a blobspace BLOB, the value is set to NULL and a warning is issued.

Using Restartable Restore to Recover Data (IDS)

If a failure occurs with the database server, media, or ON–Bar during a restore, you can restart the restore from the place that it failed. By default, the RESTARTABLE_RESTORE parameter is ON. If it is OFF, you must shut down and restart the database server before you begin the original restore. To restart a failed warm or cold restore, issue the **onbar -**RESTART command. All restarted restores resume where the last restore failed.

Important: Set RESTARTABLE_RESTORE to ON if your system is large or unstable. If your system is small, consider turning off restartable restore for faster restore performance only if you have the time to repeat a failed restore from the beginning.

If the failure occurred during a physical restore, ON–Bar restarts the restore at the storage space and level where the failure occurred. It does not matter whether the restore was warm or cold.

If a failure occurred during a cold logical restore, ON–Bar restarts the logical restore from the most recent log checkpoint. Restartable logical restore is supported for cold restores only. However, if the failure during a warm restore caused the database server to shut down, do *not* restart the restore. Instead, use the **archecker** utility to verify the backup and start the whole restore from the beginning.

Warning: Restartable restore does not work for the logical part of a warm restore.

Restartable Restore Example

The following example shows how to use restartable restore for a cold restore:

1. Make sure that RESTARTABLE_RESTORE is `ON`.

   If you just set RESTARTABLE_RESTORE to `ON`, shut down and restart the database server for the changes to take effect.
2. Restore several storage spaces:

   ```
   onbar -r rootdbs dbs1 dbs2 dbs3 dbs4
   ```

   The database server fails while restoring **dbs3**.
3. Restart the restore:

   ```
   onbar -RESTART
   ```

   ON–Bar automatically starts restoring **dbs3**, **dbs4**, and the logical logs.
4. If necessary, bring the database server online:

   ```
   onmode -m
   ```

Important: If a restore fails with RESTARTABLE_RESTORE set to OFF, the **onbar -**RESTART option will not work. Use the **onbar -r** command to repeat the restore from the beginning.

Restarting a Restore

You can restart a point-in-time, whole-system, or parallel restore. The physical restore restarts at the storage space and level where the failure occurred. If the restore failed while some, but not all, chunks of a storage space were restored,

even a restarted restore must restore all chunks of that storage space again. If storage spaces and incremental backups are restored successfully before the failure, they are not restored again.

Figure 7-1 shows how a restartable restore works when the restore failed during a physical restore of **dbspace2**. For example, you set RESTARTABLE_RESTORE to ON before you begin the restore. The level-0, level-1, and level-2 backups of **rootdbs**, and the level-0 and level-1 backups of **dbspace1** and **dbspace2** are successfully restored. The database server fails while restoring the level-1 backup of **dbspace2**. When you restart the restore, ON–Bar restores the level-2 backup of **dbspace 1**, the level-1 and level-2 backups of **dbspace2**, and the logical logs.

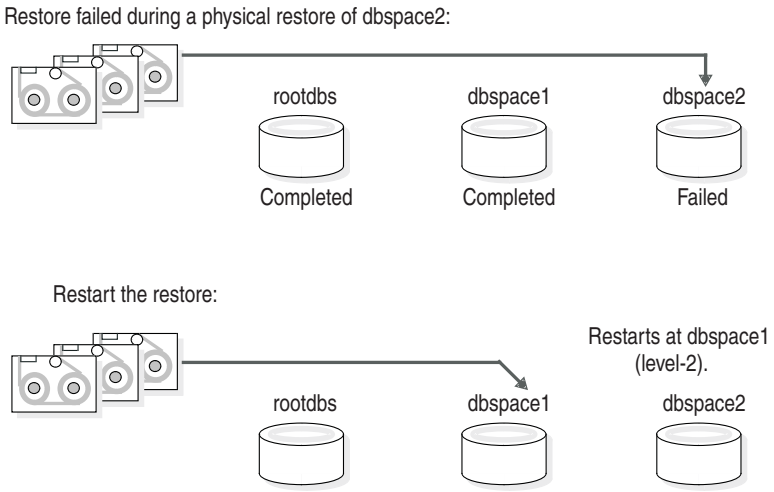


Figure 7-1. Restartable Physical Restore

Interaction Between Restartable Restore and BAR_RETRY Value

If BAR_RETRY > 1, ON–Bar automatically retries the failed storage space or logical log. If this retry is successful, the restore continues and no restart is needed.

If BAR_RETRY = 0 or 1, ON–Bar does not retry the failed storage space or logical log. If the database server is still running, ON–Bar skips the failed storage space and attempts to restore the remaining storage spaces.

Table 7-2 shows what to expect with different values for BAR_RETRY in a different restarted restore example.

Table 7-2. Restartable Restore Results with Different BAR_RETRY Values

ON–Bar Command	BAR_RETRY = 2	BAR_RETRY = 0
onbar -r dbs1 dbs2 dbs3	restore level-0 dbs1, dbs2, dbs3 restore level-1 dbs1 FAILS restore level-1 dbs1 RETRY PASSES restore level-1 dbs2, dbs3 restore level-2 dbs1, dbs2, dbs3 restore logical logs	restore level-0 dbs1, dbs2, dbs3 restore level-1 dbs1 FAILS
onbar -RESTART	No restart is needed because everything was successfully restored.	restore level-1 dbs1, dbs2, dbs3 restore level-2 dbs1, dbs2, dbs3 restore logical logs
onbar -r dbs1 dbs2 dbs3	restore level-0 dbs1, dbs2, dbs3 restore level-1 dbs1 FAILS restore level-1 dbs1 RETRY FAILS restore level-1 dbs2, dbs3 restore level-2 dbs2, dbs3 restore logical logs **onbar -r dbs1 dbs2** restore level-1 dbs1 restore level-2 dbs1 restore logical logs	restore level-0 dbs1, dbs2, dbs3 restore level-1 dbs1 FAILS **onbar -RESTART** restore level-1 dbs1, dbs2, dbs3 restore level-2 dbs1, dbs2, dbs3 restore logical logs

Restarting a Logical Restore

If a restore fails during the logical phase and you restart the restore, ON–Bar verifies that the storage spaces have been restored successfully, skips the physical restore, and restarts the logical restore. Figure 7-2 on page 7-38 shows a cold restore that failed while restoring logical log LL-3. When you restart the cold logical restore, log replay starts from the last restored checkpoint. In this example, the last checkpoint is in logical log LL-2.

If a failure occurs during a cold logical restore, ON–Bar restarts it at the place that it failed.

Important: If a failure occurs during a warm logical restore, you have to restart it from the beginning. If the database server is still running, use the **onbar -r -l** command to complete the restore.

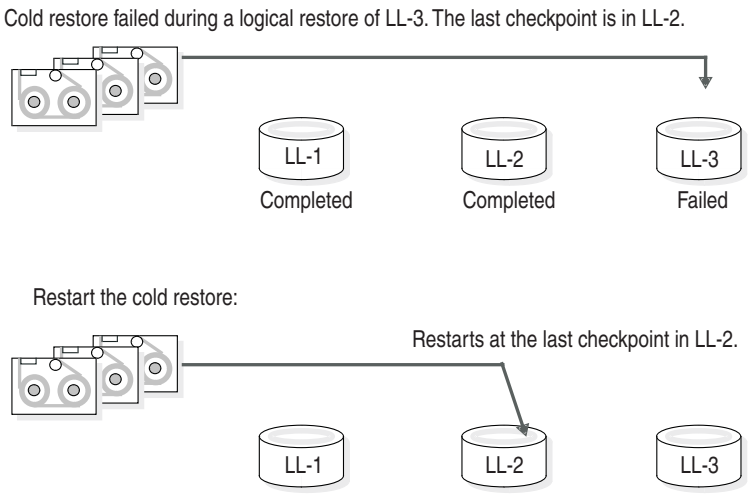


Figure 7-2. Restartable Cold Logical Restore

Set the RESTARTABLE_RESTORE parameter to `ON`. A restartable restore makes the logical restore run more slowly if many logical logs need to be restored, but it saves you time if something goes wrong and you need to restart. Restartable restore does not affect the speed of the physical restore.

Resolving a Failed Restore

What is retried, what is restartable, and what command you use to restart the restore depends on what failed and how serious it was. You can save some failed restores even if restartable restore is turned off. For example, if the restore fails because of a storage-manager or storage-device error, you can fix the tape drive or storage-manager problem, remount a tape, and then continue the restore.

Table 7-3 shows what results to expect when physical restore fails. Assume that BAR_RETRY > 1 in each case.

Table 7-3. Failed Physical Restore Scenarios

Type of Error	RESTARTABLE_ RESTORE Setting	What to Do When the Physical Restore Fails?
Database server, ON–Bar, or storage-manager error (database server is still running)	ON or OFF	ON–Bar retries each failed restore. If the storage manager failed, fix the storage-manager error. If the retried restore fails, issue **onbar -r** *spaces* where *spaces* is the list of storage spaces not yet restored. Use **onstat -d** to obtain the list of storage spaces that need to be restored. ON–Bar restores the level-0 backup of each storage space, then the level-1 and level-2 backups, if any.
ON–Bar or storage-manager error (database server is still running)	ON	Issue the **onbar -RESTART** command. If the storage manager failed, fix the storage-manager error. The restore restarts at the storage space and backup level where the first restore failed. If the level-0 backup of a storage space was successfully restored, the restarted restore skips the level-0 backup and restores the level-1 and level-2 backups, if any.
Database server failure	ON or OFF	Because the database server is down, perform a cold restore. Use **onbar -r** to restore the critical dbspaces and any noncritical spaces that were not restored the first time.
Database server failure	ON	Issue the **onbar -RESTART** command. The restore restarts at the storage space and backup level where the first restore failed. If the level-0 backup of a storage space was successfully restored, the restarted restore skips the level-0 backup and restores the level-1 and level-2 backups, if any.

Table 7-4 shows what results to expect when logical restore fails.

Table 7-4. Failed Logical Restore Scenarios

Type of Error	RESTARTABLE_ RESTORE Setting	What to Do When a Logical Restore Fails?
Database server or ON–Bar error in a cold restore (database server is still running)	ON	Issue the **onbar -RESTART** command. The logical restore restarts at the last checkpoint. If this restore fails, shut down and restart the database server to initiate fast recovery of the logical logs. All logical logs not restored are lost.
Database server or ON–Bar error (database server is still running)	ON or OFF	Issue the **onbar -r -l** command. The restore should restart at the failed logical log. If **onbar -r -l** still fails, shut down and restart the database server. The database server will complete fast recovery. All logical logs that were not restored are lost. If fast recovery does not work, you have to do a cold restore.
Database server error	ON	If the cold logical restore failed, issue **onbar -RESTART**. If the warm logical restore failed, issue the **onbar -r -l** command. If that fails, restart the entire restore from the beginning.
Storage-manager error (IDS)	ON or OFF	ON–Bar retries each failed logical restore. If the retried restore fails, the logical restore is suspended. Fix the storage-manager error. Then issue the **onbar -r -l** command. The restore should restart at the failed logical log.

Understanding ON-Bar Restore Processes

This section explains how ON–Bar performs restore operations on the database server. If the database server is in quiescent mode or is online, you can perform a warm restore. ON–Bar gathers storage-space and logical-log backup data from the **sysutils** database and then requests a restore from the database server.

If you have lost critical dbspaces, you must perform a cold restore. ON–Bar gathers backup data from the emergency boot file and then restores the storage spaces and logical logs.

Warm-Restore Sequence on Dynamic Server

Figure 7-3 on page 7-41 describes the ON–Bar warm-restore sequence.

In a warm restore, the **onbar-driver** creates a list of restore objects. In a parallel restore (if BAR_MAX_BACKUP is not set to 1), the ON–Bar driver starts **onbar_d** child processes. The **onbar_d** processes transfer data between the storage manager and the database server until the warm restore is complete. Each **onbar_d** process processes one storage space. In a serial restore, the **onbar-driver** restores the storage spaces one at a time. Then the **onbar-driver** performs the logical backup and restore. After each object is restored, information about it is added to the **sysutils** database.

For each storage space, ON–Bar restores the last level-0 backup, the level-1 backup (if it exists), and the level-2 backup (if it exists). After the physical restore is complete, ON–Bar backs up the logical logs to get the latest checkpoint and then restores the logical logs. This logical backup allows data to be restored up to the

moment of failure.

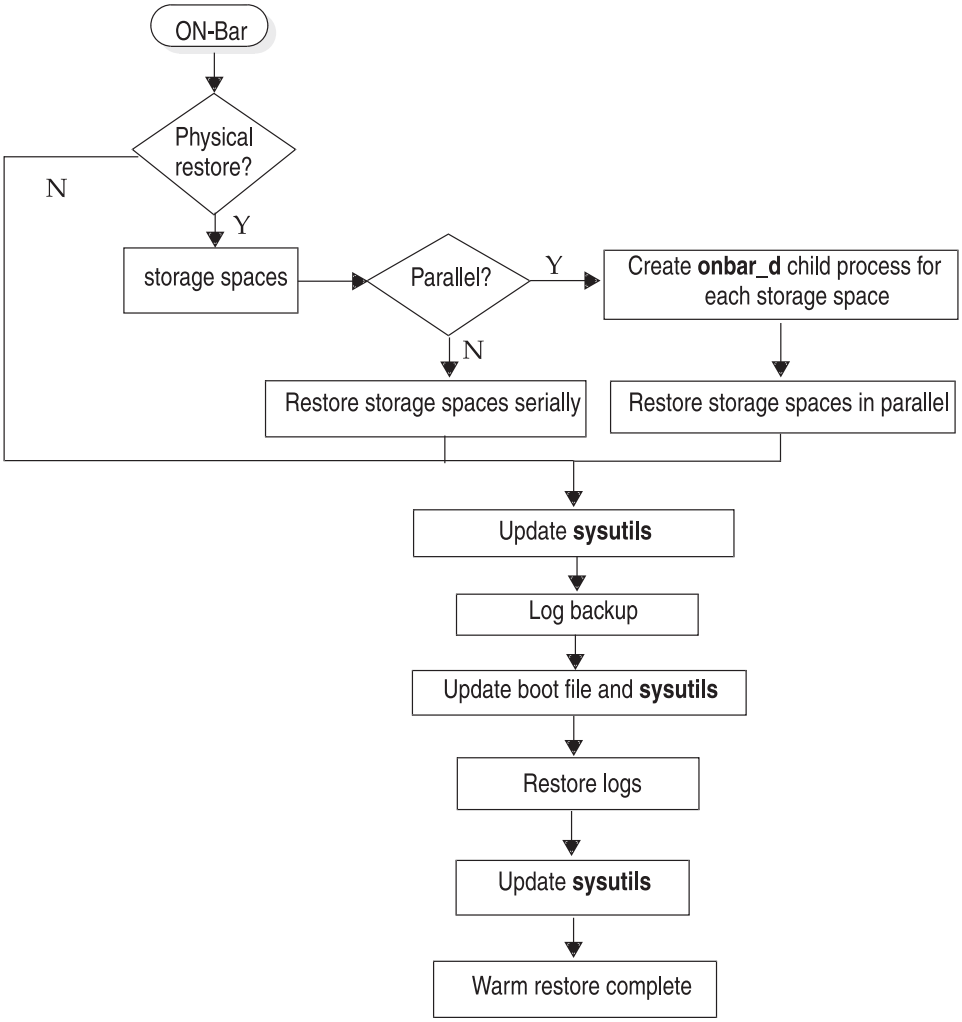


Figure 7-3. ON-Bar Warm-Restore Sequence on Dynamic Server

Cold-Restore Sequence on Dynamic Server

Figure 7-4 on page 7-42 describes the ON–Bar cold-restore sequence. ON–Bar uses the backup emergency boot file to determine what restores are required.

In a cold restore, ON–Bar performs the following steps in order:

- Salvages the logical logs
- Restores the root dbspace
- Restores the critical dbspaces
- Restores blobspaces
- Restores noncritical dbspaces and sbspaces
- Restores logical logs

For each storage space, ON–Bar restores the last level-0 backup, the level-1 backup (if it exists), and the level-2 backup (if it exists). Finally, ON–Bar restores the logical logs.

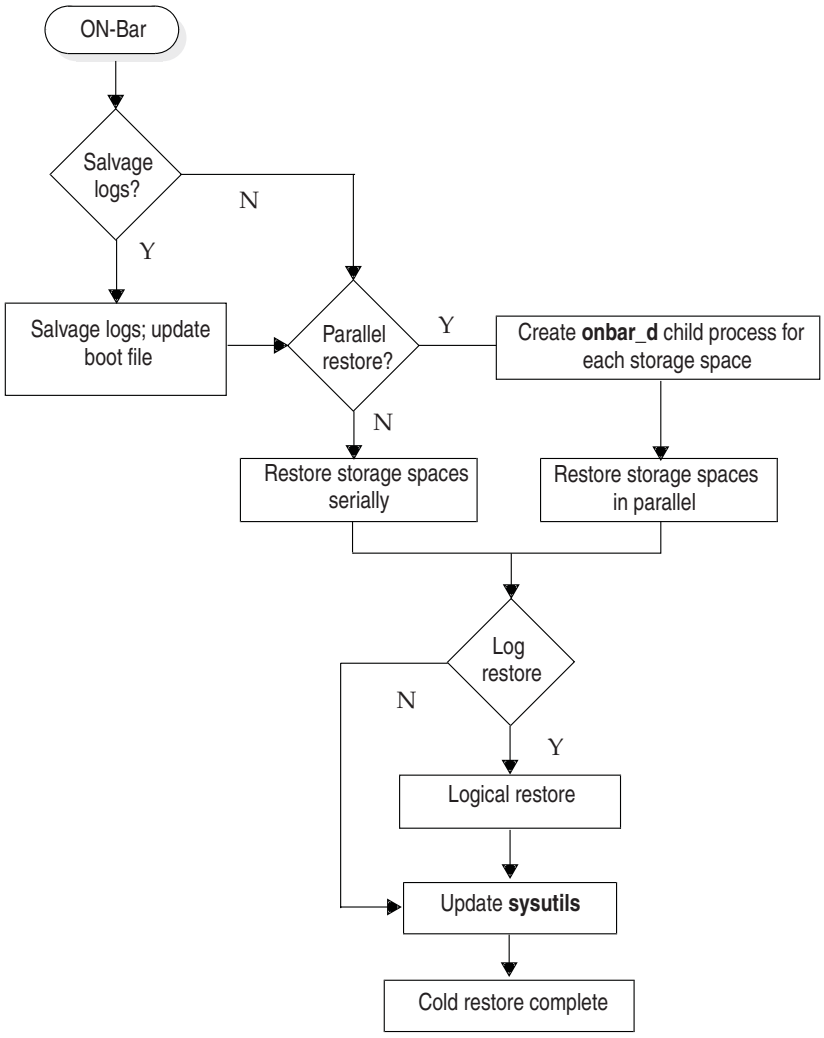


Figure 7-4. ON-Bar Cold-Restore Sequence on Dynamic Server

Warm-Restore Sequence on Extended Parallel Server

Figure 7-5 on page 7-43 describes the ON–Bar warm-restore sequence.

In a warm restore, the **onbar-driver** sends a list of backup objects to the Backup Scheduler. The Backup Scheduler creates a restore session that contains lists of backup objects to restore and might start one or more **onbar-worker** processes. The **onbar-worker** transfers data between the storage manager and the database server until the warm restore is complete. For each storage space, ON–Bar restores the last level-0 backup, the level-1 backup (if it exists), and the level-2 backup (if it exists). Next, ON–Bar backs up the logical logs to get the latest checkpoint and then restores them. This logical-log backup allows data to be restored as close to the moment of failure as possible.

As each object is restored, information about the restore is added to the **sysutils** database. As each logical log is backed up, information about it is added to **sysutils** and the emergency boot file. The emergency backup boot file is on the coserver of the **onbar-worker** that backed it up.

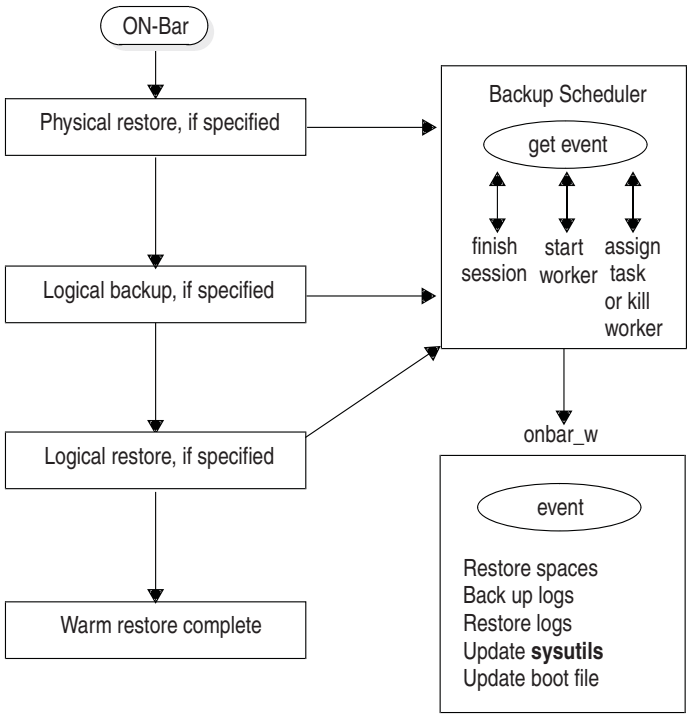


Figure 7-5. ON-Bar Warm-Restore Sequence on Extended Parallel Server

Cold-Restore Sequence on Extended Parallel Server

You must put the database server in microkernel mode to do a cold restore. If the database server or a coserver is offline, you cannot perform any restores. Figure 7-6 on page 7-44 describes the ON–Bar cold-restore sequence.

In a cold restore, ON–Bar performs the following steps in order:

- Salvages the logical logs
- Merges the boot files
- Restores the root dbspace
- Restores the critical dbspaces
- Restores the other dbspaces
- Restores logical logs

The **onbar-merger** utility collects and processes the backup emergency boot files from each coserver to determine what restores are required. The **onbar-merger** then creates the restore boot file and copies it to each coserver that contains a backup emergency boot file.

You can specify in the BAR_WORKER_COSERVER configuration parameter which coservers have boot files and run **onbar-worker** processes. For more information, see "BAR_WORKER_COSVR (XPS)" on page 10-16.

For each storage space, ON–Bar restores the last level-0 backup, the level-1 backup (if it exists), and the level-2 backup (if it exists). Finally, it restores the logical logs.

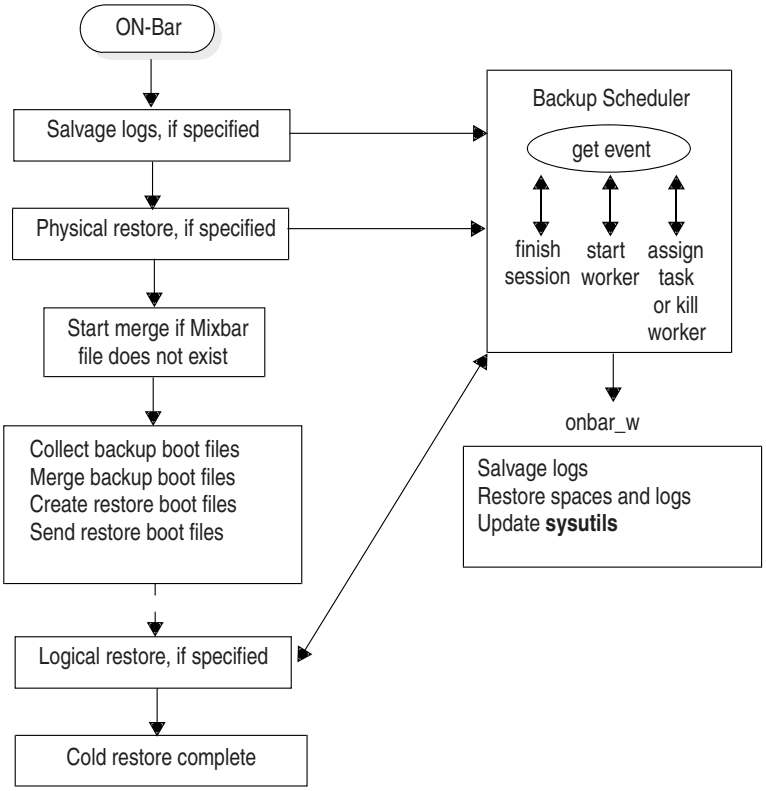


Figure 7-6. ON-Bar Cold-Restore Sequence on Extended Parallel Server

Chapter 8. Performing an External Backup and Restore

In This Chapter 8-1
External Backup and Restore Overview 8-1
Blocking Before Backing Up 8-2
Rules for an External Backup 8-2
Preparing for an External Backup 8-3
Blocking and Unblocking Dynamic Server 8-3
Blocking and Unblocking Extended Parallel Server 8-4
Monitoring an External Backup 8-7
Tracking an External Backup 8-7
Performing an External Backup 8-7
Performing an External Backup when Chunks are Mirrored by the Database Server (XPS) 8-9
Performing an External Backup Using Third-Party Mirroring Solutions (XPS) 8-10
Data Restored in an External Restore 8-10
Renaming Chunks 8-11
Using External Restore Commands 8-11
Rules for an External Restore 8-12
Performing an External Restore 8-13
Cold External Restore Procedure 8-13
Mixed External Restore Restriction 8-15
Warm External Restore Procedure 8-15
Examples of External Restore Commands 8-15
Initializing HDR with an External Backup and Restore (IDS) 8-16

In This Chapter

This chapter discusses recovering data using external backup and restore.

External Backup and Restore Overview

An external backup and restore eliminates the downtime of systems because the backup and restore operations are performed external to the Informix system. ON–Bar does not move the data during the backup or physical restore. An external backup allows you to copy disks that contain storage-space chunks without using ON–Bar. When disks fail, replace them and use third-party software to restore the data, then use ON–Bar for the logical restore. For more information, see “Data Restored in an External Restore” on page 8-10.

The following are typical scenarios for external backup and restore:

- *Availability with disk mirroring.* If you use hardware disk mirroring, you can get your system online faster with external backup and restore than with conventional ON–Bar commands.
- *Cloning.* You can use external backup and restore to clone an existing production system for testing or migration without disturbing the production system.

Figure 8-1 shows how to perform a backup using mirroring.

© Copyright IBM Corp. 1996, 2008

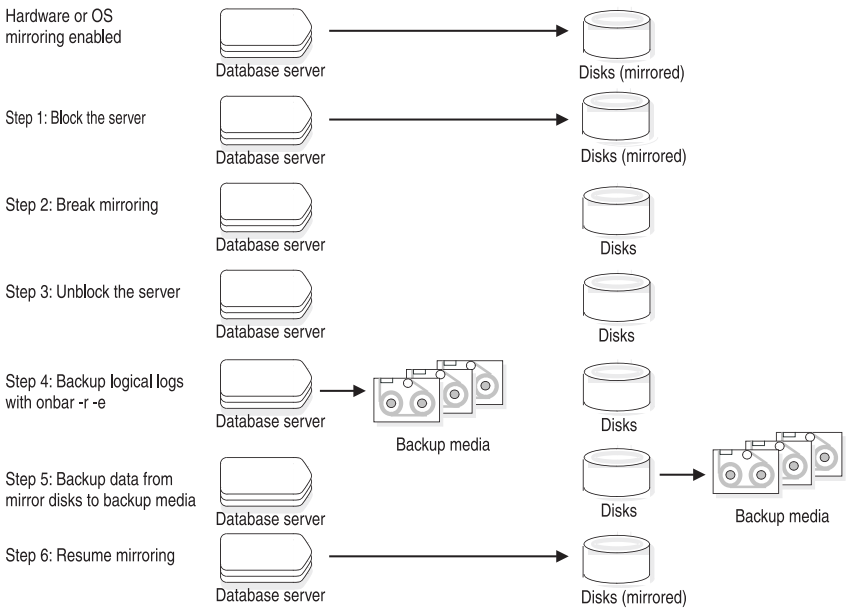


Figure 8-1. Performing a Backup with Mirroring

In this configuration, the database server is running continuously, except for the short time when the database server is blocked to break the mirror. The mirrored disks contain a copy of the database server storage spaces. To create a backup, block the database server to stop transactions and disable mirroring. The mirrored disks now contain a copy of the consistent data at a specific point in time. After disabling mirroring, unblock the database server to allow transactions to resume and then backup the logical logs. Copy the data from the offline mirrored disks to backup media using external commands. Now you can resume mirroring.

Blocking Before Backing Up

Before you begin an external backup, block the database server or one or more coservers. Blocking forces a checkpoint, flushes buffers to disk, and blocks user transactions that involve temporary tables. During the blocking operation, users can access that database server or coserver in read-only mode. Then you can physically back up or copy the data to another set of disks or storage media using operating-system or third-party tools. When you complete the external backup, unblock the database server so that transactions can resume. You should include all the chunk files in each storage space, administrative files, such as **ONCONFIG**, and the emergency boot file, in an external backup.

Important: To make tracking backups easier, it is recommended that you back up all storage spaces in each external backup.

ON–Bar treats an external backup as equivalent to a level-0 backup. You cannot perform an external backup and then use ON–Bar to perform a level-1 backup, or vice versa because ON–Bar does not have any record of the external backup. For more information, see “Performing an External Backup” on page 8-7.

Rules for an External Backup

Before you begin an external backup, keep the following rules in mind:

- The database server must be online or in quiescent mode during an external backup.

- Use ON–Bar to back up all logical logs including the current log so that you can restore the logical logs at the end of the external restore.
- Suspend continuous logical-log backups before you block the database server for an external backup. After the external backup is complete, resume the continuous logical-log backup.

Extended Parallel Server

To suspend continuous logical-log backup, use the **onbar off Log backup** ***coserver_id*** command. To resume continuous logical-log backup, use the **onbar on Log backup** ***coserver_id*** command. For more information, see "Starting and Stopping ON-Bar Sessions (XPS)" on page 9-10.

End of Extended Parallel Server

Dynamic Server

To stop continuous logical-log backup, use the CTRL-C command. To resume continuous logical-log backup, use the **onbar -b -l -C** command.

End of Dynamic Server

- Wait until all ON–Bar backup sessions have completed before you block the database server. If any backup sessions are active, the block command displays an error message.
- Any OLTP work or queries are suspended while the database server is blocked. They resume after the database server is unblocked.
- All critical dbspaces of the database server instance must be backed up together simultaneously within the same command bracket of **onmode -c block ... onmode -c unblock**. Backups of different critical dbspaces done at different times cannot be restored to a consistent system.

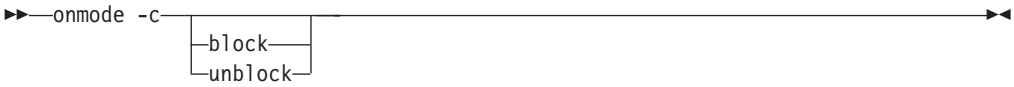
Important: Because the external backup is outside the control of ON–Bar, you must track these backups manually. For more information, see "Tracking an External Backup" on page 8-7.

Preparing for an External Backup

This section describes the commands used to prepare for an external backup. For the procedure, see "Performing an External Backup" on page 8-7.

Blocking and Unblocking Dynamic Server

This section shows the syntax of the block and unblock commands on Dynamic Server.



Element	Purpose	Key Considerations
onmode -c	Performs a checkpoint and blocks or unblocks the database server	None.
block	Blocks the database server from any transactions	Sets up the database server for an external backup. While the database server is blocked, users can access it in read-only mode. Sample command: `onmode -c block`
unblock	Unblocks the database server, allowing data transactions and normal database server operations to resume	Do not unblock until the external backup is finished. Sample command: `onmode -c unblock`

Blocking and Unblocking Extended Parallel Server

This section shows the syntax of the block and unblock commands on Extended Parallel Server.

Blocking Coservers or the Database Server: Use the **onutil EBR BLOCK** command to block the database server or coservers. The EBR, BLOCK, and SESSION keywords can be uppercase or lowercase. Before you begin an external backup, block the database server or coserver. You can leave the coservers blocked for as long as you decide.

Element	Purpose	Key Considerations
BLOCK	Forces a checkpoint that flushes the buffers to disk and blocks the database server or specified coservers from any transactions	While a coserver or database server is blocked, users can access it in read-only mode.
cogroup	Specifies the set of coservers to block You can block one or more listed coservers or cogroups, or **cogroup_all**. If you specify coserver names, use the format: ***dbservername.coserverid***. If you specify a cogroup range identifier, use the format: ***dbservername.%r(first.last)***.	If no cogroup or coserver name is specified, all coservers on the database server are blocked.
EBR	Specifies an external backup or restore command	None.
SESSION *session_name*	Assigns a session name to the **onutil EBR BLOCK** command Use the session name to monitor the external backup status.	If no session name is specified, **onutil** generates a session name of the following format: **ebr**_*connection-coserver-id.client-sql-session-id.ebr-command-number*.

When a coserver is blocked, all the data on disk is synchronized and users can access that coserver in read-only mode. When you block a coserver, the entire coserver is blocked. You can block a list of coservers, cogroups, or the entire database server. You can assign a session name to the blocking operation.

To block the entire database server (all the coservers in **cogroup_all**), use either of the following commands:

```
onutil ebr block;
```

or

```
onutil ebr block cogroup_all;
```

When the block command completes, the default session ID is printed to standard output and a message is written to **online.log**.

After you complete the external backup, unblock the database server. For the syntax diagram, see "Unblocking Coservers or the Database Server" on page 8-5.

Blocking a Cogroup and Specifying a Session Name: Specifying a session name is useful for monitoring external backup sessions. To block the coservers defined in cogroup **data_cogroup** and to name the Backup Scheduler for that cogroup as **block_data_cogroup**, use the following command:

```
onutil ebr block data_cogroup session block_data_cogroup;
```

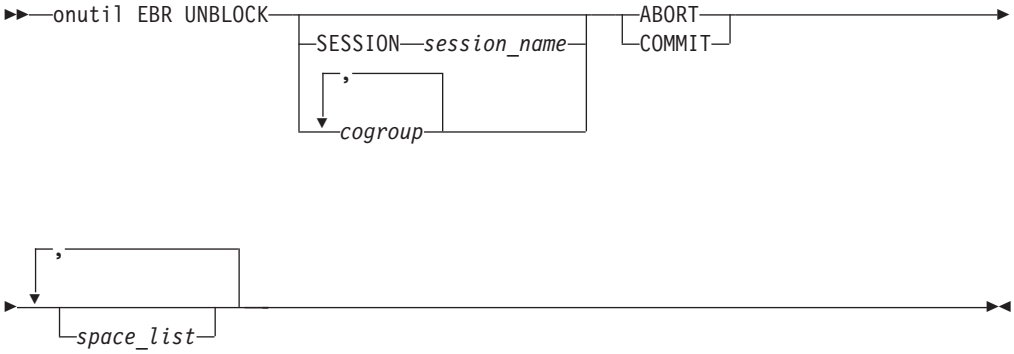
For information on how to create cogroups, see the **onutil** section of the *IBM Informix Administrator's Reference*. For information on how to monitor this session, see "Monitoring an External Backup" on page 8-7.

Blocking Coservers: To block specific coservers, either specify them by name or use the cogroup range identifier with the **onutil** command. For example, **xps.%r(1.4)** expands to coservers **xps.1**, **xps.2**, **xps.3**, and **xps.4**. The following example blocks coservers **xps.1** and **xps.2**.

```
onutil ebr block xps.1, xps.2;
```

For information on how to specify coserver names and cogroup range identifiers, see the **onutil** section of the *IBM Informix Administrator's Reference*.

Unblocking Coservers or the Database Server: After you complete the external backup, use the **onutil EBR UNBLOCK** command to unblock and allow write access for data transactions. The EBR, UNBLOCK, SESSION, ABORT, and COMMIT keywords can be uppercase or lowercase.



Element	Purpose	Key Considerations
ABORT	Unblocks the specified coservers but does not mark the dbspaces as backed up	Use the **ABORT** option when the external backup failed or when you do not want to update the backup status. The dbspaces are not restorable.

Element	Purpose	Key Considerations
cogroup	Specifies a list of one or more cogroup names to unblock the coservers You can specify **cogroup_all**, cogroup names, coserver names, or a cogroup range identifier.	All coservers in the specified cogroups that are currently blocked are unblocked. If you do not specify a cogroup name, all the coservers in the database server are unblocked. If a specific coserver is not blocked, an error is reported.
COMMIT	Marks the listed dbspaces or dbslices as successfully backed up while the set of coservers were blocked Enables newly added logical logs and dbspace mirrors. If no logged updates have occurred since the last backup, marks the dbspace for skipping log replay. To verify, issue **onstat -d**.	If you do not specify any dbspaces or dbslices after the **COMMIT** keyword, it commits the dbspaces and dbslices on the coservers that were blocked.
EBR	Specifies an external backup or restore	None.
SESSION *session_name*	Specifies a session name that can be either a name that you specified or **onutil** automatically created in the block command	The session name must be one that was used in a previous block command. All coservers that were part of that session are now unblocked.
space_list	Specifies the dbspaces or dbslices that are successfully backed up	Separate the dbspace or dbslice names with commas.
UNBLOCK	Unblocks the coservers or database server, allowing data transactions and normal database server operations to resume	Ends the external backup.

To unblock a list of coservers or cogroups, or the entire database server, you can specify a session name. The list of coservers in the **onutil** EBR UNBLOCK command does not have to match the list of coservers in the **onutil** EBR BLOCK command.

To mark selected or all dbspaces and dbslices as backed up on each blocked coserver, use the **COMMIT** option. Use the **ABORT** option when you do not want to mark any dbspaces as backed up.

Tip: Be sure not to unblock coservers that another user has blocked. (For information, see "Monitoring an External Backup" on page 8-7.)

Unblocking All Coservers and Marking Dbspaces as Backed Up: To unblock the entire database server (all the coservers in **cogroup_all**), use either of the following commands. The **COMMIT** option marks all dbspaces on each coserver as having a level-0 backup.

```
onutil ebr unblock commit;
```

or

```
onutil ebr unblock cogroup_all commit;
```

Unblocking a Cogroup and Marking Selected Dbspaces as Backed Up: The following example unblocks the coservers in cogroup **data_group** and marks selected dbspaces or dbslices as backed up. The **COMMIT** option marks dbspaces **rootdbs.1**, **rootdbs.2**, and **rootdbs.3** as backed up.

```
onutil ebr unblock data_group commit rootdbs.%r(1..3);
```

Unblocking a Specific Session and Not Marking Dbspaces as Backed Up: Use the **ABORT** option to cancel a failed external backup and unblock the set of coservers. You cannot externally restore these dbspaces because the backups are incomplete.

If you use the **ABORT** option, the skip logical replay feature does not work, even if that dbspace was completely backed up. In this case, use the **COMMIT** option to commit those dbspaces that were backed up successfully and use the **ABORT** option for the remaining dbspaces.

The following example unblocks the coservers identified in session **block_data_cogroup** and does not mark any dbspaces as backed up. You must have named the session in the previous block command. The **ABORT** option means that no dbspaces are marked as backed up.

```
onutil ebr unblock session block_data_cogroup abort;
```

Monitoring an External Backup

To monitor the external backup status, use the **onstat -g -bus** command. To find blocked coservers, use the **xctl onstat -** command. For more information, see "Using the ***onstat -g bus*** Option" on page 9-11.

To monitor the external backup status in the **sysmaster** database, use the following SQL queries. For more information, see "Backup Scheduler SMI Tables (XPS)" on page 11-8.

```
# to find all blocked coservers
SELECT * FROM sysbuobject
WHERE is_block = 1 AND is_coserver = 1;

# to find all blocked coservers in my session
SELECT * FROM sysbuobjses WHERE os_ses_id = "my_session";
```

Tracking an External Backup

The database server and ON–Bar do *not* track external backups. To track the external backup data, use a third-party storage manager or track the data manually. Table 8-1 shows which items we recommend that you track in an external backup. ON–Bar keeps a limited history of external restores.

Table 8-1. Items to Track When You Use External Backup and Restore

Items to Track	Examples
Full path names of each chunk file for each backed up storage space	**/work/dbspaces/rootdbs** (UNIX) **c:\work\dbspaces\rootdbs** (Windows)
Object type	Critical dbspaces, noncritical storage spaces
ins_copyid_hi and **ins_copyid_lo**	Copy ID that the storage manager assigns to each backup object
Backup date and time	The times that the database server was blocked and unblocked
Backup media	Tape volume number or disk path name
Database server version	The database server version from which the backup was taken.

Performing an External Backup

The database server must be online or in quiescent mode during an external backup.

Performing an External Backup when Chunks are not Mirrored:

1. To obtain an external backup, block the database server. The system takes a checkpoint and suspends all update transactions. Users can access the database server in read-only mode.

   Dynamic Server

   On Dynamic Server, use the following command:

   ```
   onmode -c block
   ```

   End of Dynamic Server

   Extended Parallel Server

   On Extended Parallel Server, use the following command to block all the coservers in **cogroup_all**:

   ```
   onutil ebr block;
   ```

   End of Extended Parallel Server

2. To back up the storage spaces and administrative files, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or **copy** on Windows, or a file-backup program. You must back up all chunks in the storage spaces.
3. To allow normal operations to resume, unblock the database server.

   Dynamic Server

   Use the following command:

   ```
   onmode -c unblock
   ```

   End of Dynamic Server

   Extended Parallel Server

   Use the following command to unblock all the coservers in **cogroup_all** and mark all dbspaces on each coserver as having a level-0 backup:

   ```
   onutil ebr unblock commit;
   ```

   End of Extended Parallel Server

4. Back up all the logical logs including the current log so that checkpoint information is available for the external restore.

   Warning: Because external backup is not done through ON–Bar, you must ensure that you have a backup of the current logical log from the time when you execute the **onutil** *EBR BLOCK* or **onmode -c block** command. Without a backup of this logical-log file, the external backup is not restorable.
5. After you perform an external backup, back up the current log.

   Dynamic Server

   Use the following command to back up the current log on Dynamic Server:

   ```
   onbar -b -l -c
   ```

End of Dynamic Server

Extended Parallel Server

Use the following commands to switch to and back up the current log on Extended Parallel Server:

```
onmode -l   # execute on coservers with external backups
onbar -b -l # back up all used logs
```

End of Extended Parallel Server

If you lose a disk, coserver, or the whole system, you are now ready to perform an external restore.

Performing an External Backup when Chunks are Mirrored by the Database Server (XPS)

When you use the mirroring support provided by the database server, you should create your backups by copying the data from the primary chunks to make tracking your backups easier. Mirroring is configured using the MIRROR configuration parameter in the ONCONFIG file. For more information on mirroring, see the chapter on mirroring in the *IBM Informix Administrator's Guide*.

To perform an external backup when chunks are mirrored:

1. To obtain an external backup, block the database server. The system takes a checkpoint and suspends all update transactions. Users can access the database server in read-only mode.

   On Extended Parallel Server, use the following command to block all the coservers in **cogroup_all**:

   ```
   onutil ebr block;
   ```

2. To back up the storage spaces and administrative files, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or **copy** on Windows, or a file-backup program. You must back up all chunks in the primary storage spaces. Do not backup mirror chunks.
3. To allow normal operations to resume, unblock the database server.

   Use the following command to unblock all the coservers in **cogroup_all** and mark all dbspaces on each coserver as having a level-0 backup:

   ```
   onutil ebr unblock commit;
   ```

4. Back up all the logical logs including the current log so that checkpoint information is available for the external restore.

   Warning: Because external backup is not done through ON–Bar, you must ensure that you have a backup of the current logical log from the time when you execute the **onutil** *EBR BLOCK* or **onmode -c block** command. Without a backup of this logical-log file, the external backup is not restorable.

5. After you perform an external backup, back up the current log.

   Use the following commands to switch to and back up the current log on Extended Parallel Server:

   ```
   onmode -l   # execute on coservers with external backups
   onbar -b -l # back up all used logs
   ```

Performing an External Backup Using Third-Party Mirroring Solutions (XPS)

When you mirror your data using third-party mirroring solutions provided by the hardware or the operating system, it might be possible to backup the chunks in your dbspaces using third-party mirror maintenance commands provided by such mirroring solutions instead of relying on commands such as **cp**, **dd,** or **tar** on UNIX, or **copy** on Windows. This section outlines a procedure for performing external backups using third-party mirror maintenance commands, based on the idea of using your chunk mirrors as your backups.

To perform an external backup using third-party mirroring solutions:

1. Copy the primary partition to the mirror partition.
2. Synchronize the mirrored disk partitions.
3. Block the database server. The system takes a checkpoint and suspends all update transactions.
4. Break the link between the disk mirrors. If needed, install a new set of disk mirrors.
5. Unblock the database server so that transactions can resume.
6. Back up the logical logs.
7. Put the mirror disk into storage, copy the mirrored data to another computer, or back it up to tape or storage media by using a file-system backup program or an operating-system copy command.
8. Reconnect and synchronize the disk mirrors, if necessary.

You could use this technique to set up a high-availability environment and use external backups for failover to a second database server.

Data Restored in an External Restore

If you lose a disk, coserver, or the whole system, you can externally restore data only if it was externally backed up. You must use the same third-party utility for both the external backup and restore. To externally restore the storage spaces, copy the backed-up data to disk. Use the **onbar -r -e** command to mark the storage spaces as physically restored, replay the logical logs, and bring the storage spaces back online. If you do not specify an external restore command, the database server thinks that these storage spaces are still down.

You can perform these types of external restores:

- **Warm external restore**. Mark noncritical storage spaces as physically restored, then perform a logical restore of these storage spaces.
- **Cold external restore**. Mark storage spaces as physically restored, then perform a logical restore of all storage spaces. Optionally, you can do a point-in-time cold external restore.

Warning: When you perform a cold external restore, ON–Bar does not first attempt to salvage logical-log files from the database server because the external backup has already copied over the logical-log data.

To salvage logical logs, perform **onbar -l -s** before you copy the external backup and perform the external restore (**onbar -r -e**).

Extended Parallel Server

In Extended Parallel Server, you must perform a logical restore on the whole system after an external backup even if all the storage spaces were backed up together.

End of Extended Parallel Server

Renaming Chunks

You can rename chunks in an external cold restore using the rename options syntax for other restores. Use the following commands to specify new chunk names during restore:

```
onbar -r -e -rename -f filename
```

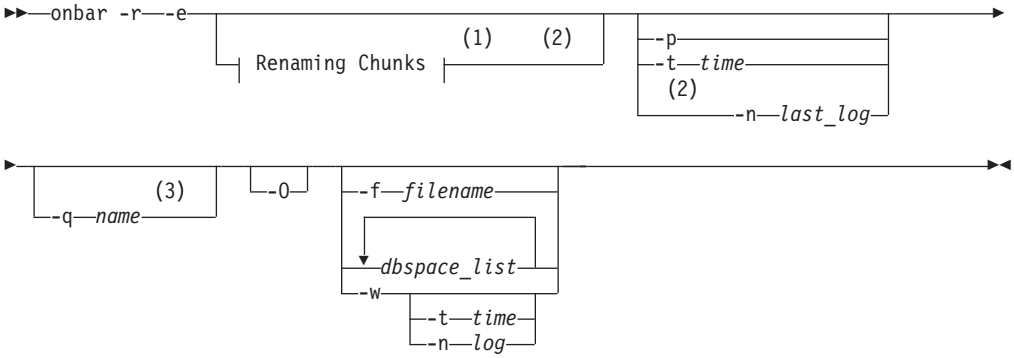
or

```
onbar -r -e rename -p old_path -o old_offset-n new_path-o new_offset
```

Using External Restore Commands

Use the **onbar -r -e** command to perform a warm or cold external restore. This command marks the storage spaces as physically restored and restores the logical logs. The following diagram shows the external restore syntax.

Performing an External Restore with ON-Bar



Notes:

1 See 7-27

2 Dynamic Server only

3 Extended Parallel Server only

Element	Purpose	Key Considerations
onbar -r	Specifies a restore	In a cold restore, if you do not specify storage space names, all of them are marked as restored.
-e	Specifies an external restore	Must be used with the **-r** option. In a warm external restore, marks the down storage spaces as restored unless the **-O** option is specified.

Element	Purpose	Key Considerations
dbspace_list	Names one or more storage spaces to be marked as restored in a warm restore	If you do not enter ***dbspace_list*** or **-f *filename*** and the database server is online or quiescent, ON–Bar marks only the down storage spaces as restored. If you enter more than one storage-space name, use a space to separate the names.
-f *filename*	Restores the storage spaces that are listed in the text file whose path name *filename* provides	To avoid entering a long list of storage spaces every time, use this option. The ***filename*** can be any valid UNIX or Windows file name.
-n *last_log*	Indicates the number of the last log to restore (IDS)	If any logical logs exist after this one, ON–Bar does not restore them and data is lost. The **-n** option does not work with the **-p** option.
-O	Restores online storage spaces	None.
-p	Specifies an external physical restore only	After the physical restore completes, you must perform a logical restore because XPS does not support whole-system restore.
-q *name*	Allows you to assign a session name to the external restore (XPS)	*DBSERVERNAMErandom_number* is the default session name. The session name must be unique and can be up to 128 characters.
-t *time*	Restores the last backup before the specified point in time. If you pick a backup made after the point in time, the restore will fail.	You can use a point-in-time restore in a cold restore only. You must restore all storage spaces. How you enter the time depends on your current GLS locale convention. If the GLS locale is not set, use English-style date format. See "Restoring Data to a Point in Time" on page 7-20.
-w	Performs a whole-system restore of all storage spaces and logical logs from the last whole-system backup (IDS)	You must specify the -w option in a cold restore. If you specify onbar -r -w without a whole-system backup, return code 147 appears because ON-Bar cannot find any storage spaces backed up as part of a whole-system backup.

Rules for an External Restore

Before you begin an external restore, keep the following rules in mind:

- You must externally restore from an external backup. Although the external backup is treated as a level-0 backup, it might actually be a non-Informix incremental backup.
- A warm external restore restores only noncritical storage spaces.
- You cannot externally restore temporary dbspaces.
- You cannot externally restore from regular ON–Bar backups.
- You cannot verify that you are restoring from the correct backup and that the storage media is readable using ON–Bar.
- If the external backups are from different times, the external restore uses the beginning logical log from the oldest backup.

These rules apply to cold external restores only:

- Salvage the logical logs (**onbar -b -l -s**) before you switch the disks that contain the critical storage spaces.
- If you are restoring critical dbspaces, the database server *must* be offline (Dynamic Server) or in microkernel mode (Extended Parallel Server).
- Point-in-time external restores must be cold and restore all storage spaces.

- The external backups of all critical dbspaces of the database server instance must have been simultaneous. All critical dbspaces must have been backed up within the same **onmode -c block ... onmode -c unblock** (IDS) or **onutil ebr block ... onutil ebr unblock commit** (XPS) command bracket.

Performing an External Restore

This section describes procedures for performing cold and warm external restores.

Cold External Restore Procedure

If you specify the **onbar -r -e** command in a cold restore, you must restore all storage spaces. Use the **onbar -r -e -p** command to restore all or specific storage spaces.

To perform a cold external restore:

1. Shut down the database server.

   Dynamic Server

   To shut down the database server, use the **onmode -ky** command.

   End of Dynamic Server

   Extended Parallel Server

   To bring the database server to microkernel mode, use the **xctl -C oninit -m** command.

   End of Extended Parallel Server

2. Salvage the logical logs:

   ```
   onbar -b -l -s
   ```

3. To restore the storage spaces from an external backup, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or a file-backup program.

   You must restore the storage spaces to the same path as the original data and include all the chunk files.

4. To perform an external restore of all storage spaces and logical logs, use the following command:

   ```
   onbar -r -e
   ```

   To perform a point-in-time external restore of all storage spaces, use the following command:

   ```
   onbar -r -e -t datetime
   ```

   Extended Parallel Server

   To perform a physical external restore of specific storage spaces, followed by a logical restore, use these commands:

   ```
   onbar -r -e -p rootdbs
   onbar -r -e -p critical_space1
   onbar -r -e -p other_dbspaces
   onbar -r -l
   ```

   End of Extended Parallel Server

   This step brings the database server to fast-recovery mode.

5. ON–Bar and the database server roll forward the logical logs and bring the storage spaces online.

Performing an External Restore when Chunks are Mirrored by the Server:
When you mirror chunks using the mirroring support provided by the database server, following a restore the database server assumes that you have restored only the primary chunks from external backup and that mirror chunks do not contain valid data. You must recover mirror chunks manually using the **alter dbspace** *dbspacename* **online chunk** *chunkpath* command using the **onutil** utility. During recovery, the server writes the appropriate **onutil** command you should use to perform the mirror recovery to the message log. You can run these commands at a time you deem appropriate. Because recreating the mirror chunks consumes server resources, if your goal is to bring the server online as soon as possible following recovery you might want to defer mirror recovery until a scheduled maintenance period.

To restore from an external backup when you use the mirroring support provided by the database server:

1. Shut down the database server.

   Dynamic Server

   To shut down the database server, use the **onmode -ky** command.

   End of Dynamic Server

   Extended Parallel Server

   To bring the database server to microkernel mode, use the **xctl -C oninit -m** command.

   End of Extended Parallel Server

2. Salvage the logical logs:

   ```
   onbar -b -l -s
   ```

3. To restore the storage spaces from an external backup, use a copy command, such as **cp**, **dd**, or **tar** on UNIX, or use a file-backup program.

   You must restore the storage spaces to the same path as the original data and include all the chunk files. You need only restore data for the primary chunks.

4. To perform an external restore of all storage spaces and logical logs, use the following command:

   ```
   onbar -r -e
   ```

   To perform a point-in-time external restore of all storage spaces, use the following command:

   ```
   onbar -r -e -t datetime
   ```

   Extended Parallel Server

   To perform a physical external restore of specific storage spaces, followed by a logical restore, use these commands:

   ```
   onbar -r -e -p rootdbs
   onbar -r -e -p critical_space1
   onbar -r -e -p other_dbspaces
   onbar -r -l
   ```

   End of Extended Parallel Server

   This step brings the database server to fast-recovery mode.

5. ON–Bar and the database server roll forward the logical logs and bring the storage spaces online.
6. Perform mirror chunk recovery using **onutil.**

Mixed External Restore Restriction

ON–Bar does not support mixed external restores. For example, the following sequence of commands might fail:

```
onbar -r -e rootdbs
onbar -r -e other_dbspaces
```

Warm External Restore Procedure

The database server is online during a warm external restore. A warm external restore involves only noncritical storage spaces.

To perform a warm external restore:

1. To restore the storage spaces from an external backup, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or a file-backup program.

   You must restore the storage spaces to the same path as the original data and include all the chunk files for each restored storage space.
2. Perform a warm external restore of the noncritical storage spaces to bring them online.
   a. To restore selected storage spaces and all logical logs, use the following command:

      ```
      onbar -r -e dbspace_list
      ```

   b. To restore all the down noncritical storage spaces and logical logs, use the following command:

      ```
      onbar -r -e
      ```

   c. To restore all the down noncritical storage spaces and logical logs in separate steps, use the following commands:

      ```
      onbar -r -e -p
      onbar -r -l
      ```

      Specify the logstreams for only the coservers from where you restored the dbspaces. For example, use the following command to restore logstreams 1, 2, and 3:

      ```
      onbar -r -l 1 2 3
      ```

   d. To restore *all* the noncritical storage spaces and logical logs, use the following command:

      ```
      onbar -r -e -0
      ```

Examples of External Restore Commands

The following table contains examples of external restore commands.

Action	External Restore Command	Comments
Complete external restore	**onbar -r -e**	In a cold restore, restores everything. In a warm restore, restores all down noncritical storage spaces.
Physical external restore and separate logical restore	**onbar -r -e -p** **onbar -r -l**	You must always perform a logical restore (XPS). If the external backups come from different times, you must perform a logical restore. The system restores the logical logs from the oldest external backup (IDS).
External restore of selected storage spaces and logical logs	**onbar -r -e** ***dbspace_list***	Use this command in a warm external restore only.
External restore of selected storage spaces and separate logical restore	**onbar -r -e -p** ***dbspace_list*** **onbar -r -l**	Use this command in a warm external restore only. (IDS) Use this command in a warm or cold external restore. (XPS)
External point-in-time (cold) restore	**onbar -r -e -t** ***datetime***	Be sure to select a collection of backups from before the specified time.
External (cold) restore with renamed chunks	**onbar -r -e rename -p** ***old_path*** **-o** ***old_offset*****-n** ***new_path*****-o** ***new_offset***	Use this command to rename chunks in cold external restore only.
Whole-system external restore (IDS)	**onbar -r -e -w** or **onbar -r -e -p -w**	When you use **onbar -r -e -w -p**, back up all storage spaces in one block and unblock session. That way, all storage spaces have the same checkpoint.

Initializing HDR with an External Backup and Restore (IDS)

You can use external backups to initialize High-Availability Data Replication (HDR). For more information on HDR, see "Initializing High-Availability Data Replication with ON-Bar (IDS)" on page 7-32 and the *IBM Informix Administrator's Guide*.

To initialize HDR with an external backup and restore:

1. Block the source database server with the following command:
   ```
   onmode -c block
   ```
2. Externally back up all chunks on the source database server.
3. When the backup completes, unblock the source database server:
   ```
   onmode -c unblock
   ```
4. Make the source database server the primary server:
   ```
   onmode -d primary secondary_servername
   ```
5. On the target database server, restore the data from the external backup with a copy or file-backup program.
6. On the target database server, restore the external backup of all chunks:
   ```
   onbar -r -e -p
   ```

   Note: Note: On HDR, secondary server can restore only level-0 archives.
7. Make the target database server the secondary server:
   ```
   onmode -d secondary primary_servername
   ```

8. If the logical-log records written to the primary database server since step 1 still reside on the primary database server disk, the secondary database server reads these records to perform the logical recovery. Otherwise, perform the logical recovery:

   ```
   onbar -r -l
   ```

   The `database server operational` messages appear in the message log on the primary and secondary servers.

Chapter 9. Customizing and Maintaining ON-Bar

In This Chapter 9-1
Customizing ON-Bar and Storage-Manager Commands 9-2
Printing the Backup Boot Files 9-2
Migrating Backed-Up Logical Logs to Tape 9-3
Using start_worker.sh to Start onbar_worker Processes Manually (XPS) 9-4
Expiring and Synchronizing the Backup Catalogs 9-5
Choosing an Expiration Policy 9-5
Using the onsmsync Utility 9-6
Removing Expired Backups 9-7
Expiring Old Backups on ISM 9-7
Regenerating the Emergency Boot File 9-7
Regenerating the sysutils Database 9-8
Deleting a Bad Backup 9-8
Expiring Backups Based on the Retention Date 9-8
Expiring a Generation of Backups. 9-8
Expiring Backups Based on the Retention Interval 9-8
Expiring Backups With Multiple Point-In-Time Restores 9-8
Expiring All Backups 9-9
Starting and Stopping ON-Bar Sessions (XPS) 9-10
Using the onbar_w Utility 9-10
Monitoring the Backup Scheduler Status (XPS) 9-11
Using the ***onstat -g bus*** Option 9-11
Sample ***onstat -g bus*** Output Without Any ON-Bar Activity 9-11
Sample ***onstat -g bus*** Output During a Dbspace Backup 9-11
Using the ***onstat -g bus_sm*** Option 9-12
Sample ***onstat -g bus_sm*** Output When ON-Bar Is Idle 9-12
Sample ***onstat -g bus_sm*** Output During a Dbspace Backup 9-12
Monitoring the Performance of ON-Bar and the Storage Managers 9-13
Setting ON-Bar Performance Statistics Levels 9-13
Viewing ON-Bar Backup and Restore Performance Statistics 9-13

In This Chapter

This chapter discusses the following topics:

- Customizing ON–Bar and storage-manager commands with the **onbar** script
- Starting **onbar-worker** processes manually
- Expiring and synchronizing the backup history

Extended Parallel Server

- Starting and stopping ON–Bar sessions
- Monitoring the backup scheduler status
- Monitoring the performance of ON-Bar and your storage manager

End of Extended Parallel Server

© Copyright IBM Corp. 1996, 2008

Customizing ON-Bar and Storage-Manager Commands

When you issue ON–Bar commands from the command line, the arguments are passed to the **onbar** script and then to **onbar_d**. Use the **onbar** shell script on UNIX or the **onbar** batch file on Windows to customize backup and restore commands, start ISM, and back up the ISM catalog. The **onbar** script is located in the **$INFORMIXDIR/bin** directory on UNIX and in the **%INFORMIXDIR%\bin** directory on Windows.

The default **onbar** script assumes that the currently installed storage manager is ISM and backs up the ISM catalogs. If you are using a different storage manager, edit the onbar script, delete the ISM-specific lines, and optionally, add storage-manager commands.

For background information on the **onbar** script or batch file, see "ON-Bar Utilities" on page 3-7 and "Your Customized onbar Script is Saved on New Installations" on page 4-6.

The default **onbar** script contains the following sections:

- Add startup processing here

  Use this section to initialize the storage manager, if necessary, and set environment variables.
- End startup processing here

  This section starts the **onbar_d** driver and checks the return code. Use this section for **onbar_d** and storage-manager commands.
- Add cleanup processing here

  The code in this section backs up the ISM catalogs to the ISMData volume pool after the backup operation is complete. If you are using a third-party storage manager, delete the ISM-specific information.

  If you use a name other than ISMData for the volume pool, change it to the name specified in the ISM_DATA_POOL configuration parameter.

  Dynamic Server

  The **archecker** temporary files are also removed.

  End of Dynamic Server
- End cleanup processing here

  Use this section to return **onbar_d** error codes.

Warning: Edit the **onbar** script carefully. Accidental deletions or changes might cause undesired side effects. For example, backup verification might leave behind temporary files if the cleanup code near the end of the **onbar** script is changed.

Printing the Backup Boot Files

Use the following examples of what to add to the **onbar** script to print the emergency boot file if the backup is successful. Each time that you issue the **onbar -b** command, the emergency boot file is printed.

UNIX Only

```
onbar_d "$@"    # receives onbar arguments from command line return_code = $?
# check return code
```

```
# if backup (onbar -b) is successful, prints emergency boot file
if [$return_code -eq 0 -a "$1" = "-b"]; then
   servernum='awk '/^SERVERNUM/ {print $2}' $INFORMIXDIR/etc/$ONCONFIG '
   lpr \$INFORMIXDIR/etc/ixbar.$servernum
fi
exit $return_code
```

End of UNIX Only

Extended Parallel Server

To print the backup boot files on all coservers, replace the line:

```
lpr \$INFORMIXDIR/etc/ixbar.$servernum
```

with:

```
xctl lpr \$INFORMIXDIR/etc/Bixbar_\'hostname\'.$servernum
```

If more coservers than hosts are on the system, this script prints the same boot files twice.

End of Extended Parallel Server

Windows Only

```
@echo off
%INFORMIXDIR%\bin\onbar_d %*
set onbar_d_return=%errorlevel%

if "%onbar_d_return%" == "0" goto backupcom
goto skip

REM Check if this is a backup command

:backupcom
if "%1" == "-b" goto printboot
goto skip

REM Print the onbar boot file

:printboot
print %INFORMIXDIR%\etc\ixbar.???

REM Set the return code from onbar_d (this must be on the last line of the script)

:skip
%INFORMIXDIR%\bin\set_error %onbar_d_return%
:end
```

End of Windows Only

Migrating Backed-Up Logical Logs to Tape

You can set up your storage manager to back up logical logs to disk and then write a script to automatically migrate the logical logs from disk to tape for off-site storage. Edit the **onbar** script to call this migration script after the **onbar_d** process

completes. The following example shows a script that calls the migration script:

UNIX Only

```
onbar_d "$@" # starts the backup or restore
EXIT_CODE=$?         # any errors?

PHYS_ONLY=false      #if it's physical-only, do nothing
for OPTION in $*; do
    if [$OPTION = -p]; then
        PHYS_ONLY = true

    fi
done
if ! PHYS_ONLY; then   # if logs were backed up, call another
     migrate_logs      # program to move them to tape
fi
```

End of UNIX Only

Windows Only

This example for Windows invokes the migration script:

```
%INFORMIXDIR%\bin\onbar_d %*
set onbar_d_return=%errorlevel%

if "%onbar_d_return%" == "0" goto backupcom
goto skip

REM Check if the command is a backup command

:backupcom
if "%1" == "-b" goto m_log
if "%1" == "-l" goto m_log
goto skip

REM Invoke the user-defined program to migrate the logs

:m_log
migrate_log

REM Set the return code from onbar_d (this must be on the last line of the script)

:skip
%INFORMIXDIR%\bin\set_error %onbar_d_return%

:end
```

End of Windows Only

Using start_worker.sh to Start onbar_worker Processes Manually (XPS)

To start **onbar-worker** processes manually, use the **start_worker.sh** script in **$INFORMIXDIR/etc** or the **onbar_w** utility. Edit this file to specify additional setup and cleanup tasks when the database server starts the **onbar-worker** processes or to set environment variables.

The default **start_worker.sh** script contains only one line, which calls **onbar_w** to start an **onbar-worker** process.

If the storage manager does not have special requirements for worker processes that pass data to it, you do not have to change the **start_worker.sh** script.

If the storage manager has special requirements, edit **start_worker.sh** to include operating-system commands that set up the environment before you start the **onbar-worker** process or perform other required actions after **onbar-worker** processes start.

The storage-manager documentation should describe any special requirements. If **onbar-worker** processes are not working correctly with a storage manager, check if the storage manager has any special requirements. If they are not listed in the documentation, or if they are not clearly stated so that you can add them to **start_worker**, contact the storage-manager manufacturer directly for more information.

Expiring and Synchronizing the Backup Catalogs

ON–Bar maintains a history of backup and restore operations in the **sysutils** database and an extra copy of the backup history in the emergency boot file. ON–Bar uses the **sysutils** database in a warm restore when only a portion of the data is lost. ON–Bar uses the emergency boot file in a cold restore because the **sysutils** database cannot be accessed. You can use the **onsmsync** utility to regenerate the emergency boot file and expire old backups.

Depending on the command options you supply, the **onsmsync** utility can remove the following items from the **sysutils** database and the emergency boot file:

- Backups that the storage manager has expired
- Old backups based on the age of backup
- Old backups based on the number of times they have been backed up

Use **onsmsync** with the database server online or in quiescent mode to synchronize both the **sysutils** database and the emergency boot file.

To synchronize the sysutils database:

1. Bring the database server online or to quiescent mode.
2. Run the **onsmsync** utility without any options.

The **onsmsync** utility synchronizes the **sysutils** database, the storage manager, and the emergency boot file as follows:

- Adds backup history to **sysutils** that is in the emergency boot file but is missing from the **sysutils** database.
- Removes the records of restores, whole-system restores (IDS), fake backups, successful and failed backups from the **sysutils** database.
- Expires old logical logs that are no longer needed.
- Regenerates the emergency boot file from the **sysutils** database.

Choosing an Expiration Policy

You can choose from the following three expiration policies:

- **Retention date** (**-t**) deletes all backups before a particular date and time.
- **Retention interval** (**-i**) deletes all backups older than some period of time.
- **Retention generation** (**-g**) keeps a certain number of versions of each backup.

ON–Bar always retains the latest level-0 backup for each storage space. It expires all level-0 backups older than the specified time unless they are required to restore from the oldest retained level-1 backup.

ON–Bar expires all level-1 backups older than the specified time unless they are required to restore from the oldest retained level-2 backup.

Dynamic Server

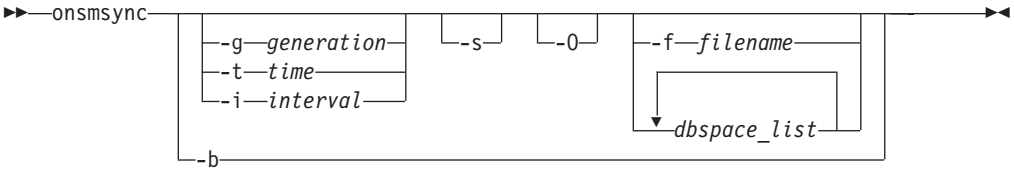
ON–Bar retains a whole-system backup that starts before the specified retention time and ends after the specified retention time.

End of Dynamic Server

Using the onsmsync Utility

The order of the commands does not matter except that the storage-space names or filename must come last.

Tip: To control whether the **sysutils** database maintains a history for expired backups and restores, use the BAR_HISTORY configuration parameter. For information, see "BAR_HISTORY" on page 10-10.



Element	Purpose	Key Considerations
no options	Synchronizes the **sysutils** database and emergency boot file with the storage-manager catalog	None.
-b	Regenerates both the emergency boot file (**ixbar.***servernum*) and the **sysutils** database from each other.	If the **ixbar** file is empty or does not exist, **onsmsync -b** recreates the **ixbar** file and populates it from the **syutils** tables. If the **ixbar** is not empty and contains object data, **onsmsync -b** updates the **sysutils** database and the **ixbar** file so that they are in sync. If the **ixbar** file has entries and the **sysutils** database has been rebuilt, but is empty because it does not contain data, **onsmsync -b** recreates the **sysutils** data from the **ixbar** file. The **-b** element is not used with the other **onsmsync** options. Additionally, it does not synchronize with the storage manager.
dbspace_list	Lists the storage spaces to expire	If you enter more than one storage space, use a space to separate the names.
-f ***filename***	Specifies the path name of a file that contains a list of storage spaces to expire	Use this option to avoid entering a long list of storage spaces. The filename can be any valid UNIX or Windows filename.
-g ***generation***	Retains a certain number of versions of each level-0 backup	The latest generation of backups are retained and all earlier ones are expired.

Element	Purpose	Key Considerations
-i *interval*	Expires all backups older than some period of time	Retains backups younger than this interval. Backups older than interval are not expired if they are needed to restore from other backups after that interval. Use the ANSI or GLS format for the *interval*: YYYY-MM or DD HH:MM:SS
-s	Skips backups that the storage manager has expired	Use this option to skip synchronizing objects that are already expired from the storage manager. The object expiration will be based on other arguments if the **-s option is provided.**
-O	Enforces expiration policy strictly	If used with the **-t**, **-g**, or **-i** option, expires all levels of a backup, even if some of them are needed to restore from a backup that occurred after the expiration date. The **-O** option does not affect logical-log expiration. See "Expiring All Backups" on page 9-9.
-t *datetime*	Expires all backups before a particular date and time	Retains backups younger than this datetime. Backups older than *datetime* are not expired if they are needed to restore from other backups after that *datetime*. Use the ANSI or **GLS_DATETIME** format for ***datetime***.

Removing Expired Backups

If called with no options, the **onsmsync** utility compares the backups in the **sysutils** database and emergency boot file with the backups in the storage-manager catalog. The **onsmsync** utility removes all backups that are not in the storage manager catalog from the **sysutils** database and emergency boot file.

Extended Parallel Server

The **onsmsync** utility starts **onbar-merger** processes that delete backups on all nodes that contain storage managers.

End of Extended Parallel Server

Expiring Old Backups on ISM

ISM and certain third-party storage managers do not allow **onsmsync** to delete backups from the storage manager. First, manually expire or delete the old backups from the storage manager. Then, run **onsmsync** without any parameters.

To expire old backups on ISM:

1. To manually expire the old backups from ISM, use the **ism_config -retention #*days*** command.

   For more information, see the *IBM Informix Storage Manager Administrator's Guide*.
2. Run **onsmsync** without any options.

Regenerating the Emergency Boot File

To regenerate the emergency boot file only, use the following command:

```
onsmsync -b
```

Dynamic Server

On Dynamic Server, this command saves the old emergency boot file as **ixbar.*server_number.system_time*** and regenerates it as **ixbar.*server_number***.

Extended Parallel Server

On Extended Parallel Server, this command saves the old backup boot file on each coserver as **Bixbar**_*hostname_system_time.server_number* and regenerates it as **Bixbar**_*hostname.server_number*.

End of Extended Parallel Server

Regenerating the sysutils Database

If you lose the **sysutils** database, use the **bldutil** utility in **$INFORMIXDIR/etc** on UNIX or **%INFORMIXDIR%\etc** on Windows to recreate the **sysutils** database with empty tables.

Then use the **onsmsync** utility to recreate the backup and restore data in **sysutils**.

Important: If both the **sysutils** database and emergency boot file are missing, you cannot regenerate them with **onsmsync**. Be sure to back up the emergency boot file with your other operating-system files.

Deleting a Bad Backup

The **onsmsync** utility cannot tell which backups failed verification. If the *latest* backup failed verification but an earlier one was successful, you must manually delete the failed backup records from the storage manager and then run **onsmsync** with no options to synchronize ON–Bar. For more information, see Chapter 6, "Verifying Backups," on page 6-1.

Expiring Backups Based on the Retention Date

The following example expires backups that started before November 24, 2006 and *all* fake backups, failed backups, and restores:

```
onsmsync -t "2006-11-24 00:00"
```

Expiring a Generation of Backups

The following example retains the latest three sets of level-0 backups and the associated incremental backups, and expires *all* earlier backups and all restores, fake backups, and failed backups:

```
onsmsync -g 3
```

Expiring Backups Based on the Retention Interval

The following example expires all backups that are older than three days and *all* fake backups, failed backups, and restores:

```
onsmsync -i "3 00:00:00"
```

The following example expires all backups older than 18 months (written as 1 year + 6 months):

```
onsmsync -i "1-6"
```

Expiring Backups With Multiple Point-In-Time Restores

If you perform more than one point-in-time restores, multiple timelines for backups exist. Figure 9-1 shows three timelines with their backups.

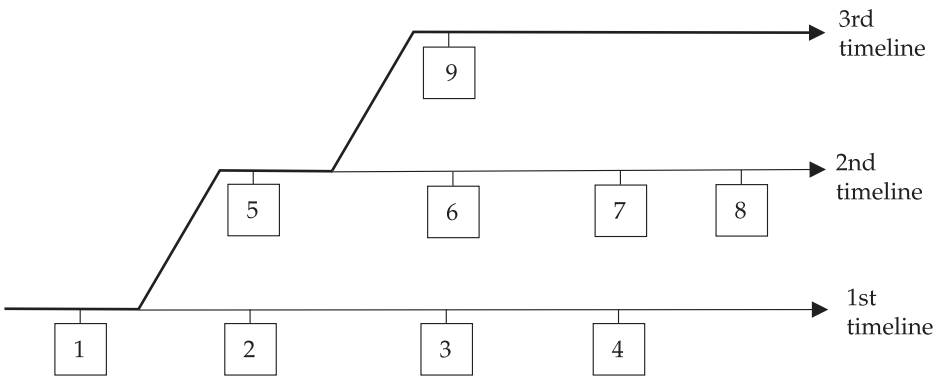


Figure 9-1. Multiple Timelines for Backups

In this example, the second timeline begins with a point-in-time restore to backup 1. The second timeline consists of backups 1, 5, 6, 7, and 8. The third timeline (in bold) consists of backups 1, 5, and 9. The third timeline is considered the current timeline because it contains the latest backup.

When you run **onsmsync** to expire old backups, **onsmsync** removes the old backups from the current timeline, and make sure that the current timeline is restorable from the backup objects that are retained. All other backups that are not in the current timeline are also expired but **onsmsync** does not make sure that the other timelines are restorable from the objects retained.

The **onsmsync** utility applies expiration policies in the following order to make sure that objects from current timeline are expired according to the specified expiration policy and that the current timeline is restorable:

- Apply the expiration policy on all sets of backup objects.
- Unexpire backup objects that belong to the current timeline.
- Apply the expiration policy on the current timeline to ensure that the current timeline is restorable.

At the same time, the expiration policy is applied to backups in other timelines.

For example, if you execute the **onsmsync -g 2** command on the example in Figure 9-1, backup 1 from the current timeline is expired, and backups 2, 3, 4, 6, and 7 from the first and second timelines are expired. Backups 1, 5, and 9 from the current timeline are retained. Backup 8 is retained from other timelines.

Expiring All Backups

The **onsmsync** utility retains the latest level-0 backup unless you use the **-O** option. If you use the **-O** and **-t** options, all backups from before the specified *time* are removed even if they are needed for restore. If you use the **-O** and **-i** options, all backups from before the specified *interval* are removed even if they are needed for restore.

For example, to expire *all* backups, specify the following:

```
onsmsync -O -g 0
```

Warning: If you use the **-O** option with the **-t**, **-i**, or **-g** options, you might accidentally delete critical backups, making restores impossible.

Starting and Stopping ON-Bar Sessions (XPS)

Session control commands let you stop and restart ON–Bar sessions. You might stop and restart a session to:

- Temporarily stop continuous-log backup
- Temporarily stop some or all ON–Bar sessions while computer traffic is heavy

Starting and Stopping ON-Bar Sessions

Element	Purpose	Key Considerations
off	Suspends a session	None.
on	Resumes a previously suspended session	None.
-d	Destroys a session	ON–Bar completes backing up the current storage spaces and logical logs. Storage spaces and logical logs that are not assigned to an **onbar-worker** are not processed.
-q ***session***	Specifies the session name The **-q** option is optional but ***session*** is required.	The session name can be up to 128 characters. Put quotation marks around the session name if it contains white spaces.

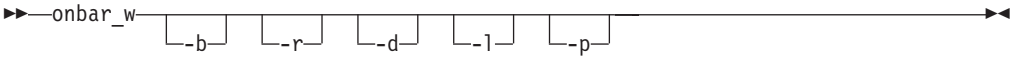
The order of the options does not matter. For example, to destroy the ON–Bar session, **myses10**, use one of the following commands:

```
onbar -d -q myses10
onbar -q myses10 -d
```

Using the onbar_w Utility

To start **onbar-worker** processes manually, use the **onbar_w** utility.

Using onbar_w to Start onbar_worker



Element	Purpose	Key Considerations
-b	Accepts request to back up storage spaces and logical logs	None.
-r	Accepts request to restore storage spaces and logical logs	None.
-d	Accepts request to back up and restore storage spaces	None.
-l	Accepts requests to back up and restore logical logs	None.
-p	Accepts queries from the Backup Scheduler that asks where particular objects have been backed up	None.

Specifying **onbar_w** without any options (the default) is the same as specifying **onbar_w -b -r -d -l -p**. Table 9-1 shows some examples of **onbar_w** options. Some

option combinations also do the same thing.

Table 9-1. onbar_w Options

onbar_w Options	What Happens
none	The onbar-worker process does everything. It backs up and restores storage spaces and logical logs, and places the storage spaces and logical logs in the correct storage manager.
-b	Backs up storage spaces and logical logs
-b -d	Backs up storage spaces
-b -l	Backs up logical logs
-b -p	Backs up storage spaces and logical logs, and places them in the storage manager
-b -r	Backs up and restores storage spaces and logical logs
-d -p	Backs up and restores storage spaces, and places them in the storage manager
-l	Backs up and restores logical logs
-p	Places storage spaces and logical logs in the storage manager
-r	Restores storage spaces and logical logs
-r -p	Restores storage spaces and logical logs, and places them in the storage manager
-b -r -d -l -p	Does everything
-b -r -p	Does everything
-d -l -p	Does everything

Monitoring the Backup Scheduler Status (XPS)

Use the **onstat -g bus** and **onstat -g bus_sm** options to monitor the status of the Backup Scheduler. The Backup Scheduler tracks scheduled and active ON–Bar sessions. For SQL access to the Backup Scheduler, see "Backup Scheduler SMI Tables (XPS)" on page 11-8.

Using the *onstat -g bus* Option

The **onstat -g bus** option shows the current Backup Scheduler sessions, what work is scheduled for each ON–Bar session, and what work is currently in progress. Both options display identical information.

Sample *onstat -g bus* Output Without Any ON-Bar Activity

In the following example, two logical-log backup sessions are suspended:

```
onstat -g bus

    Backup scheduler sessions
    -------------------------

    Session "Log backup 2" state SUSPENDED error 0
    Session "Log backup 1" state SUSPENDED error 0
```

Sample *onstat -g bus* Output During a Dbspace Backup

In the following example, ON–Bar and the Backup Scheduler are working on session **gilism824589**. Currently, dbspace **dbs1.2** is being backed up. Dbspaces **dbs12.1**, **dbs12.2**, and **other** are waiting to be backed up.

```
onstat -g bus

    Backup scheduler sessions
    -------------------------

    Session "Log backup 2" state SUSPENDED error 0
    Session "Log backup 1" state SUSPENDED error 0
    Session "gilism824589" state WAITING error 0
   DBSPACE(dbsl.2) level 0 BACKUP,RUNNING
   DBSPACE(dbsl2.1) level 0 BACKUP,READY
   DBSPACE(dbsl2.2) level 0 BACKUP,READY
   DBSPACE(other) level 0 BACKUP,READY
```

Using the *onstat -g bus_sm* Option

The **onstat -g bus_sm** option shows the current storage-manager configuration, what storage managers are assigned to each coserver, and what work each storage manager is currently performing.

Sample *onstat -g bus_sm* Output When ON-Bar Is Idle

The following example shows the storage-manager version, storage-manager name, the number of **onbar-worker** processes, the number of coservers, the maximum number of **onbar-worker** processes started, and the ON–Bar idle timeout:

```
onstat -g bus_sm

    Configured storage managers
    ---------------------------

    BAR_SM              1
    BAR_SM_NAME         ism
    BAR_WORKER_COSVR    1
    BAR_DBS_COSVR       1,2
    BAR_LOG_COSVR       1,2
    BAR_WORKER_MAX      1
    BAR_IDLE_TIMEOUT    5
    END
```

Sample *onstat -g bus_sm* Output During a Dbspace Backup

When a backup or restore session is active, **onstat -g bus_sm** also displays information about the active **onbar-worker** processes, as the following example shows:

```
onstat -g bus_sm

    Configured storage managers
    ---------------------------

    BAR_SM              1
    BAR_SM_NAME         ism
    BAR_WORKER_COSVR    1
    BAR_DBS_COSVR       1,2
    BAR_LOG_COSVR       1,2
    BAR_WORKER_MAX      1
    BAR_IDLE_TIMEOUT    5
    END

Active workers:

   Worker 2 Coserver 1 Pid 4590 State BUSY "dbsl.2.0"
```

The **onbar-worker** process is backing up dbslice **dbsl.2.0**.

Monitoring the Performance of ON-Bar and the Storage Managers

You can monitor the performance of ON-Bar and your storage manager. You can specify the level of performance monitoring and have the statistics print to the ON-Bar activity log. The BAR_PERFORMANCE configuration parameter specifies whether to gather statistics. The following statistics are gathered:

- Total time spent in XBSA calls.
- Total time spent in Archive API calls.
- Time spent by ON-Bar in transferring data to and from XBSA (storage manager calls).
- Time spent by ON-Bar in transferring data between ON-Bar to IDS.
- Amount of data transferred to or from the XBSA API.
- Amount of data transferred to or from the Archive API.

Setting ON-Bar Performance Statistics Levels

To specify the level of performance statistics that are printed to the ON-Bar activity log, set the BAR_PERFORMANCE configuration parameter in **onconfig**.

For example, the following setting displays the time spent transferring data between the IDS instance and the storage manager:

```
BAR_PERFORMANCE 1
```

See "BAR_PERFORMANCE" on page 10-13 for information about the options for this parameter.

Viewing ON-Bar Backup and Restore Performance Statistics

To view ON-Bar performance results, open the ON-Bar activity log. See "BAR_ACT_LOG" on page 10-6 to determine where the activity log is located.

When BAR_PEFORMANCE is set to **1** or **3**, the activity report will display a transfer rate report:

```
2007-06-03 15:38:02 8597  8595 Begin restore logical log 310 (Storage Manager
                               copy ID: 28206 0).
2007-06-03 15:38:03 8597  8595 Completed restore logical log 310.
2007-06-03 15:38:08 8597  8595 Completed logical restore.
2007-06-03 15:38:19 8597  8595 PERFORMANCE INFORMATION

                                                    TRASFER RATES
-------------------------------------------------------------------------------------------------------------------
|       OBJECT       |                  XBSA API                                  |                  SERVER API                                  |
|        NAME        | xfer-kbytes   xfer-time RATIO(kb/s)    API-TIME | xfer-kbytes   xfer-time RATIO(kb/s)    API-TIME |
-------------------------------------------------------------------------------------------------------------------
| 309                |          62       0.479         129       1.078 |          62       0.019        3310       0.310 |
| 310                |          62       0.407         152       1.098 |          62       0.025        2522       0.025 |
| rootdbs            |        5828       0.618        9436       1.864 |        5828       8.922         653       8.931 |
| datadbs01          |          62       0.488         127       1.768 |          62       0.004       17174       0.004 |
| datadbs02          |          62       0.306         203       1.568 |          62       0.008        8106       0.008 |
| datadbs03          |          62       0.304         204       1.574 |          62       0.007        8843       0.007 |
| datadbs04          |          62       0.306         202       1.563 |          62       0.007        8664       0.007 |
| datadbs05          |          62       0.315         197       1.585 |          62       0.007        8513       0.007 |
| datadbs06          |          62       0.310         200       1.583 |          62       0.002       25348       0.002 |
---------------- ------------------------------- ... ------------------------------------------------------------------
| PID =       8597 |       14722      26.758         550     107.476 |       14756      10.678        1382      15.829 |
-------------------------------------------------------------------------------------------------------------------
2007-06-03 15:38:19 8597  8595 PERFORMANCE INFORMATION

                                                  PERFORMANCE CLOCKS
-------------------------------------------------------------------------------------------------------------------
|                      ITEM DESCRPTION                                |                TIME SPENT                    |
-------------------------------------------------------------------------------------------------------------------
| Time to Analyze ixbar file                                          |                                        0.000 |
---------------------------- ----------------------------------------------------------------------------------------
```

Figure 9-2. Sample transfer rate performance in the ON-Bar activity log.

When BAR_PEFORMANCE is set to **2** or **3**, the activity report will have microsecond timestamps as shown in the following example:

```
2007-06-03 16:34:04 15272  15270 /usr/informix/bin/onbar_d complete,
                                 returning 0 (0x00)
2007-06-03 16:45:11.608424 17085  17083 /usr/informix/bin/onbar_d -r -w
2007-06-03 16:46:07.926097 17085  17083 Successfully connected to Storage Manager.
2007-06-03 16:46:08.590675 17085  17083 Begin salvage for log 311.
2007-06-03 16:48:07.817487 17085  17083 Completed salvage of logical log 311.
2007-06-03 16:48:08.790782 17085  17083 Begin salvage for log 312.
2007-06-03 16:48:10.129534 17085  17083 Completed salvage of logical log 312.
2007-06-03 17:06:00.836390 17085  17083 Successfully connected to Storage Manager.
                                   ...
2007-06-03 17:07:26.357521 17085  17083 Completed cold level 0 restore datadbs07.
2007-06-03 17:07:28.268562 17085  17083 Begin cold level 0 restore datadbs08
  (Storage Manager copy ID: 28122 0).
2007-06-03 17:07:29.378405 17085  17083 Completed cold level 0 restore datadbs08.
```

Figure 9-3. Sample processing rates, in microseconds, in the ON-Bar activity log.

Chapter 10. ON-Bar Configuration Parameters

In This Chapter 10-2
archecker Configuration Parameters in AC_CONFIG 10-2
AC_CONFIG File Environment Variable 10-2
AC_MSGPATH 10-3
AC_STORAGE. 10-3
AC_TIMEOUT. 10-4
AC_VERBOSE 10-4
ON-Bar Parameters in ONCONFIG 10-5
ALARMPROGRAM 10-5
ALRM_ALL_EVENTS (IDS) 10-5
BACKUP_FILTER. 10-6
BAR_ACT_LOG 10-6
Specifying BAR_ACT_LOG with a Filename Only 10-7
Using BAR_ACT_LOG on Extended Parallel Server 10-7
BAR_BOOT_DIR (XPS) 10-7
Accessing the Emergency Boot Files 10-7
Saving the Emergency Boot Files During Reversion 10-8
BAR_BSALIB_PATH 10-8
BAR_DBS_COSVR (XPS) 10-9
BAR_DEBUG 10-9
BAR_DEBUG_LOG 10-10
BAR_HISTORY 10-10
BAR_IDLE_TIMEOUT (XPS). 10-10
BAR_IXBAR_PATH (IDS) 10-11
BAR_LOG_COSVR (XPS) 10-11
BAR_MAX_BACKUP (IDS) 10-12
Specifying Serial Backups and Restores 10-12
Specifying Parallel Backups and Restores 10-12
BAR_NB_XPORT_COUNT (IDS) 10-12
BAR_PERFORMANCE 10-13
BAR_PROGRESS_FREQ 10-13
BAR_RETRY 10-14
BAR_RETRY on Dynamic Server (IDS) 10-14
BAR_RETRY on Extended Parallel Server 10-14
BAR_SIZE_FACTOR 10-15
BAR_SM (XPS) 10-16
BAR_SM_NAME (XPS) 10-16
BAR_WORKER_COSVR (XPS) 10-16
BAR_WORKER_MAX (XPS) 10-16
Specifying a Parallel Backup or Restore 10-17
Specifying a Serial Backup or Restore. 10-17
BAR_XFER_BUF_SIZE (IDS) 10-17
BAR_XFER_BUFSIZE (XPS) 10-18
BAR_XPORT_COUNT (XPS). 10-18
ISM_DATA_POOL 10-19
ISM_LOG_POOL 10-19
LOG_BACKUP_MODE (XPS) 10-20
LTAPEDEV (IDS) 10-20
RESTARTABLE_RESTORE (IDS) 10-21
RESTORE_FILTER 10-21

© Copyright IBM Corp. 1996, 2008

In This Chapter

This chapter describes the ON–Bar configuration parameters that you can set in the ONCONFIG file and the **archecker** configuration parameters that you can set in the AC_CONFIG file.

Be sure to configure your storage manager. Depending on the storage manager that you choose, you might set different ON–Bar configuration parameters. If you are using a third-party storage manager, see "Configuring a Third-Party Storage Manager" on page 4-1, before you start ON–Bar.

The following table describes the following attributes (if relevant) for each parameter.

Attribute	Description
ac_config.std value	For **archecker** configuration variables. The default value that appears in the **ac_config.std** file.
onconfig.std value	For **onconfig** configuration variables. The default value that appears in the **onconfig.std** file in Dynamic Server or the **onconfig.xps** file in Extended Parallel Server
if value not present	The value that the database server supplies if the parameter is missing from your ONCONFIG file If this value is present in **onconfig.std**, the database server uses the **onconfig.std** value. If this value is not present in **onconfig.std**, the database server uses this value.
units	The units in which the parameter is expressed
range of values	The valid values for this parameter
takes effect	The time at which a change to the value of the parameter affects ON–Bar operation. Except where indicated, you can change the parameter value between a backup and a restore.
refer to	Cross-reference to further discussion

archecker Configuration Parameters in AC_CONFIG

ON–Bar calls the **archecker** utility to verify backups and you must configure the **archecker** environment variable and parameters before you can use the **onbar -v** option. For more information about using **archecker**, see Chapter 6, "Verifying Backups," on page 6-1.

AC_CONFIG File Environment Variable

default v*alue*

UNIX: **$INFORMIXDIR/etc/ac_config.std**

Windows: **%INFORMIXDIR%\etc\ac_config.std**

takes effect

When ON-Bar starts

Set the **AC_CONFIG** environment variable to the full path name for the **archecker** configuration file (either **ac_config.std** or user defined). You must specify the entire path, including the configuration filename in the AC_CONFIG file or else the **archecker** utility might not work correctly. The following are examples of valid AC_CONFIG path names:

- UNIX: **/usr/informix/etc/ac_config.std** and **/usr/local/my_ac_config.std**
- Windows: **c:\Informix\etc\ac_config.std** and **c:\Informix\etc\my_ac_config.std**

If **AC_CONFIG** is not set, the **archecker** utility sets the default location for the archecker configuration file to **$INFORMIXDIR/etc/ac_config.std** on UNIX or **%INFORMIXDIR%\etc\ac_config.std** on Windows.

The following table shows the **archecker** configuration parameters that you specify in the AC_CONFIG file.

Configuration Parameter	Purpose	IDS	XPS
AC_MSGPATH	Specifies the location of the **archecker** message log	X	X
AC_STORAGE	Specifies the location of the temporary files that **archecker** builds	X	X
AC_TIMEOUT	Specifies the timeout value for the **onbar** and the **archecker** processes if one of them exits prematurely	X	
AC_VERBOSE	Specifies either verbose or quiet mode for **archecker** messages	X	X

AC_MSGPATH

ac_config.std *value*

UNIX: **/tmp/ac_msg.log**

Windows: **c:\temp\ac_msg.log**

takes effect

When ON-Bar starts

The AC_MSGPATH parameter in the AC_CONFIG file specifies the location of the **archecker** message log (**ac_msg.log**). You must specify the entire path of the message log in the AC_CONFIG file or else the **archecker** utility might not work correctly.

When you verify backups with **onbar -v**, the **archecker** utility writes summary messages to the **bar_act.log** and indicates whether the verification succeeded or failed. It writes detailed messages to the **ac_msg.log**. If the backup fails verification, discard the backup and retry another backup, or give the **ac_msg.log** to Technical Support. For sample messages, see "Interpreting Verification Messages" on page 6-6.

AC_STORAGE

ac_config.std *value*

UNIX: **/tmp**

Windows: **c:\temp**

takes effect
When ON-Bar starts

The AC_STORAGE parameter in the **AC_CONFIG** file specifies the location of the directory where **archecker** stores its temporary files. You must specify the entire path of the storage location in **AC_CONFIG** or else the **archecker** utility might not work correctly.

Table 10-1 lists the directories and files that **archecker** builds. If verification is successful, these files are deleted.

Table 10-1. archecker Temporary Files

Directory	Files
CHUNK_BM	Bitmap information for every backed up storage space.
INFO	Statistical analysis and debugging information for the backup.
SAVE	Partition pages in the **PT.########** file.
	Chunk-free pages in the **FL.########** file.
	Reserved pages in the **RS.########** file.
	Blob-free map pages in the **BF.########** file (Dynamic Server only).

To calculate the amount of free space that you need, see "Estimating the Amount of Temporary Space for archecker" on page 6-4. We recommend that you set AC_STORAGE to a location with plenty of free space.

AC_TIMEOUT

ac_config.std *value*
```
UNIX 300
Windows 300
```

units seconds

takes effect
When the **onbar-v** command starts

The AC_TIMEOUT parameter in the AC_CONFIG file specifies the timeout value for the **onbar** and the **archecker** processes if one of them exits prematurely. This parameter was introduced to avoid **onbar** and **archecker** processes waiting for each other indefinitely if one of them exits prematurely, thus avoiding the creation of an orphan and zombie process during data server initialization.

AC_VERBOSE

ac_config.std *value*	1
range of values	1 for verbose messages in **ac_msg.log** 0 for terse messages in **ac_msg.log**
takes effect	When **onbar** starts

The AC_VERBOSE parameter in the **AC_CONFIG** file specifies either verbose or terse output in the **archecker** message log.

ON-Bar Parameters in ONCONFIG

The following section lists the ON–Bar configuration parameters and indicates the database servers to which they apply.

Important: ON–Bar does not use the TAPEDEV, TAPEBLK, TAPESIZE, LTAPEBLK, and LTAPESIZE configuration parameters. ON–Bar checks if LTAPEDEV is set to **/dev/null** on UNIX or **\dev\nul** or NUL on Windows. For more information, see "LTAPEDEV (IDS)" on page 10-20.

ALARMPROGRAM

onconfig.std value	Dynamic Server: On UNIX: **$INFORMIXDIR/etc/alarmprogram.sh** On Windows: **%INFORMIXDIR%\etc\ alarmprogram.bat** Extended Parallel Server: blank
if value not present	Dynamic Server: **no_log.sh** or **no_log.bat** Extended Parallel Server: blank
range of values	Full path name
takes effect	When the database server is shutdown and restarted
refer to	*IBM Informix Administrator's Reference*

Use the ALARMPROGRAM configuration parameter to handle event alarms and start or end automatic log backups.

Use the shell script, **log_full.sh** or **log_full.bat**, for starting automatic log backups. Modify this script and set it to the full path of ALARMPROGRAM in the ONCONFIG file.

To generate event alarms, set ALARMPROGRAM to **$INFORMIXDIR/etc/ alarmprogram.sh** or **%INFORMIXDIR%\etc\alarmprogram.bat** and modify the file according. For more information, see the *IBM Informix Administrator's Reference*.

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

Do not use the continuous log backup command (**onbar -b -l -C**) if you have automatic log backup set up through the ALARMPROGRAM parameter, and conversely.

ALRM_ALL_EVENTS (IDS)

onconfig.std value	0
if value not present	No effect
range of values	0 or 1
takes effect	When the database server is shut down and restarted.
refer to	*IBM Informix Administrator's Reference*

The ALRM_ALL_EVENTS configuration parameter is used to alter the default behavior of the ALARMPROGRAM script. When this configuration parameter is set to 1, ALARMPROGRAM will run for all events that are logged in the MSGPATH and email is sent to the administrator including relevant information about the event.

BACKUP_FILTER

onconfig.std *value*	none
if value not present	none
range of values	Full path name of command and any options.
takes effect	When ON-Bar starts

Set the BACKUP_FILTER configuration parameter to specify the path name of a filter program, and any options. For example:

```
BACKUP_FILTER  /bin/compress
```

With the above configuration, the backup filter will be called from ON–Bar as:

```
BACKUP_FILTER  /bin/compress
```

Output produced by this filter will be saved as a single object in the Storage Manager.

The BACKUP_FILTER configuration parameter can include command line options as well as the filter name. If you include command line options, both the filter name and the options must be surrounded by single quotation marks. For example, specify:

```
BACKUP_FILTER 'my_encrypt -file /var/adm/encryption.pass'
```

In this example, the command in quotation marks is used as the filter.

For security purposes, filters should not have write permission to non-privileged users. Permission on the filters will be same as that of permission on other executable files that are called by the IDS server or utilities.

For more information, see “Transforming with Filters during Backup and Restore” on page 3-12.

BAR_ACT_LOG

takes effect	IDS: When **onbar-driver** starts XPS: When **onbar** starts or an **onbar-worker** process starts

The BAR_ACT_LOG configuration parameter specifies the full path name of the ON–Bar activity log. You should specify a path to an existing directory with an appropriate amount of space available or use **$INFORMIXDIR/bar_act.log**.

Whenever a backup or restore activity or error occurs, ON–Bar writes a brief description to the activity log. The format of the file resembles the format of the database server message log. You can examine the activity log to determine the results of ON–Bar actions.

Specifying BAR_ACT_LOG with a Filename Only

If you specify a filename only in the BAR_ACT_LOG parameter, ON–Bar creates the ON–Bar activity log in the working directory in which you started ON–Bar. For example, if you started ON–Bar from **/usr/mydata** on UNIX, the activity log is written to that directory.

UNIX Only

If the database server launches a continuous logical-log backup, ON–Bar writes to the ON–Bar activity log in the working directory for the database server.

End of UNIX Only

Windows Only

If the database server launches a continuous logical-log backup, ON–Bar writes to the activity log in the **%INFORMIXDIR%\bin** directory instead.

End of Windows Only

Using BAR_ACT_LOG on Extended Parallel Server

The **onbar_w** and **onbar_m** utilities also write messages to the activity log.

If you set BAR_ACT_LOG to a local path name, each onbar-driver and onbar-worker process writes to the ON–Bar activity log on the node where it runs. Unless your system is configured to have shared directories, and the ON–Bar activity log is specified to be in a shared directory, each node has its own ON–Bar activity log.

Warning: Even if you set the path of BAR_ACT_LOG to another directory, check to see if the ON–Bar activity log was placed in the default directory. When **onbar-merger** first starts, it writes messages to **/tmp/bar_act.log** until it has a chance to read the ONCONFIG file.

BAR_BOOT_DIR (XPS)

onconfig.std *value*	**/usr/informix/etc**
if value not present	blank
takes effect	When the next **onbar-worker** process starts

The BAR_BOOT_DIR configuration parameter allows you to change the directory path for the emergency boot files. You must specify it in the global section of the **ONCONFIG** file.

Accessing the Emergency Boot Files

The **onbar_w**, **onbar_m**, and **onbar_d** utilities must be able to access the directory that you select for the emergency boot files. All coservers must be able to write to the directory for user root. The BAR_BOOT_DIR parameter is designed to be used on a system where the **INFORMIXDIR** directory is shared but the root user does not have write permissions to the **INFORMIXDIR** directory. For example, you can set BAR_BOOT_DIR to the **/var/informix** directory on a multinode system. Because each node has its own **/var** file system, it prevents read/write permissions problems.

If you change the BAR_BOOT_DIR value while an **onbar-worker** process is running, it will use the old value. Any new **onbar-worker** process started after the change will pick up the new value.

Warning: After you change the BAR_BOOT_DIR value, you must kill all **onbar-worker** processes on all coservers. When the processes end, either perform a level-0 backup of all storage spaces or move the files from the **etc** subdirectory to the new location.

Saving the Emergency Boot Files During Reversion

When you revert to an earlier database server version, the reversion script saves the emergency boot files in the directory that the BAR_BOOT_DIR parameter specifies. Otherwise, the reversion script saves the emergency boot files to the **$INFORMIXDIR/etc** directory if BAR_BOOT_DIR is not set in the **ONCONFIG** file or is set to an invalid directory.

BAR_BSALIB_PATH

onconfig.std *value*	UNIX	None
	Windows	None
if value not present	Windows	None
takes effect	IDS: When **onbar-driver** starts XPS: When an **onbar-worker** process starts	

ON–Bar and the storage manager rely on a shared library to integrate with each other. Configure the BAR_BSALIB_PATH configuration parameter for your storage-manager library. Support for BAR_BSALIB_PATH is platform-specific. Check your machine notes to determine if you can use it with your operating system. You can change the value of BAR_BSALIB_PATH between a backup and restore.

To ensure that this integration takes place, specify the shared-library path name. Set one of the following options:

- Place the storage-manager library in the default directory.

  UNIX Only

  For example, the suffix for Solaris is ***so***, so you specify **$INFORMIXDIR/lib/ibsad001.so** on a Solaris system.

  End of UNIX Only

UNIX Only

- Place the storage-manager library in any directory and create a symbolic link from **$INFORMIXDIR/lib/ibsad001.*platform_extension*** to it.

  If you use ISM, create a symbolic link to **$INFORMIXDIR/lib/libbsa.*platform_extension*** or set BAR_BSALIB_PATH to this absolute path value.

  If you use TSM, create a symbolic link to **$INFORMIXDIR/lib/libtxbsa.*platform_extension*** or set BAR_BSALIB_PATH to this absolute path value.
- Set the **LD_LIBRARY_PATH** environment variable. For example, to use ISM on Solaris, set **LD_LIBRARY_PATH** to **$INFORMIXDIR/lib**.

Windows Only

- Set the path name for the ISM shared library to the installation directory for ISM: **%ISMDIR%\bin\libbsa.dll**.

  The **%ismDIR%** variable includes a version or release number. For example: **set ISMDIR=C:\program files\informix\ism\2.20**. This directory is set when the database server is installed on Windows. This path name is different if you use a different storage manager.

End of Windows Only

Tip: Be sure that the shared library can access the backup data in the storage manager in a restore. You cannot back up using one storage manager and restore using a different storage manager.

BAR_DBS_COSVR (XPS)

onconfig.std *value*	all coservers
range of values	A list of unique positive integers greater than or equal to one
takes effect	When the database server starts

The optional BAR_DBS_COSVR configuration parameter specifies the coservers from which the storage manager that BAR_SM specifies can be sent storage-space backup and restore data. If BAR_DBS_COSVR is set to `0`, the storage manager is not given dbspaces from any coserver. You might specify BAR_DBS_COSVR `0` to reserve a storage manager for logical-log backups only.

The values are coserver numbers, separated by commas. Hyphens indicate ranges. For example, BAR_DBS_COSVR 1-5,7,9 specifies a storage manager that backs up dbspaces on coservers 1, 2, 3, 4, 5, 7, and 9. Do not put spaces between coserver names.

To provide flexibility and improved performance, this list of coservers can overlap with values listed for other storage managers.

BAR_DEBUG

onconfig.std *value*	`0`
if value not present	`0`
units	Levels of debugging information
range of values	`0` to `9`
takes effect	When ON-Bar starts

Set the BAR_DEBUG configuration parameter to a higher value to display more detailed debugging information in the ON-Bar activity log. The default value of `0` displays no debugging information. You can dynamically update the value of BAR_DEBUG in the ONCONFIG file during a session. For more information, see "Specifying the Level of ON-Bar Debugging (IDS)" on page 3-11.

BAR_DEBUG_LOG

onconfig.std *value*	**/usr/informix/bar_dbug.log**
if value not present	On UNIX: **/tmp/bar_dbug.log** On Windows: **\tmp\bar_dbug.log**
takes effect	When ON-Bar starts

The BAR_DEBUG_LOG parameter specifies the location and name of the ON-Bar debug log. For security reasons, you should set BAR_DEBUG_LOG to a directory with restricted permissions, such as the $INFORMIXDIR directory.

BAR_HISTORY

onconfig.std *value*	Not in **onconfig.std**
if value not present	0
range of values	0 to remove records for expired backup objects from **sysutils** 1 to keep records for expired backup objects in **sysutils**
takes effect	When **onsmsync** starts

The BAR_HISTORY parameter specifies whether the **sysutils** database maintains a backup history when you use **onsmsync** to expire old backups. For more information, see "Using the onsmsync Utility" on page 9-6.

If you set the value to 0, **onsmsync** removes the **bar_object**, **bar_action**, and **bar_instance** rows for the expired backup objects from the **sysutils** database. If you set the value to 1, **onsmsync** sets the **act_type** value to 7 in the **bar_action** row and keeps the **bar_action** and **bar_instance** rows for expired backup objects in the **sysutils** database. If you do not set BAR_HISTORY to 1, the restore history is removed.

Regardless of the value of BAR_HISTORY, **onsmsync** removes the line that describes the backup object from the emergency boot file and removes the object from the storage manager when the storage manager expires the object.

BAR_IDLE_TIMEOUT (XPS)

onconfig.std *value*	0
if value not present	0
units	Minutes
range of values	0 to unlimited
takes effect	When the database server starts

The BAR_IDLE_TIMEOUT configuration parameter determines the maximum amount of time in minutes that an **onbar-worker** process can be idle before it is shut down.

The BAR_IDLE_TIMEOUT configuration parameter is optional. If BAR_IDLE_TIMEOUT is set to 0, the **onbar-worker** processes never shut down until you shut down the database server.

You can set this option locally for individual storage managers to override the default or specified global setting.

BAR_IXBAR_PATH (IDS)

onconfig.std *value*

- UNIX or Linux®: **$INFORMIXDIR/etc/ixbar.***servernum*
- Windows: *%**INFORMIXDIR**%**etc**\ **ixbar.***servernum*

takes effect When ON-Bar or **onsmsync** starts

Use the BAR_IXBAR_PATH configuration parameter to change the path and name of the ON-Bar boot file.

By default, the ON-BAR boot file is created in the **%INFORMIXDIR%\etc** folder on Windows and in the **$INFORMIXDIR/etc** folder on UNIX or Linux. The default name for this file is **ixbar.***servernum*, where **servernum** is the value of the SERVERNUM configuration parameter.

For example, in an instance with the SERVERNUM configuration parameter equal to 41, the ON-Bar boot file is created by default with this path and name in UNIX:

```
BAR_IXBAR_PATH $INFORMIXDIR/etc/ixbarboot.41
```

You can change the path to create the file in another location. For example, if you want to create the ON-Bar boot file in the directory **/usr/informix**with the name **ixbar.new**, specify:

```
BAR_IXBAR_PATH=/usr/informix/ixbar.new
```

BAR_LOG_COSVR (XPS)

onconfig.std *value* all coservers

range of values A list of unique positive integers greater than or equal to one

takes effect When the database server starts

The BAR_LOG_COSVR configuration parameter specifies the coservers from which the storage manager that BAR_SM specifies can be sent logical-log backup and restore data. BAR_LOG_COSVR is optional. The default is all coservers.

If BAR_LOG_COSVR is set to `0`, the storage manager is not given logical logs from any coserver. You might specify BAR_LOG_COSVR 0 to reserve a storage manager for dbspace backups only.

The values are coserver numbers, separated by commas. Hyphens indicate ranges. For example, BAR_LOG_COSVR 1-5,7,9 specifies a storage manager that backs up logical logs on coservers 1, 2, 3, 4, 5, 7, and 9. Do not put spaces between coserver names.

Extended Parallel Server restricts BAR_LOG_COSVR settings to guarantee that no two storage-manager instances can back up logs for the same coserver.

BAR_MAX_BACKUP (IDS)

onconfig.std *value*	0
if value not present	4
units	**onbar** processes
range of values	0 = maximum number of processes allowed on system 1 = serial backup or restore n = specified number of processes spawned
takes effect	When **onbar** starts

The BAR_MAX_BACKUP parameter specifies the maximum number of parallel processes that are allowed for each **onbar** command. Both UNIX and Windows support parallel backups. Although the database server default value for BAR_MAX_BACKUP is 4, the **onconfig.std** value is 0.

Specifying Serial Backups and Restores

To perform a serial backup or restore, including a serial whole system backup or restore, set BAR_MAX_BACKUP to 1.

Specifying Parallel Backups and Restores

To specify parallel backups and restores, including parallel whole system backups and restores, set BAR_MAX_BACKUP to a value higher than 1. For example, if you set BAR_MAX_BACKUP to 5 and execute an ON–Bar command, the maximum number of processes that ON–Bar will spawn concurrently is 5. Configure BAR_MAX_BACKUP to any number up to the maximum number of storage devices or the maximum number of streams available for physical backups and restores. ON–Bar groups the dbspaces by size for efficient use of parallel resources.

If you set BAR_MAX_BACKUP to 0, the system creates as many ON–Bar processes as needed. The number of ON–Bar processes is limited only by the number of storage spaces or the amount of memory available to the database server, whichever is less.

The amount of memory available is based on SHMTOTAL. ON–Bar performs the following calculation where N is the maximum number of ON–Bar processes that are allowed:

```
N = SHMTOTAL / (# transport buffers * size of transport buffers / 1024)
```

If SHMTOTAL is 0, BAR_MAX_BACKUP is reset to 1. If N is greater than BAR_MAX_BACKUP, ON–Bar uses the BAR_MAX_BACKUP value. Otherwise, ON–Bar starts N backup or restore processes.

BAR_NB_XPORT_COUNT (IDS)

onconfig.std *value*	10
if value not present	10
units	Buffers
range of values	3 to unlimited
takes effect	When **onbar** starts

The BAR_NB_XPORT_COUNT configuration parameter specifies the number of data buffers that each **onbar_d** process can use to exchange data with the database 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 ON–Bar commands, the resulting 25 ON–Bar processes will use a total of 125 buffers.

To calculate the amount of memory that each **onbar_d** process requires, use the following formula. For information on the page size for your system, see the release notes.

```
required_memory = (BAR_NB_XPORT_COUNT * BAR_XFER_BUF_SIZE
                   * page_size) + 5 MB
```

BAR_PERFORMANCE

onconfig.std *value* 0

units Statistics levels

range of values

0	Does not collect performance statistics
1	Reports time spent transferring data between the IDS instance and the storage manager.
2	Reports ON-Bar processing performance, in microseconds, in the timestamps in the activity log and the error log.
3	Reports both microsecond timestamps and transfer statistics.

takes effect When **onbar** starts

The BAR_PERFORMANCE configuration parameter specifies the type of performance statistics to report to the ON–Bar activity log for backup and restore operations. For example, if you set BAR_PERFORMANCE to 3, ON–Bar reports the time spent transferring data between the IDS instance and the storage manager, in the activity log. If you set BAR_PERFORMANCE to 0 or do not set it, ON–Bar does not report performance statistics.

- To turn performance monitoring off, set the value to **0**. This is the default.
- To display the time spent transferring data between the IDS instance and the storage manager, set the parameter to **1**.
- To display timestamps in microseconds, set the parameter to **2**.
- To display both timestamps and transfer statistics, set the parameter to **3**.

BAR_PROGRESS_FREQ

onconfig.std *value* 0

if value not present 0

units Minutes

range of values 0, then 5 to unlimited

takes effect When **onbar** starts

The BAR_PROGRESS_FREQ configuration parameter specifies, in minutes, the frequency of the progress messages in the ON–Bar activity log for backup and restore operations. For example, if you set BAR_PROGRESS_FREQ to 5, ON–Bar reports the percentage of the object backed up or restored every 5 minutes. If you set BAR_PROGRESS_FREQ to 0 or do not set it, ON–Bar does not write any progress messages to the activity log.

Specify a value 5 minutes or over. Do not set BAR_PROGRESS_FREQ to 1, 2, 3, or 4, ON–Bar automatically resets it to 5 to prevent overflow in the ON–Bar activity log.

If ON–Bar cannot determine the size of the backup or restore object, it reports the number of transfer buffers sent to the database server instead of the percentage of the object backed up or restored.

BAR_RETRY

The BAR_RETRY configuration parameter specifies how many times **onbar** should retry a data backup, logical-log backup, or restore operation if the first attempt fails.

BAR_RETRY on Dynamic Server (IDS)

onconfig.std *value*	1
if value not present	1
range of values	BAR_ABORT(0), BAR_CONT(1), or n
takes effect	When **onbar** starts

The setting of the BAR_RETRY parameter determines ON–Bar behavior in the following ways:

- If set to BAR_ABORT, ON–Bar aborts the backup or restore session when an error occurs for a storage space or logical log, returns an error, and quits. If ON–Bar is running in parallel, the already running processes finish but no new ones are started.
- If set to BAR_CONT, ON–Bar aborts the backup or restore attempt for that particular storage space, returns an error, and attempts to back up or restore any storage spaces or logical logs that remain.
- If set to a specific number (n), ON–Bar attempts to back up or restore this storage space or logical log the specified number of times before it gives up and moves on to the next one.

BAR_RETRY on Extended Parallel Server

onconfig.std *value*	0. Does not retry
range of values	0 to 5
takes effect	When the database server starts

If a backup or restore fails, ON–Bar attempts to back up or restore the object the specified number of times before it gives up and moves on to the next object.

The Backup Scheduler maintains two retry counts: object retries and storage-manager retries.

Object retries is the number of times that the Backup Scheduler attempts a backup or restore operation. If the backup or restore of a particular object gets an error, the

Backup Scheduler retries it BAR_RETRY times. If it continues to fail, the Backup Scheduler removes the object from the backup session.

To restart the backup or restore operation:

1. Resolve the error.
2. Issue another ON–Bar command to back up or restore that object.

Storage-manager retries is the number of times that the Backup Scheduler attempts to start an **onbar-worker** process before giving up. If an **onbar-worker** process registers with the Backup Scheduler but exits with an error, then the Backup Scheduler tries to start another **onbar-worker** process, up to BAR_RETRY times. If the **onbar-worker** process fails before it registers or if the Backup Scheduler already has tried to start an **onbar-worker** process BAR_RETRY times, it does not start another **onbar-worker** process.

To reset the storage-manager retry counter and restart the operation:

1. Resolve the error.
2. Start an **onbar-worker** process manually, either directly on the command line or by calling **start_worker.sh** on UNIX or **start_worker.bat** on Windows. The backup or restore operation can then resume.

You can monitor the storage-manager retry count with **onstat -g bus_sm**. For more information, see "Monitoring the Backup Scheduler Status (XPS)" on page 9-11.

BAR_SIZE_FACTOR

onconfig.std *value*	`0`
range of values	Positive integer
takes effect	When the database server starts

Use this parameter to augment the estimate for the size of a backup object, prior to the backup.

The estimate is handled prior to the backup and is calculated so that the storage manager can allocate the storage media appropriately. Because the backup is done online, the number of pages to backup can change during the backup. Some storage managers are very strict and if the backup estimate is too low, the backup will result in an error.

The value of BAR_SIZE_FACTOR is taken as percentage of the original backup object size, and then added to the estimate, before communicating it to the storage manager. BAR_SIZE_FACTOR is used only for dbspace backup objects, not for logical log backup objects.

The formula used for calculating the new estimated backup object size is:

new_estimate = *original_estimate* x (1 + (*BAR_SIZE_FACTOR* / 100))

The value to which this parameter should be set in a specific server environment depends on the activity on the system during backup or archive. Therefore, determining the value needs to be based on the individual experience with that system.

BAR_SM (XPS)

onconfig.std *value*	1
range of values	Positive integer greater than or equal to 1
takes effect	When the database server starts

The required BAR_SM configuration parameter identifies a specific storage-manager instance. Only ON–Bar and the Backup Scheduler use this value. The storage manager does not use this value.

The system uses the BAR_SM to track the location of backups. If you change the identification number after you use the storage manager to perform a backup, you invalidate the backups that you have made. Perform a new level-0 backup of all data.

BAR_SM_NAME (XPS)

onconfig.std *value*	None
range of values	Any character string that does not contain white spaces or the pound sign (#)
takes effect	When the database server starts

The BAR_SM_NAME configuration parameter is the name of the storage manager. It can be up to 128 characters.

BAR_WORKER_COSVR (XPS)

onconfig.std *value*	1
if value not present	an error occurs
range of values	A list or range of unique positive integers greater than or equal to one
takes effect	When the database server starts

The required BAR_WORKER_COSVR configuration parameter specifies which coservers can access the storage manager that BAR_SM identifies. If BAR_SM is specified, BAR_WORKER_COSVR must also be specified. Any coserver on the list can restore data that other coservers on the list back up.

Enter the numbers of the coservers where **onbar-worker** processes can be started for this storage manager. If you enter only one coserver number, use the number of a coserver on a node with a physically attached storage device so that the storage manager does not have to transfer data across the network. This step improves performance.

The list must not overlap with the list of any other storage manager. The values are coserver numbers, separated by commas. Hyphens indicate ranges. For example, BAR_WORKER_COSVR 1-3,5,7 specifies a storage manager that can access **onbar-worker** processes running on coservers 1, 2, 3, 5, and 7. Do not put spaces between coserver names.

BAR_WORKER_MAX (XPS)

onconfig.std *value*	0

if value not present	0
takes effect	When the database server starts

The BAR_WORKER_MAX configuration parameter determines how many storage spaces and logical logs can be backed up or restored in parallel on each storage manager. It also specifies the maximum number of **onbar-worker** processes that the database server automatically starts for this storage-manager instance. If the BAR_WORKER_MAX value is 0, you must start the **onbar-worker** processes manually.

Specifying a Parallel Backup or Restore

For example, to start five **onbar-worker** processes in parallel, set BAR_WORKER_MAX to 5. Set BAR_WORKER_MAX to the number of storage devices available to the storage manager. If the database server has multiple storage managers configured, the number of parallel operations is the sum of BAR_WORKER_MAX values for all storage managers. The maximum number of **onbar-worker** processes that run simultaneously depends on the capabilities of the storage manager.

If **onbar-worker** processes for a specific storage manager have special startup requirements, such as environment variables, you can specify these by editing the **start_worker** script file. For information, see "Using start_worker.sh to Start onbar_worker Processes Manually (XPS)" on page 9-4. If storage managers have dynamic requirements for **onbar-worker** processes, you might have to start them manually.

Specifying a Serial Backup or Restore

You can specify serial backups and restores with Extended Parallel Server in two ways:

- Set BAR_WORKER_MAX to 0 and manually start one **onbar-worker** process.

  You must start the **onbar-worker** processes manually before you can back up or restore data.
- Set BAR_WORKER_MAX to 1.

  The database server starts one **onbar-worker** process and backs up or restores the data serially for this storage-manager instance.

BAR_XFER_BUF_SIZE (IDS)

onconfig.std *value*	31 if the PAGESIZE is 2 kilobytes 15 if the page size is 4 kilobytes
units	pages
range of values	1 to 15 pages when the PAGESIZE is 4 kilobytes 1 to 31 pages when the PAGESIZE is 2 kilobytes The maximum buffer size is 64 kilobytes, so BAR_XFER_BUF_SIZE * pagesize <= 64 kilobytes
takes effect	When **onbar** starts

The BAR_XFER_BUF_SIZE configuration parameter specifies the size of each transfer buffer. The database server passes the buffer to ON–Bar and the storage manager. To calculate the size of the transfer buffer in a storage space or logical-log backup, use the following formula:

```
transfer buffers = BAR_XFER_BUF_SIZE * pagesize
```

Where *pagesize* is the largest page size used by any of the dbspaces that are backed up.

To calculate how much memory the database server needs, use the following formula:

```
memory = (BAR_XFER_BUF_SIZE * PAGESIZE) + 500
```

The extra 500 bytes is for overhead. For example, if BAR_XFER_BUF_SIZE is 15, the transfer buffer should be 61,940 bytes.

Important: You cannot change the buffer size between a backup and restore. AC_TAPEBLOCK and AC_LTAPEBLOCK need to be same value as BAR_XFER_BUFSIZE was at the time of archive.

BAR_XFER_BUFSIZE (XPS)

onconfig.std *value*	8 pages
if value not present	32,768 / pagesize for platform
units	PAGESIZE which can be 2, 4, or 8 kilobytes
range of values	1 to 15 pages
takes effect	When the **onbar_w** utility starts

The BAR_XFER_BUFSIZE configuration parameter specifies the size of each transfer buffer. The actual size of a transfer buffer is BAR_XFER_BUFSIZE * PAGESIZE + 500. The extra 500 is for overhead. For example, if BAR_XFER_BUFSIZE is 15 and the page size is 4 kilobytes, the transfer buffer should be 61,440 bytes.

For generally good performance, set to 8, although different storage managers might suggest other values. The maximum value that XBSA allows is 64 kilobytes.

If you set BAR_XFER_BUFSIZE to 8 when the page size is 8 kilobytes, the database server decrements the transfer buffer size to 7 pages so that the maximum value would be under 64 kilobytes (7 * 8 = 56). ON–Bar displays a warning message if the transfer buffer size is decremented.

To calculate how much memory the database server needs, use the following formula:

```
kilobytes = BAR_WORKER_MAX * BAR_XFER_BUFSIZE *
            BAR_XPORT_COUNT * PAGESIZE
```

For example, you would need at least 1600 kilobytes of memory for the transfer buffers for five **onbar-worker** processes if the page size is 4 kilobytes.

```
1600 kilobytes = 5 workers * 8 pages* 10 buffers * 4 kilobytes
```

Important: You cannot change the buffer size (value of BAR_XFER_BUFSIZE) between a backup and restore.

BAR_XPORT_COUNT (XPS)

onconfig.std *value*	10
if value not present	10
units	Buffers

range of values	3 to unlimited
takes effect	When an **onbar-worker** process starts

The BAR_XPORT_COUNT configuration parameter specifies the number of data buffers that each **onbar-worker** process can use to exchange data with Extended Parallel Server. The value of this parameter affects **onbar-worker** performance.

To calculate the amount of memory that each **onbar_w** process requires, use the following formula:

```
required_memory = (BAR_XPORT_COUNT * BAR_XFER_BUFSIZE
   * page_size) + 5 MB
```

For information on configuring the page size, see the *IBM Informix Administrator's Guide*.

ISM_DATA_POOL

onconfig.std *value*	ISMData
takes effect	When **onbar** starts

The ISM_DATA_POOL parameter, when listed in the **ONCONFIG** file for the database server, specifies the volume pool that you use for backing up storage spaces. The value for this parameter can be any volume pool that ISM recognizes. If this parameter is not present, ISM uses the ISMData volume pool. For details, see the *IBM Informix Storage Manager Administrator's Guide*.

Extended Parallel Server

Insert the ISM_DATA_POOL parameter inside the BAR_SM paragraph in the storage-manager section of the **ONCONFIG** file if you want it to apply to one storage-manager instance. Insert ISM_DATA_POOL in the global section of the **ONCONFIG** file if you want it to apply to all storage-manager instances.

End of Extended Parallel Server

For more information, see "Files That ON-Bar, ISM, and TSM Use" on page 10-22.

ISM_LOG_POOL

onconfig.std *value*	ISMLogs
takes effect	When **onbar** starts

The ISM_LOG_POOL parameter, when listed in the **ONCONFIG** file for the database server, specifies the volume pool that you use for backing up logical logs. The value for this parameter can be any volume pool that ISM recognizes. If this parameter is not present, ISM uses the ISMLogs volume pool. For details, see the *IBM Informix Storage Manager Administrator's Guide*.

Extended Parallel Server

Insert the ISM_LOG_POOL parameter inside the BAR_SM paragraph in the storage-manager section of the **ONCONFIG** file if you want it to apply to one storage-manager instance. Insert ISM_LOG_POOL in the global section of the

ONCONFIG file if you want it to apply to all storage-manager instances.

End of Extended Parallel Server

For more information, see "Files That ON-Bar, ISM, and TSM Use" on page 10-22.

LOG_BACKUP_MODE (XPS)

Use the LOG_BACKUP_MODE configuration parameter to determine how logical-log files are backed up after they fill.

onconfig.std *value* MANUAL

range of values

CONT Continuous = an **onbar-worker** process backs up each logical-log file as soon as it fills.

MANUAL Manual = queues the logical-log files until you can issue an **onbar -b -l** command.

NONE Use the NONE option if you do not want to back up the logs. The database server marks the logical logs as backed up as soon as they are full so that ON–Bar cannot back them up. When the database server starts up, it writes a message to the **online.log** if LOG_BACKUP_MODE = NONE.

takes effect When the database server restarts

Warning: If you set LOG_BACKUP_MODE to NONE, you cannot back up logs. All transactions in those logs are lost, and you will not be able to restore them. Your physical backups also will not be restorable.

LTAPEDEV (IDS)

If you specify a tape device in the LTAPEDEV configuration parameter, ON–Bar ignores the value. ON–Bar also ignores the value of the LTAPEBLK, LTAPESIZE, TAPEDEV, TAPEBLK, and TAPESIZE parameters. Consider leaving these parameter values blank when you use ON–Bar because the storage manager sets these values.

Warning: Set LTAPEDEV to **/dev/null** or leave it blank on UNIX or **\dev\nul** or NUL on Windows only if you do **not** want to back up the logical logs. The ON–Bar activity log will display a warning and return code 152. Because the database server marks the logical logs as backed up when they are no longer current, ON–Bar cannot find logical logs to back up. All transactions in those logs are lost, and you will not be able to restore them.

If you performed a whole-system backup with LTAPEDEV set to null, you must use the **onbar -r -w -p** command during restore to notify ON–Bar that you do not want to restore the logs. For more information, see "Considerations When LTAPEDEV is Set to Null" on page 7-18.

RESTARTABLE_RESTORE (IDS)

Use the RESTARTABLE_RESTORE configuration parameter to enable or disable restartable restores.

onconfig.std *value*	ON
if value not present	ON
range of values	
	OFF Disables restartable restore. If a restore fails and RESTARTABLE_RESTORE is OFF, you will not be able to restart it.
	ON Enables restartable restore. Set RESTARTABLE_RESTORE to ON before you begin a restore. Otherwise, you will be unable to restart the restore after a failure.
takes effect	If you need to restart a physical restore, you do not need to restart the database server before you can use RESTARTABLE_RESTORE. If you need to restart a logical restore, you must restart the database server before you can use restartable restore.

Turning on RESTARTABLE_RESTORE slows down logical restore performance. For more information, see "Using Restartable Restore to Recover Data (IDS)" on page 7-35. For information on the physical log, see the *IBM Informix Administrator's Guide*.

RESTORE_FILTER

onconfig.std *value*	none
if value not present	none
range of values	Full path name of command and any options.
takes effect	When ON-Bar starts

Set the RESTORE_FILTER configuration parameter to specify the path name of a filter program, and any options. For example, specify:

```
RESTORE_FILTER  /bin/uncompress
```

In this example:

- The filter will be called from ON–Bar as:
  `/bin/uncompress`
- The data passed to the filter was produced by backup filter.

The RESTORE_FILTER configuration parameter can include command line options as well as the filter name. If you include command line options, both the filter name and the options must be surrounded by single quotation marks. For example, specify:

```
RESTORE_FILTER 'my_decrypt –file /var/adm/encryption.pass'
```

In this example, the command in quotation marks is used as the filter.

For security purposes, filters should not have write permission to non-privileged users. Permission on the filters will be same as that of permission on other executable files that are called by the IDS server or utilities.

For more information, see "Transforming with Filters during Backup and Restore" on page 3-12.

Files That ON-Bar, ISM, and TSM Use

Table 10-2 lists the files that ON–Bar, ISM, and TSM use and the directories in which they reside. These names and locations change if you set up the **ONCONFIG** file to values different than the defaults.

Table 10-2. List of Files That ON-Bar, ISM, and TSM Use

Filename	Directory	Purpose
ac_config.std	**$INFORMIXDIR/etc, %INFORMIXDIR%\etc**	Template for **archecker** parameter values. The **ac_config.std** file contains the default **archecker** (archive checking) utility parameters. To use the template, copy it into another file and modify the values.
ac_msg.log	**/tmp**, **%INFORMIXDIR%\etc**	**archecker** message log. When you use **archecker** with ON–Bar to verify a backup, it writes brief status and error messages to the ON–Bar activity log and writes detailed status and error messages to the **archecker** message log. Technical Support uses the **archecker** message log to diagnose problems with backups and restores. Specify the location of the **archecker** message log with the AC_MSGPATH configuration parameter.
bar_act.log	**/tmp**, **%INFORMIXDIR%**	ON–Bar activity log. For more information, see "ON-Bar Activity Log" on page 3-11.
bldutil.*process_id*	**/tmp**, **\tmp**	When the **sysutils** database is created, error messages appear in this file.
dsierror.log	**$DSMI_LOG**	TSM API error log.
dsm.opt	**$DSMI_CONFIG**	TSM client user option file.
dsm.sys	**$DSMI_DIR**	TSM client system option file.
Emergency boot files (**ixbar*** files)	**$INFORMIXDIR/etc, %INFORMIXDIR%\etc**	Used in a cold restore. For more information, see "ON-Bar Boot Files" on page 3-10.
ISM catalog	**$INFORMIXDIR/ism, %ISMDIR%**	Records information about backup and restore save sets and storage volumes that ISM uses. ISM creates the ISM catalog during the **ism_startup** initialization. The ISM catalog records are stored in the **mm**, **index**, and **res** files. For more information, see the *IBM Informix Storage Manager Administrator's Guide*.
ISM logs	**$INFORMIXDIR /ism/logs**, **%ISMDIR%\logs**	Operator alert messages, backend status, additional ISM information. The ISM log names are daemon.log, messages, and summary.
ISMversion	**$INFORMIXDIR/ism, %ISMDIR%\bin**	Identifies the ISM version. Do not edit this file.

Table 10-2. List of Files That ON-Bar, ISM, and TSM Use (continued)

Filename	Directory	Purpose
oncfg_*servername.servernum* (IDS)	**$INFORMIXDIR/etc, %INFORMIXDIR%\etc**	Configuration information for ON–Bar restores. The database server creates the **oncfg**_***servername.servernum*** file when you initialize disk space. The database server updates the file every time that you add or delete a dbspace, a logical-log file, or a chunk. The database server uses the **oncfg*** file when it salvages logical-log files during a cold restore. The database server uses the **oncfg*** files, so do not delete them.
oncfg_*servername.servernum.coserverid* (XPS)	**$INFORMIXDIR/etc, %INFORMIXDIR%\etc**	Configuration information for ON–Bar restores. On XPS, the **oncfg*** files are on each coserver.
save, **savegrp**, **savefs**	**$INFORMIXDIR/bin, %ISMDIR%\bin**	ISM commands use these executable files. Do not edit them.
sm_versions	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Identifies storage manager in use. To update the storage-manager version, edit the **sm_versions** file directly or run the **ism_startup** script. For more information, see "Updating the sm_versions File" on page 4-4.
xbsa.messages	**$INFORMIXDIR /ism/applogs, %ISMDIR%\applogs**	XBSA library call information. ON–Bar and ISM use XBSA to communicate with each other. Technical Support uses the **xbsa.messages** log to fix problems with ON–Bar and ISM communications.
xcfg_*servername.servernum* (XPS)	**$INFORMIXDIR/etc, %INFORMIXDIR%\etc**	Internal server configuration information. The **xcfg*** file contains information about coserver location and dbslice definitions.

Chapter 11. ON-Bar Catalog Tables

In This Chapter 11-1
bar_action 11-2
bar_instance 11-2
bar_ixbar 11-4
bar_object 11-6
bar_server 11-6
bar_syncdeltab 11-6
ON-Bar Catalog Map 11-7
Backup Scheduler SMI Tables (XPS) 11-8
sysbuobject 11-9
sysbuobjses 11-9
sysbusession 11-10
sysbusm 11-10
sysbusmdbspace 11-10
sysbusmlog 11-10
sysbusmworker 11-11
sysbuworker 11-11

In This Chapter

This chapter describes the ON–Bar tables that are stored in the **sysutils** database. ON–Bar uses these tables for tracking backups and performing restores. You can query these tables for backup and restore data to evaluate performance or identify object instances for a restore.

© Copyright IBM Corp. 1996, 2008

bar_action

The **bar_action** table lists all backup and restore actions that are attempted against an object, except during certain types of cold restores. Use the information in this table to track backup and restore history.

Column Name	Type	Explanation
act_aid	SERIAL	Action identifier. A unique number within the table. Can be used with **act_oid** to join with the **bar_instance** table.
act_oid	INTEGER	Object identifier. Identifies the backup object against which a backup or restore attempt is 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.
act_type	SMALLINT	Identifies the attempted action: `1` for backup, `2` for restore, `3` for a foreign or imported restore, `4` for a fake backup, `5` for a whole-system backup, `6` for a whole-system restore, `7` for expired or deleted objects, `8` for an external restore. Fake backups, whole-system backups, and whole-system restores are available on IDS only.
act_status	INTEGER	Identifies the result of the action: `0` if successful, otherwise an ON-Bar-specific error code. For more information, see Chapter 12, "ON-Bar Messages and Return Codes," on page 12-1.
act_start	DATETIME YEAR TO SECONDS	The date and time when the action began.
act_end	DATETIME YEAR TO SECONDS	The date and time when the action finished.

bar_instance

ON–Bar writes a record to the **bar_instance** table for each successful backup. The table describes each object that is backed up. ON–Bar might later use the information for a restore operation. For example, if you specify a level-2 backup, ON–Bar uses this table to ensure that a level-1 backup was done previously.

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_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_prevtime	INTEGER	Timestamp (real clock time). This value specifies the timestamp of the previous object. The value represents the number of seconds after midnight, January 1, 1970 Greenwich mean time.

Column Name	Type	Explanation
ins_time	INTEGER	Time stamp (real clock time). The database server uses this value when it creates the next-level backup. Value represents the number of seconds since midnight, January 1, 1970, Greenwich mean time. For IDS, the **ins_time** value is `0`.
rsam_time	INTEGER	The backup checkpoint time stamp. Not a clock time. The database server uses this value when it creates the next level backup.
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. This value is always `0` for logical-log backups.
ins_copyid_hi	INTEGER	The high bits of the instance copy identifier. Combined with **ins_copyid_lo**, it is a unique value that the storage manager assigns 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 the storage manager assigns to link the ON-Bar object identifier with the storage-manager object identifier.
ins_req_aid	INTEGER	Stores the required action ID for a backup object. Used in a restore to determine which level-0 backup goes with the level-1 backup, and which level-1 backup goes with the level-2 backup. For a level-0 backup, the value of **ins_req_aid** is the same as **ins_aid** in this table. For example, if this backup is level-1, **ins_req_aid** holds the action ID of the corresponding level-0 backup of this object.
ins_logstream (XPS only)	INTEGER	The coserver ID from which this object came.
ins_first_log	INTEGER	In a standard backup, identifies the first logical log required to restore from this backup.
ins_chpt_log (XPS only)	INTEGER	The ID of the log that contains the **rsam_time** checkpoint. Used during backup to verify that logs needed for restore are backed up.
ins_last_log (XPS only)	INTEGER	Log ID of the last log needed during logical restore for this storage space to restore it to the time of the backup.
ins_lbuflags (XPS only)	INTEGER	Flag field storing attributes for log backups.
ins_sm_id (XPS only)	INTEGER	Storage-manager instance ID. Created from BAR_SM in **$ONCONFIG** or **%ONCONFIG%**.
ins_sm_name (XPS only)	VARCHAR(128, 0)	Storage-manager instance name. Created from the BAR_SM_NAME parameter in the ONCONFIG file.
ins_verify	INTEGER	Value is 1 if the backup is verified. Value is 0 if the backup is not verified.

Column Name	Type	Explanation
ins_verify_date	DATETIME YEAR TO SECOND	The current date is inserted when a backup is verified. If this backup has not been not verified, a dash represents each date and time.

bar_ixbar

The **bar_ixbar** table is maintained and used by **onsmsync** only. It keeps a history of all unexpired successful backups in all timelines. The schema of the **bar_ixbar** table is identical to the schema of the **bar_syncdeltab** table, with the exception of its primary key.

Column Name	Type	Explanation
ixb_sm_id	INTEGER	Storage-manager instance ID. Created from BAR_SM in $ONCONFIG or %ONCONFIG%.
ixb_copyid_hi	INTEGER	The high bits of the instance copy identifier. Combined with **ixb_copyid_lo**, it is a unique value that the storage manager assigns to link the ON-Bar object identifier with the storage-manager object identifier.
ixb_copyid_lo	INTEGER	The low bits of the instance copy identifier. Combined with **ixb_copyid_hi**, it is a unique value that the storage manager assigns to link the ON-Bar object identifier with the storage-manager object identifier.
ixb_aid	INTEGER	Action identifier, Identifies the successful action that created this instance of the backup object.
ixb_old	INTEGER	Object identifier. Identifies the affected object.
ixb_time	INTEGER	Time stamp (real clock time). The database server uses this value when it creates the next-level backup. Value represents the number of seconds since midnight, January 1, 1970, Greenwich mean time.
ixb_prevtime	INTEGER	Time stamp (real clock time). This value specifies the time stamp of the previous object. Value represents the number of seconds since midnight, January 1, 1970 Greenwich mean time.
ixb_rsam_time	INTEGER	The backup checkpoint time stamp. Not a clock time. The database server uses this value when it creates the next level backup.
ixb_act_start	datetime year to second	The date and time when the action began.
ixb_act_end	datetime year to second	The date and time when the action finished.
ixb_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. This value is always 0 for logical-log backups.

Column Name	Type	Explanation
ixb_req_aid	INTEGER	Stores the required action ID for a backup object. Used in a restore to determine which level-0 backup Goes with the level-1 backup, and which level-1 backup goes with the level-2 backup. For a level-0 backup, the value of **ixb_req_aid**is the same as **ixb_aid** in this table. For example, if this backup is level-1, **ixb_req_aid** holds the action ID of the corresponding level-0 backup of this object.
ixb_logstream (XPS only)	INTEGER	The coserver ID from which this object came.
ixb_first_log	INTEGER	In a standard backup, identifies the first logical log. Required to restore from this backup.
ixb_chpt_log	INTEGER	The ID of the log that contains the **rsam_time** checkpoint. Used during backup to verify that logs needed for restore are backed up.
ixb_last_log	INTEGER	Log ID of the last log needed during logical restore for this storage space to restore it to the time of the backup.
ixb_lbuflags	INTEGER	Flags describing log backup.
ixb_verify	INTEGER	Value is 1 if the backup is verified. Value is 0 if the backup is not verified.
ixb_verify_date	datetime year to second	The current date is inserted when a backup is verified. If this backup has not been verified, a dash represents each date and time.
ixb_sm_name	VARCHAR(128)	Storage-manager instance name. Created from the BAR_SM_NAME parameter in the ONCONFIG file.
ixb_srv_name	VARCHAR(128)	The database server name. Used to ensure that objects are restored to the correct database 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 database server name can be up to 128 characters.
ixb_obj_name	VARCHAR(128)	The user name for the object. The name can be up to 128 characters. On XPS 15:3 is the name of log file 3 on coserver 15.
ixb_obj_type	CHAR(2)	Backup object type: • CD = critical dbspace • L = logical log • ND = noncritical dbspace or sbspace • R = rootdbs • B = blobspace (IDS only)

bar_object

The **bar_object** table describes each backup object. This table is a list of all storage spaces and logical logs from each database server for which at least one backup attempt was made.

Column Name	Type	Explanation
obj_srv_name	VARCHAR(128,0)	The database server name. Used to ensure that objects are restored to the correct database 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 database server name can be up to 128 characters.
obj_oid	SERIAL	The object identifier. A unique number within the table. Can be used to join with the **bar_action** and **bar_instance** tables.
obj_name	VARCHAR(128,0)	The user name for the object. The name can be up to 128 characters. On XPS, **15:3** is the name of log file 3 on coserver 15.
obj_type	CHAR(2)	Backup object type: `CD` = critical dbspace `L` = logical log `ND` = noncritical dbspace or sbspace `R` = rootdbs `B` = blobspace (Dynamic Server only)

bar_server

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.

Column Name	Type	Explanation
srv_name	VARCHAR(128,0)	DBSERVERNAME value specified in the ONCONFIG file. Database server name can be up to 128 characters.
srv_node	CHAR(256)	Host name of the computer where the database server resides. The host name can be up to 256 characters.
srv_synctime	INTEGER	The time **onsmsync** was run.

bar_syncdeltab

The **bar_syncdeltab** table is maintained and used by the **onsmsync** utility only. It is normally empty except when **onsmsync** is running. The schema of the **bar_syncdeltab** table is identical to the schema of the **bar_ixbar** table, with the exception of its primary key.

ON-Bar Catalog Map

Dynamic Server

Figure 11-1 maps the ON–Bar tables on Dynamic Server. The gray lines show the relations between tables. The arrows show that the **ins_req_aid** value must be a valid **ins_aid** value.

End of Dynamic Server

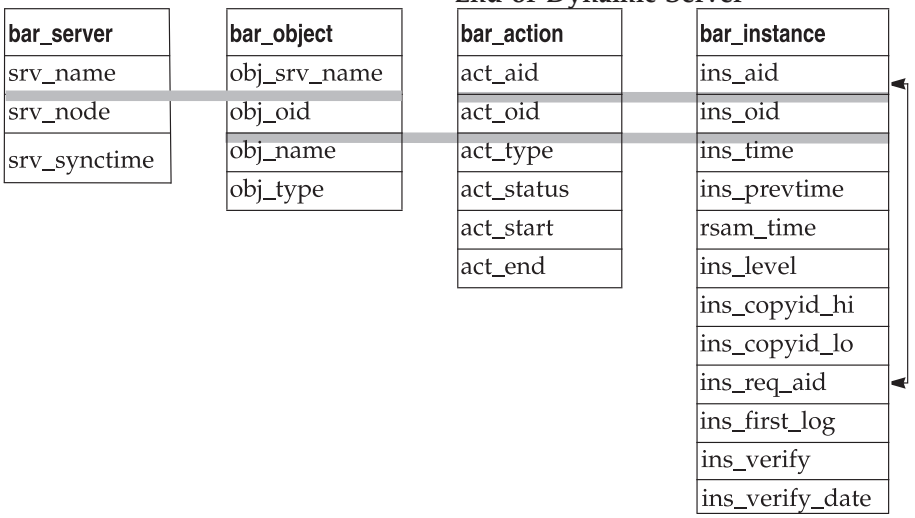


Figure 11-1. ON-Bar Catalog Map on Dynamic Server

Extended Parallel Server

Figure 11-2 maps the ON–Bar tables on Extended Parallel Server. The gray lines in the table show the relations between tables. The arrows show that the **ins_req_aid** value must a valid **ins_aid** value.

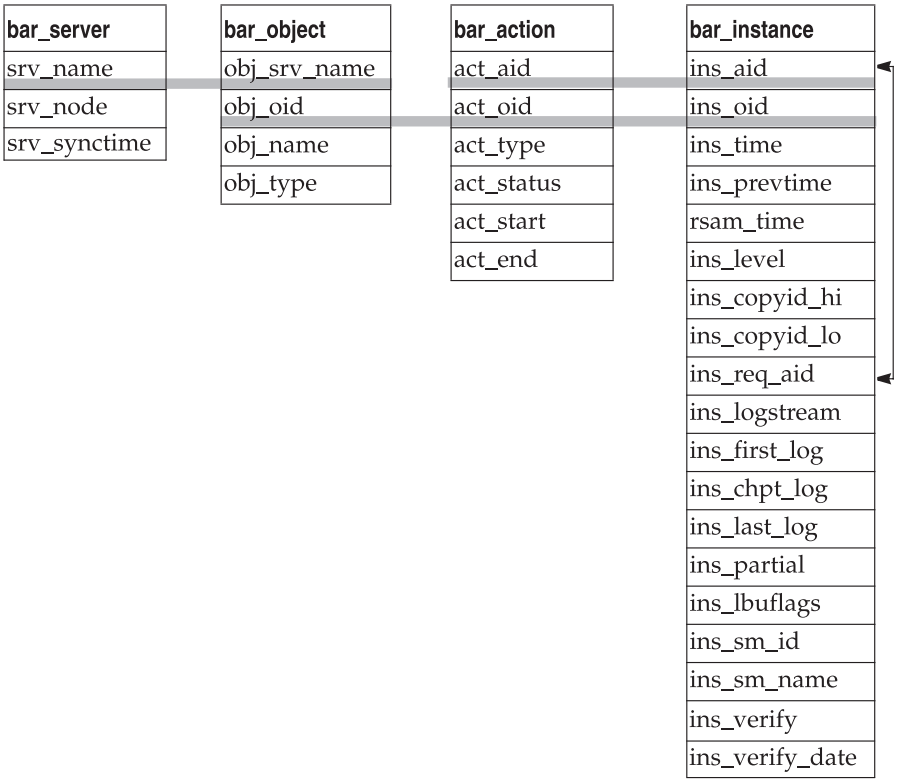


Figure 11-2. ON-Bar Catalog Map on Extended Parallel Server

End of Extended Parallel Server

Backup Scheduler SMI Tables (XPS)

The following system-monitoring interface (SMI) pseudo-tables in the **sysmaster** database contain information about the Backup Scheduler. For information about other SMI tables, see the *IBM Informix Administrator's Reference*.

Table	Description	Reference
sysbuobject	Detailed information about backup objects that the Backup Scheduler is processing	11-9
sysbuobjses	Backup object name and session ID	11-9
sysbusession	Session status information	11-10
sysbusm	Storage-manager configuration information	11-10
sysbusmdbspace	Storage manager for backing up dbspaces on a coserver	11-10
sysbusmlog	Storage manager for backing up logs on a coserver	11-10
sysbusmworker	Coservers where **onbar-worker** processes can access a storage manager	11-11
sysbuworker	Detailed **onbar-worker** status	11-11

sysbuobject

The **sysbuobject** table provides information about the backup and restore objects that the Backup Scheduler is processing.

Column	Type	Description
owner	INTEGER	The coserver owner of the object
priority	INTEGER	Priority in the Backup Scheduler queue for this session
order_hi	INTEGER	The high-order word of the restore order for the object (high 32 bits of a 64-bit number)
order_lo	INTEGER	The low-order word of the restore order for the object (low 32 bits of a 64-bit number)
placement	INTEGER	The storage-manager instance where this object is located
retries	INTEGER	Number of times that the backup or restore is retried
timestamp	INTEGER	The global time for the object
dbsname	CHAR(128)	The dbspace name
level	INTEGER	The backup level of the dbspace
log_num	INTEGER	The logical-log number
is_backup	INTEGER	A value of 1 means this object is to be backed up.
is_restore	INTEGER	A value of 1 means this object is to be restored.
is_check	INTEGER	A value of 1 means this object is to be verified with the **archecker** utility.
is_block	INTEGER	A value of 1 means the coserver is blocked for an external backup.
is_external	INTEGER	A value of 1 means an external restore of a dbspace is in progress.
is_dbspace	INTEGER	A value of 1 means this object is a dbspace.
is_log	INTEGER	A value of 1 means this object is a logical-log file.
is_coserver	INTEGER	A value of 1 means this object is a coserver (used for external backup and restore).
is_running	INTEGER	A value of 1 means this object is being processed.
is_ready	INTEGER	A value of 1 means this object is ready to be processed.
is_waiting	INTEGER	A value of 1 means this object cannot be processed yet.
is_suspend	INTEGER	A value of 1 means processing of this object is suspended.
is_cold	INTEGER	A value of 1 means this object is to be cold-restored.

sysbuobjses

The **sysbuobjses** table lists the backup object name and session ID. You can join the **sysbuobjses** table with the **sysbusession** table on the **ses_id** column to get all the backup objects in a specific session.

Column	Type	Description
obj_name	CHAR(149)	Object name
ses_id	CHAR(128)	Name of the backup or restore session

sysbusession

The **sysbusession** table shows the status of the current backup or restore session. The information in the **sysbusession** table is similar to the **onstat -g bus** output.

Column	Type	Description
id	CHAR(128)	Name of the backup or restore session.
count	INTEGER	Number of objects in the backup or restore session.
is_complete	INTEGER	A value of 1 means the session is complete.
is_waiting	INTEGER	A value of 1 means the session is waiting for work.
is_suspended	INTEGER	A value of 1 means the session is suspended.
error	INTEGER	Error value that is returned to the ON–Bar driver.

sysbusm

The **sysbusm** table provides information about the storage-manager configuration, storage-manager retry status, and active **onbar-worker** processes.

Column	Type	Description
id	INTEGER	ID of this storage-manager instance.
name	CHAR(128)	Storage-manager name.
worker	INTEGER	Number of **onbar-worker** processes.
max_idle	INTEGER	The idle time-out value. If an **onbar-worker** has been idle the specified time, it is terminated.
retries	INTEGER	Times to retry the backup or restore for a storage manager.
pending	INTEGER	Number of pending **onbar-worker** processes.
registered	INTEGER	Number of registered **onbar-worker** processes for this storage manager.

sysbusmdbspace

The **sysbusmdbspace** table provides information about which storage manager is configured to back up dbspaces for a coserver. This table contains the value of the BAR_DBS_COSVR parameter.

Column	Type	Description
id	INTEGER	Storage-manager ID.
cosvr_id	INTEGER	Dbspaces on this coserver are backed up or restored to this storage manager.

sysbusmlog

The **sysbusmlog** table provides information about which storage manager is configured to back up logical logs for a coserver. This table contains the value of the BAR_LOG_COSVR parameter.

Column	Type	Description
id	INTEGER	Storage-manager ID.
cosvr_id	INTEGER	Logical logs on this coserver are backed up or restored to this storage manager.

sysbusmworker

The **sysbusmworker** table lists the coserver IDs of **onbar-worker** processes that are configured for a storage manager. To display the list of active **onbar-worker** processes for a storage manager, join the **sysbuworker** table with the **sysbusmworker** table (`sysbusmworker.id = sysbuworker.sm`). This table contains the same list of storage managers as the BAR_WORKER_COSVR parameter. For example, use the **SELECT *** FROM **sysbusmworker** WHERE **id = 1** command to display information on the storage manager listed in BAR_SM 1.

Column	Type	Description
id	INTEGER	Storage-manager ID.
cosvr_id	INTEGER	Coserver on which the **onbar-worker** is running.

sysbuworker

The **sysbuworker** table provides status information about **onbar-worker** processes.

Column	Type	Description
id	INTEGER	ID of the **onbar-worker**.
cosvr	INTEGER	Coserver on which the **onbar-worker** is running.
pid	INTEGER	Process ID.
sm	INTEGER	Storage-manager ID.
obj_name	CHAR(149)	Name of the current backup or restore object.
is_new	INTEGER	A value of 1 means this **onbar-worker** is initializing.
is_free	INTEGER	A value of 1 means this **onbar-worker** is free for another job.
is_wait	INTEGER	A value of 1 means this **onbar-worker** is waiting for its next task.
is_busy	INTEGER	A value of 1 means this **onbar-worker** is processing an object.
is_dead	INTEGER	A value of 1 means this **onbar-worker** is dead.
event_dbu	INTEGER	A value of 1 means this **onbar-worker** can accept dbspace backup jobs.
event_phr	INTEGER	A value of 1 means this **onbar-worker** can accept dbspace restore jobs.
event_lbu	INTEGER	A value of 1 means this **onbar-worker** can accept logical backup jobs.
event_lgr	INTEGER	A value of 1 means this **onbar-worker** can accept logical restore jobs.
event_lbuplace	INTEGER	A value of 1 means this **onbar-worker** determines the correct storage manager to handle logical backups.
event_phrplace	INTEGER	A value of 1 means this **onbar-worker** determines the correct storage manager to handle dbspace restores.
event_dbplace	INTEGER	A value of 1 means this **onbar-worker** determines the correct storage manager to handle dbspace backups.
event_lgrplace	INTEGER	A value of 1 means this **onbar-worker** determines the correct storage manager to handle logical restores.

Column	Type	Description
event_cold	INTEGER	A value of 1 means this **onbar-worker** can process an object that is part of a cold restore.
idle_time	INTEGER	How long this **onbar-worker** has been idle.

Chapter 12. ON-Bar Messages and Return Codes

In This Chapter 12-1
About ON-Bar Messages 12-1
Message Format 12-1
Message Numbers 12-2
ON-Bar Usage Messages 12-2
ON-Bar Return Codes 12-6

In This Chapter

The first part of this chapter describes the ON–Bar activity log file and the ON–Bar usage messages. Both Dynamic Server and Extended Parallel Server share common ON–Bar messages.

The second part of this chapter describes the ON–Bar return codes.

For information on ON–Bar informational, progress, warning, and error messages, use the **finderr** or **Error Messages** utility or view *IBM Informix Error Messages* at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp.

About ON-Bar Messages

This section explains how to read and interpret messages in the ON–Bar activity log file. For more information, see "ON-Bar Activity Log" on page 3-11.

Message Format

A message in the ON–Bar activity log file has the following format:

```
timestamp process_id parent_process_id message
```

Table 12-1 describes each field in the message. No error message numbers appear in the ON–Bar activity log.

Table 12-1. ON-Bar Message Format

Message Field	Description
timestamp	Date and time when ON-Bar writes the message.
process_id	The number that the operating system uses to identify this instance of ON-Bar.
parent_process_id	The number that the operating system uses to identify the process that executed this instance of ON-Bar.
message	The ON-Bar message text.

The following example illustrates a typical entry in the ON–Bar activity log:

```
1999-08-18 10:09:59 773     772  Completed logical restore.
```

Important: If you receive an XBSA error message, consult the storage-manager logs for more details.

© Copyright IBM Corp. 1996, 2008

Message Numbers

The ON–Bar message numbers range from -43000 to -43421. The following table lists the ON–Bar message groups. Because message numbers do not display in the activity log, the best way to find information about ON–Bar messages is to search for the message text in the error messages file, which is located in the subdirectory for your locale under the **$INFORMIXDIR/msg directory. You can search for error messages in English in** *IBM Informix Error Messages* at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp.

ON–Bar Message Type	Message Numbers
ON–Bar usage	-43000 to -43424
Options checking	-43010 to -43034
Permission checking	-43035 to -43039
Emergency boot file interface	-43040 to -43059
ONCONFIG file interface	-43060 to -43074
Operating system interface	-43075 to -43099
Database server interface	-43100 to -43229
Backup and restore status	-43230 to -43239
Onbar-worker processes	-43240 to -43254
XBSA interface	-43255 to -43301
onsmsync	-43302 to -43319
archecker	-43320 to -43334
ondblog	-43400 to -43424

ON-Bar Usage Messages

This section lists usage messages only. All informational, progress, warning, and error messages appear in **finderr** and *IBM Informix Error Messages* at http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp.

Important: You must specify the **-b**, **-v**, **-r** or **-m** option first in the command so that ON–Bar can determine whether it is performing a backup, verification, restore, message display. The only exception is when you start or stop the ON–Bar session command on Extended Parallel Server.

-43000:

ON–Bar backup and verification usage.

Dynamic Server

```
For Dynamic Server
-b [-L level] [-w | -f filename | spaces] [-O]
-b -l [-c | -C | -s] [-O]
-v [-p] [-t time] [-f filename | spaces]

    -b back up
    -c back up current logical log
    -C continuous logical-log backup
    -f pathname of file containing list of storage spaces
    -F fake backup
    -l back up full logical logs (no spaces)
```

```
    -L back up level: 0, 1, or 2; defaults to 0
    -O override internal error checks - enforce policy strictly
    -w whole-system backup
    -s salvage logs
    -v verify consistency of most recent backups
    spaces list of storage spaces
```

End of Dynamic Server

Extended Parallel Server

```
For IBM Informix Extended Parallel Server:
-b [-S] [-v] [-p] [-q <name>] [-L <level>] [-f <filename> | <spaces>]
        -b -l [-q <name>] [-s] [-f <filename> | <coservers>]
        -v [-p] [-t <time>] [-q <name>] [-L <level>] [-f <filename> | <spaces>]

    -b backup
    -c backup current logical log
    -C continuous logical log backup
    -f pathname of file containing list of storage spaces
    -F fake backup
    -l backup full logical logs (no spaces)
    -S skip backing up static dbspaces which are backed up already
    -L backup level: 0, 1, or 2, defaults to 0
    -O  override internal error checks - enforce policy strictly
    -p backup spaces only (no logs)
    -q name to identify the backup session, default <DBSERVERNAME><pid>
    -w whole system backup
    -s salvage logs
    -v verify consistency of most recent backups
    <spaces>, <coservers> list of storage spaces, coserver logs to backup
```

End of Extended Parallel Server

The backup or verification command was entered incorrectly. Revise the command and try again. The **-b** or **-v** parameter must come first in the command.

-43001:

ON–Bar restore usage.

Dynamic Server

```
For Dynamic Server
-r [-e] [-O] [-f filename | spaces]
-r [-e] [-t time | -n log] [-O]
-r -p [-e] [-t time] [-O] [-f filename | spaces]
-r -l [-t time | -n log]
-r -w [-e] [[-p] [ -t time] | -n log] [-O]
-RESTART
-r -rename -f filename [-w] [-p] [-t time] [-n log] [-f filename | spaces]
-r {-rename -p old path -o old offset -n new path -o new offset...}
   [-w] [-p] [-t time] [-n log] [-f filename | spaces]
```

End of Dynamic Server

Extended Parallel Server

```
For IBM Informix Extended Parallel Server:
        -r [-e] [-I] [-q <name>] [-O] [-f <filename> | <spaces>]
        -r [-e] [-i] [-I] [-q <name>] [-t time] [-O]
        -r -p [-e] [-q <name>] [-t time] [-O] [-f <filename> | <spaces>]
```

```
        -r -l [-I] [-q <name>] [-t <time>] [-f <filename> | <logstreams>]

    -e external restore
    -f pathname of file containing list of storage spaces
    -i index rebuild only (do not restore spaces or logs)
    -I do not rebuild indices following roll-forward
    -l logical log only restore (no spaces)
    -n last logical log to restore
    -O  override internal error checks - enforce policy strictly
    -p physical only restore (no logs)
    -q name to identify the restore session
    -r restore
    -t point in time to stop restore
    -w whole system to restore
    -RESTART restart an interrupted restore
    <spaces> list of storage spaces
```

End of Extended Parallel Server

The restore command was entered incorrectly. Revise the command and try again. You must specify the **-r** parameter first.

-43002:

ON–Bar session usage.

Extended Parallel Server

```
For Extended Parallel Server
    {on | ON | off | OFF | -d} [-q] session_name

    on/ON    resume a suspended backup/restore session
    off/OFF  suspend a backup/restore session
    -d       destroy backup/restore session
    -q       name to identify the session
```

The session command was entered incorrectly. Revise the command and try again. The session name is required.

End of Extended Parallel Server

-43003:

onbar_w usage.

Extended Parallel Server

```
For Extended Parallel Server
[-b] [-d] [-l] [-r] [-p]

-b accept backup requests
-d accept db/blobspaces
-l accept logical logs
-r accept restore requests
-p accept storage manager placement requests
If no commands are entered, accept all requests by default.
```

The **onbar_w** command was entered incorrectly. Revise the command and try again.

End of Extended Parallel Server

-43006:

onsmsync usage.

```
onsmsync [-g gen | -t time | -i interval] [-O]
         [-f filename | spaces]

onsmsync -b

    -b just regenerate the emergency boot file
    -f pathname of file containing list of storage spaces
    -g number of generations/versions of level-0 backup to
       retain
    -i time interval (age) before which objects should be
       expired
    -t datetime before which objects should be expired
    -O override internal error checks - enforce policy strictly
    spaces list of storage spaces to check for expiration
```

The **onsmsync** command was entered incorrectly. Revise the command and try again.

-43007:

ON-Bar messaging usage.

Dynamic Server

```
For IBM Informix Dynamic Server:

   -m [lines] [-r[seconds]]

-m Displays last lines of onbar's activity log file (default: 20 lines)
-r Repeat display every seconds seconds (default: 5 seconds)
```

End of Dynamic Server

-43357:

Logical log display

Dynamic Server

```
For IBM Informix Dynamic Server:
{-P} {-n log unique identifier | starting log unique identifier - ending log
unique identifier} [-l] [-q] [-b] [-u username] [-t TBLspace number]
[-x transaction number]

    -P Print backed up logical-log(s) information
    -n Display the specified log identifier(s)
    -l Display the maximum information about each log record
    -q Do not display program header
    -b Display information about logged BLOB pages
    -u Display the specified user(s)
    -t Display the specified TBLspace(s)
    -x Display the specified transaction(s)
```

End of Dynamic Server

ON-Bar Return Codes

Table 12-2 shows the ON–Bar return codes for all Informix database servers. These return codes are accompanied by messages in the ON–Bar activity log. For details about an ON–Bar or storage-manager error, review the activity log before you call Technical Support.

Table 12-2. Common ON-Bar Return Codes

Decimal Value	ON-Bar Return Code Description
2 through 34	These return codes are produced by XBSA. For more information, consult your storage-manager documentation and log files.
100	ON–Bar cannot find something in **sysutils**, the emergency boot file, or storage-manager catalogs that it needs for processing. Check the ON–Bar activity log for messages that say what could not be found and try to resolve that problem. If the problem recurs, contact Technical Support.
104	Adstar Distributed Storage Manager (ADSM) is in generate-password mode. ON–Bar does not support ADSM running in generate-password mode. For information on changing the ADSM security configuration, refer to your ADSM manual.
115	A critical dbspace is not in the set of dbspaces being cold-restored.
116	The **onsmsync** utility is already running.
117	The information contained in the sysutils database and the Emergency Boot File are inconsistent.
118	An error trying to commit a backup object to the storage manager.
119	The logical log is full on one or more coservers. Perform a logical-log backup.
120	The transport buffer size has changed since this object was last backed up. This object cannot be restored. Set the transport-buffer size to its original value and retry the restore.
121	Error occurred on dbslice name expansion (XPS). ON–Bar was unable to determine the list of dbspaces in a dbslice.
122	Deadlock detected. The ON–Bar command is contending with another process. Retry the ON–Bar command.
123	The root dbspace was not in the cold restore. You cannot perform a cold restore without restoring the root dbspace. To resolve the problem, try one of the following procedures: • Bring the database server to quiescent or online mode and restore just the storage spaces that need to be restored. • If the database server is offline, issue the **onbar -r** command to restore all the storage spaces. • Make sure that the root dbspace and other critical dbspaces are listed on the command line or in the **-f** filename.
124	The buffer had an incomplete page during the backup. For assistance, contact Technical Support.
126	Error processing the emergency boot file. Check the ON–Bar activity log for descriptions of the problem and the emergency boot file for corruption such as non-ASCII characters or lines with varying numbers of columns. If the source of the problem is not obvious, contact Technical Support.

Table 12-2. Common ON-Bar Return Codes (continued)

Decimal Value	ON-Bar Return Code Description
127	Could not write to the emergency boot file. Often, an operating-system error message accompanies this problem. Check the permissions on the following files and directories: • **$INFORMIXDIR/etc** on UNIX or **%INFORMIXDIR%\etc** on Windows • The emergency boot file • The directory that BAR_BOOT_DIR points to (XPS only).
128	Data is missing in the object description. For assistance, contact Technical Support.
129	ON–Bar received a different object for restore than it had expected. (The backup object did not match.) The requested backup object might have been deleted or expired from the storage manager. Run **onsmsync** to synchronize the **sysutils** database, emergency boot file, and storage-manager catalogs. For assistance, contact Technical Support.
130	Database server is not responding. The database server probably failed during the backup or restore. Run the **onstat -** command to check the database server status and then: • If the operation was a cold restore, restart it. • If the operation was a backup or warm restore, restart the database server and retry the backup or warm restore.
131	A failure occurred in the interface between ON–Bar and the database server. For assistance, contact Technical Support.
132	Function is not in the XBSA shared library. Verify that you are using the correct XBSA for the storage manager. For information, consult your storage-manager manual.
133	Failed to load the XBSA library functions. Verify that you are using the correct XBSA for the storage manager. Ensure that the BAR_BSALIB_PATH value in the ONCONFIG file points to the correct location of the XBSA shared library. For information, consult your storage-manager manual.
134	User wants to restore a logical-log file that is too early. You probably tried a point-in-log restore (**onbar -r -l -n**) after performing a separate physical restore. The specified logical log is too old to match the backups used in the physical restore. Perform either of the following steps: • Rerun the physical restore from an older set of physical backups. • Specify a later logical log in the **-n** option when you rerun the point-in-log restore. To find the earliest logical log that you can use, look at the emergency boot file. For assistance, contact Technical Support.
136	ON–Bar cannot warm restore the critical dbspaces. Perform either of the following steps: • Reissue the warm-restore command without listing any critical dbspaces. • Shut down the database server and perform a cold restore.
137	The MAX_DBSPACE_COUNT was exceeded. For assistance, contact Technical Support.

Table 12-2. Common ON-Bar Return Codes (continued)

Decimal Value	ON-Bar Return Code Description
138	An XBSA error occurred. Verify that you are using the correct XBSA for the storage manager. Also check the **bar_act.log** for XBSA error messages. For information, consult your storage-manager manual.
139	Either the XBSA version is missing from the **sm_versions** file or the incorrect XBSA version is in the **sm_versions** file. Insert the correct XBSA version into the **sm_versions** file. For more information, consult your storage-manager manual.
140	A fake backup failed. Retry the fake backup using the **onbar -b -F** command. Only IDS supports fake backups. If the fake backup fails again, contact Technical Support.
141	ON–Bar received an operating-system signal. Most likely, the user entered the Ctrl-C command to stop an ON–Bar process. Fix the cause of the interruption and then retry the ON–Bar command.
142	ON–Bar was unable to open a file. Verify that the named file exists and that the permissions are correct. Check the ON–Bar activity log for an operating-system error message.
143	ON–Bar was unable to create a child process. If BAR_MAX_BACKUP is not 0, ON–Bar could not create child processes to perform the parallel backup or restore. The operating system probably ran out of resources. Either not enough memory is available to start a new process or no empty slot exists in the process table. Check the operating-system logs, the ON–Bar activity log, or the console.
144	The log backup was aborted because one or more blobspaces were down. Attempt to restore the blobspace. If the restore fails, retry the log backup using the **onbar -l -O** command. Executing this command might make the blobspace unrestorable.
145	ON–Bar was unable to acquire more memory space. Wait for system resources to free up and retry the ON–Bar command.
146	ON–Bar was unable to connect to the database server. The network or the database server might be down. For assistance, contact Technical Support.
147	ON–Bar was unable to discover any storage spaces or logical logs to back up or restore. For example, if you specify a point-in-time restore but use a *datetime* value from before the first standard backup, ON–Bar cannot build a list of storage spaces to restore. This return code also displays if you specify a whole-system restore without having performed a whole-system backup. Verify that the database server and the storage spaces are in the correct state for the backup or restore request. Contact Technical Support.
148	An internal SQL error occurred. Provide Technical Support with the information from the ON–Bar activity log.

Table 12-2. Common ON-Bar Return Codes (continued)

Decimal Value	ON-Bar Return Code Description
149	Either you entered the wrong ON–Bar syntax on the command line or entered an invalid or incorrect *datetime* value for your GLS environment. Check the command that you tried against the usage message in the ON–Bar activity log. If that does not help, then retry the command with quotes around the *datetime* value. If your database locale is not English, use the **GL_DATE** or **GL_DATETIME** environment variables to set the date and time format.
150	Error collecting data from the ONCONFIG file. Check the permissions, format, and values in the ONCONFIG file. Check that the **ONCONFIG** environment variable is set correctly.
151	The database server is in an incorrect state for this backup or restore request, or an error occurred while determining the database server state. Either you attempted an operation that is not compatible with the database server mode or ON–Bar is unable to determine the database server state. For example, you cannot do a physical backup with the database server in recovery mode. Check the error message in the ON–Bar activity log. If an ASF error occurred, the following message displays in the ON–Bar activity log: `Fatal error initializing ASF; asfcode = code` To determine the cause of the ASF error, refer to the ASF error code in this message and repeat the backup or restore command. If an ASF error did not occur, change the database server state and repeat the backup or restore command.
152	ON–Bar cannot back up the logical logs. The logical logs are not backed up for either of the following reasons: • If another log backup is currently running. • If you perform a logical-log backup with the LTAPEDEV parameter set to **/dev/null** (UNIX) or **nul** (Windows). You receive this return code when no log backups can be done. To enable log backups, change the LTAPEDEV parameter to a valid value.
153	ON–Bar cannot set the process group id. If BAR_MAX_BACKUP is set to any value other than 1 and ON–Bar encounters an error setting the process group id, this value is returned. This message is a warning of a possible operating-system problem.
154	The ON–Bar user does not have the correct permissions. You must be user **root** or **informix** or a member of the **bargroup** group on UNIX or a member of the **Informix-Admin** group on Windows to execute ON–Bar commands.
155	The **INFORMIXSERVER** environment variable is not set. Set the **INFORMIXSERVER** environment variable to the correct database server name.
156	Backup or restore was not performed because the LTAPEDEV parameter value is not valid. If LTAPEDEV is not set or /dev/null on UNIX, or if it is \dev\nul or NUL on Windows, the logical logs are not backed up, and ON-Bar returns warning 152.
157	Error attempting to set the **INFORMIXSHMBASE** environment variable to -1. ON–Bar could not set **INFORMIXSHMBASE** to -1. For assistance, contact either the system administrator or Technical Support.

Table 12-2. Common ON-Bar Return Codes (continued)

Decimal Value	ON-Bar Return Code Description
158	An internal ON–Bar error occurred. Contact Technical Support.
159	An unexpected error occurred. Contact Technical Support.
160	External restore failed. To determine what went wrong with the external restore, look at the **bar_act.log** and the **online.log** files. Ensure that you already performed the manual part of the external restore before you retry the **onbar-r -e** command to complete the external restore. If that does not work, try the external restore from a different external backup.
161	Restarted restore failed. Verify that RESTARTABLE_RESTORE is set to ON and try the original restore again. For more information, check the ON–Bar activity log and database server message logs (IDS).
162	The ON-Bar log file cannot be a symbolic link. Remove the symbolic link or change the ONCONFIG file so that the ON-Bar parameters BAR_DEBUG_LOG or BAR_ACT_LOG point to non-symbolic linked files.
163	The ON-Bar log file must be owned by user **informix**. Change the ownership of the log file to be owned by user **informix** or change the BAR_ACT_LOG or BAR_DEBUG_LOG values in the ONCONFIG file to point to different log files.
164	Unable to open file. The file or its directory permissions prevent it from being created or opened. Verify the permissions on the file and its directory.
177	An online dbspace was restored. This return code notifies the user that the **-O** option overrode the internal checks in ON–Bar. You do not need to take any action.
178	The logical log was backed up while one or more blobspaces were down. This return code notifies the user that the **-O** option overrode the internal checks in ON–Bar. Examine the data in the blobspace to determine which simple large objects you need to recreate. These blobspaces might not be restorable. For assistance, contact Technical Support.
179	ON–Bar created the chunk needed to restore the dbspace. This return code notifies the user that the **-O** option overrode the internal checks in ON–Bar. You do not need to take any action.
180	ON–Bar could not create the chunk needed to restore the dbspace. Create the chunk file manually. Retry the restore without the **-O** option.
181	ON–Bar expired an object that was needed for a backup or restore. The **onsmsync** utility expired an object that might be needed for a restore. You probably specified **onsmsync** with the **-O** option. If you used the **-O** option by mistake, contact Technical Support to recover the object from the storage manager.
183	ON-Bar could not obtain the logical-log unique ID from the storage manager. The backup of the specified logical log is missing. Query your storage manager to determine if the backup of the specified logical-log file exists and if it is restorable.

Table 12-2. Common ON-Bar Return Codes (continued)

Decimal Value	ON-Bar Return Code Description
247	Merging of emergency boot files timed out because it took too long. (XPS) On UNIX, look in **/tmp/bar_act.log** and the file that the BAR_ACT_LOG parameter points to for clues. (The **onbar-merger** writes to **/tmp/bar_act.log** until it has enough information to read the ONCONFIG file.) Resolve the problems that the **bar_act.log** describes and retry the cold restore. If the cold restore still fails, contact Technical Support.
252	Received the wrong message from **onbar_m** that merges the emergency boot files. (XPS) For assistance, contact Technical Support.

Part 3. ontape Backup and Restore System (IDS)

© Copyright IBM Corp. 1996, 2008

Chapter 13. Configuring ontape

In This Chapter 13-1
ontape Configuration Parameters 13-1
ontape Data Transformation Filter Parameters 13-2
ontape Tape and Tape Device Parameters 13-2
Setting the Tape-Device Parameters 13-3
Specify Separate Devices for Storage-Space and Logical-Log Backups 13-3
Specifying Tape Devices as Symbolic Links 13-3
Specifying a File System Directory 13-4
Specifying a Remote Device 13-4
Specifying /dev/null for a Tape Device 13-4
Setting TAPEDEV to stdio 13-5
Rewinding Tape Devices Before Opening and on Closing 13-5
Specifying the Tape-Block-Size Parameters 13-5
Specifying the Tape-Size Parameters 13-5
Tape Size for Remote Devices 13-5
Checking and Changing ontape Configuration Parameters 13-5
Who Can Change ontape Parameters? 13-6
When Can You Change ontape Parameters? 13-6
Changing TAPEDEV to /dev/null 13-6
Changing LTAPEDEV to /dev/null 13-6
Verifying That the Tape Device Can Read the Specified Block Size 13-6
Changing ontape Parameters 13-6
Changing Backup-Tape Parameters 13-6
Changing Logical-Log Backup Tape Parameters 13-7

In This Chapter

This chapter explains how to set the configuration parameters that the **ontape** utility uses for backups of storage spaces and logical logs. For a description of how **ontape** differs from ON–Bar, see "Comparing ON-Bar and ontape" on page 1-6.

This chapter describes the following tasks:

- Setting the configuration parameters for **ontape**
- Checking configuration parameters for **ontape**
- Changing configuration parameters for **ontape**

Chapter 14, "Backing Up with ontape," on page 14-1 describes how to use the **ontape** utility to back up storage spaces and logical-log files.

You can also run ontape commands from SQL statements. For more information, see *Administrator's Guide* and *Guide to SQL: Syntax*.

ontape Configuration Parameters

The **ontape** utility uses eight configuration parameters in the ONCONFIG file

The ONCONFIG file is located in the $INFORMIXDIR**/etc** directory. You specify that file in the **ONCONFIG** environment variable. For a description of the **ONCONFIG** environment variable and instructions on how to set it, see the *IBM Informix Guide to SQL: Reference*.

© Copyright IBM Corp. 1996, 2008

Two of the configuration parameters are used to specify filter programs for transforming data during backup and restore; the other six are used to create storage-space and logical-log backups.

ontape Data Transformation Filter Parameters

The filter parameters specify the names of external programs that you can use to transform date prior to backup and following restore.

BACKUP_FILTER

Specifies the location and name of an external filter program used in data transformation. This filter transforms data prior to backing it up, such as compressing it. The transformed data is then backed up and stored as a single file. The filter path points to the $INFORMIXDIR/bin directory by default, or an absolute path of the program

RESTORE_FILTER

Specifies the location and name of an external filter program used in data transformation. This filter transforms data back to its original state prior to the backup, such as decompressing it, before returning the data to the server. The filter path points to the $INFORMIXDIR/bin directory by default, or an absolute path of the program

Prerequisite: The data must have previously been transformed with the BACKUP_FILTER parameter

For more information and examples about using these filters, see "Transforming with Filters during Backup and Restore" on page 3-12. See "BACKUP_FILTER" on page 10-6 and "RESTORE_FILTER" on page 10-21 for syntax and usage information, which is the same for ON-Bar and **ontape**.

ontape Tape and Tape Device Parameters

The first set of ONCONFIG parameters specifies the characteristics of the tape device and tapes for storage-space backups; the second set specifies the characteristics of the tape device and tapes for logical-log backups.

The following list shows backup tape devices and their associated tape parameters.

TAPEDEV The absolute path name of the tape device or directory file system that is used for storage-space backups. Specify the destination where **ontape** will write storage space data during an archive and the source from which **ontape** will read data during a restore.

To configure **ontape** to use **stdio**, set TAPEDEV to STDIO.

TAPEBLK The block size of the tapes used for storage-space backups, in kilobytes.

TAPESIZE The size of the tapes used for storage-space backups, in kilobytes. The value can be 0 to 2,097,151.

The following list shows the logical-log tape devices and their associated tape parameters.

LTAPEDEV The logical-log tape device or a directory of a file system.

LTAPEBLK The block size of tapes used for logical-log backups, in kilobytes.

LTAPESIZE The size of tapes used for logical-log backups, in kilobytes. The value can be 0 to 2,097,151.

The following sections contain information about how to set the tape-device, tape-block-size, and tape-size parameters for both storage-space and logical-log backups.

Setting the Tape-Device Parameters

Specify values for TAPEDEV and LTAPEDEV in the following ways:

- Use separate tape devices, when possible.
- Use symbolic links.
- Specify a directory of a file system.
- For tape devices, specify **/dev/null**.
- Rewind tape devices.

The following sections explain each of these points.

Specify Separate Devices for Storage-Space and Logical-Log Backups

When backing up to a tape device, specify different devices for the LTAPEDEV and TAPEDEV parameters in the ONCONFIG file. You can schedule these backups independently of each other. You can create a backup on one device at the same time you continuously back up the logical-log files on the other.

If you specify the same device for the LTAPEDEV and TAPEDEV, the logical log can fill, which causes the database server to stop processing during a backup. When this happens, you have two options.

- Abort the backup to free the tape device and back up the logical-log files.
- Leave normal processing suspended until the backup completes.

Precautions to Take When You Use One Tape Device: When only one tape device exists and you want to create backups while the database server is online, take the following precautions:

- Configure the database server with a large amount of logical-log space through a combination of many or large logical-log files. (See your *IBM Informix Administrator's Guide*.)
- Store all explicitly created temporary tables in a dedicated dbspace and then drop the dbspace before backing up.
- Create the backup when low database activity occurs.
- Free as many logical-log files as possible before you begin the backup.

The logical log can fill up before the backup completes. The backup synchronizes with a checkpoint. A backup might wait for a checkpoint to synchronize activity, but the checkpoint cannot occur until all virtual processors exit critical sections. When database server processing suspends because of a full logical-log file, the virtual processors cannot exit their critical sections and a deadlock results.

Specifying Tape Devices as Symbolic Links

You can specify the values of LTAPEDEV and TAPEDEV as symbolic links. Using symbolic links enables you to switch to other tape or tape-compatible devices without changing the path name in the ONCONFIG file. For example, you can specify the following symbolic link for tape device **/dev/rst0**:

```
ln -s /dev/rst0 /dbfiles/logtape
```

When you set the LTAPEDEV configuration parameter, as the following example shows, you can switch to a different device without changing the LTAPEDEV parameter:

```
LTAPEDEV /dbfiles/logtape
```

You only need to change the symbolic link, as the following example shows:

```
ln -s /usr/backups /dbfiles/logtape
```

A user with one tape device could redirect a logical-log backup to a disk file while using the tape device for a backup.

Specifying a File System Directory

You can perform a storage-space (level 0, 1, or 2) archive, or a logical-log backup to a directory in the file system by using the **ontape** utility. For each storage-space archive and logical-log backup, ontape creates a file in the specified directory.

To specify a file system directory, set the TAPEDEV or LTAPDEV configuration parameters to the absolute path name for the directory.

When ontape repeats an archive operation, it renames the existing files so that old files are not rewritten. A timestamp is added to the file name, which provides a way for related storage space or logical log files to be organized together.

To learn about the file naming schema, see "Renaming of existing files" on page 14-7.

Specifying a Remote Device

You can perform a storage-space or logical-log backup across your network to a remote device attached to another host computer. You should not do a continuous backup to a remote device. To specify a tape device on another host computer, use the following syntax:

```
host_machine_name:tape_device_pathname
```

The following example specifies a tape device on the host computer **kyoto**:

```
kyoto:/dev/rmt01
```

For information on the tape size for remote devices, see "Tape Size for Remote Devices" on page 13-5.

Specifying /dev/null for a Tape Device

It is recommended that you do not use **/dev/null** as the device when backing up. However, when you specify **/dev/null** as a backup tape device, you can avoid the overhead of a level-0 backup that is required after some operations, such as changing the logging status of a database. Obviously, you cannot restore storage spaces from a backup to **/dev/null**.

You can specify **/dev/null** as a tape device for logical-log backups when you decide that you do not need to recover transactions from the logical log. When you specify the tape device as **/dev/null**, block size and tape size are ignored. If you set LTAPEDEV either to or from **/dev/null, you must use ON-Monitor or restart the database server for the new setting to take effect.**

Warning: When you set the ONCONFIG parameter LTAPEDEV to **/dev/null**, the database server marks the logical-log files as backed up as soon as they become full, effectively discarding logical-log information.

Setting TAPEDEV to stdio

To configure ontape to read from standard input or write to standard output, set the TAPEDEV configuration parameter to STDIO.

Rewinding Tape Devices Before Opening and on Closing

With **ontape**, you must use *rewindable* tape devices. Before reading from or writing to a tape, the database server performs a series of checks that require the rewind.

Specifying the Tape-Block-Size Parameters

Specify the block-size parameters TAPEBLK and LTAPEBLK as the largest block size, in kilobytes, that your tape device permits.

When you set the tape parameter to **/dev/null**, the corresponding block size is ignored.

The **ontape** utility does not check the tape device when you specify the block size. Verify that the tape device can read the block size that you specified. If not, you cannot restore the tape.

Specifying the Tape-Size Parameters

The number of blocks specify tape sizes. TAPESIZE and LTAPESIZE specify the maximum amount of data that you can write to a tape.

To write or read the tape to the end of the device, set TAPESIZE and LTAPESIZE to `0`. You cannot use this option for remote devices.

When you specify the tape device as **/dev/null**, the corresponding tape size is ignored.

The range of values for these parameters is 0 to 2,097,151.

Tape Size for Remote Devices

When you perform a continuous logical-log backup to a remote device, the amount of data written to the tape is the smaller of LTAPESIZE and the following formula:

```
(sum of space occupied by all logical-log files on disk) -
(largest logical-log file)
```

The I/O to the remote device completes and the database server frees the logical-log files before a log-full condition occurs.

Note: You cannot set tape size to 0 for remote devices.

Checking and Changing ontape Configuration Parameters

To examine your ONCONFIG file (the file specified in **$INFORMIXDIR/etc/$ONCONFIG**), use one of the following:

- Execute **onstat -c** while the database server is running.
- Use IBM Informix Server Administrator (**Configuration > ONCONFIG**).

This section provides information on how to change configuration parameters for **ontape**.

Who Can Change ontape Parameters?

When you log in as either user **informix** or **root**, you can use a text editor, ON-Monitor, or ISA to change the value of configuration parameters for **ontape**.

When Can You Change ontape Parameters?

You can change the values of parameters for **ontape** while the database server is online. The change takes effect immediately.

Note: If you want to set either the TAPEDEV parameter or the LTAPEDEV parameter to **/dev/null**, you must use the ON-Monitor utility to make this change while the database server is online. If you use any other method to alter the value of the configuration parameters to or from **/dev/null**, you must restart the database server to make the change effective.

Changing TAPEDEV to /dev/null

The **ontape** utility reads the value of the TAPEDEV parameter at the start of processing. When you set TAPEDEV to **/dev/null** and request a backup, the database server bypasses the backup but still updates the dbspaces with the new backup time stamps. When you set TAPEDEV to **/dev/null,** you must do it before you start **ontape** to request the backup. No problems exist when you change TAPEDEV to **/dev/null** with ON-Monitor while the database server is online and **ontape** is not running.

Changing LTAPEDEV to /dev/null

Take the database server offline before you change the value of LTAPEDEV to **/dev/null**. When you make the change while the database server is either quiescent or online, you can create a situation where you back up one or more logical-log files but do not free them. This situation can interrupt processing because the database server stops when it finds that the next logical-log file (in sequence) is not free.

When you set LTAPEDEV to **/dev/null**, the database server frees the logical logs without requiring that you back up those logs. The logical logs do not get marked as free, but the database server can reuse them.

Verifying That the Tape Device Can Read the Specified Block Size

The **ontape** utility does not check the tape device when you specify the block size. Verify that the tape device specified in TAPEDEV and LTAPEDEV can read the block size you specify for their block-size parameters. If not, you cannot restore the tape.

Changing ontape Parameters

Before you change the parameters for **ontape**, perform a level-0 backup, as explained in "Performing a Backup" on page 14-7.

Changing Backup-Tape Parameters

To change the value of TAPEDEV, TAPEBLK, and TAPESIZE from the command line, use a text editor to edit your ONCONFIG file. Save the file. Most change takes effect immediately. However, if you set TAPEDEV to or from **dev/null,** you must restart the database server for the setting to take effect. You also can change these parameters in ISA.

Changing Logical-Log Backup Tape Parameters

To change the value of LTAPEDEV, LTAPEBLK, and LTAPESIZE from the command line, use a text editor to edit your ONCONFIG file. Save the file. Most change takes effect immediately. However, if you set LTAPEDEV to or from **dev/null,** you must restart the database server for the setting to take effect.

You also can change these parameters in ISA.

Chapter 14. Backing Up with ontape

In This Chapter 14-1
Summary of ontape Tasks 14-1
Starting ontape 14-2
Using ontape Exit Codes 14-2
Changing Database Logging Status 14-2
Creating a Backup 14-3
Backup Levels 14-3
Level-0 Backups 14-3
Level-1 Backups 14-3
Level-2 Backups 14-3
Back Up After Changing the Physical Schema 14-3
Prepare for a Backup 14-4
Avoid Using Temp Tables During Heavy Activity 14-4
Making Sure Enough Logical-Log Space Exists 14-4
Keep a Copy of Your ONCONFIG File. 14-4
Verify Consistency Before a Level-0 Backup 14-4
Online and Quiescent Backups 14-4
Back Up to Tape 14-5
Back Up to Standard Output 14-6
Back Up to a Directory 14-6
Performing a Backup 14-7
Backup Examples. 14-8
Backing Up Raw Tables. 14-9
When the Logical-Log Files Fill During a Backup 14-9
When You Can Use Two Tape Devices. 14-9
When Only One Tape Device Is Available 14-10
When a Backup Terminates Prematurely. 14-10
Monitoring Backup History Using oncheck 14-10
Backing Up Logical-Log Files with ontape 14-10
Before You Back Up the Logical-Log Files 14-11
Using Blobspace TEXT and BYTE Data Types and Logical-Log Files 14-11
Using /dev/null When You Do Not Need to Recover 14-11
When to Back Up Logical-Log Files 14-12
Starting an Automatic Logical-Log Backup 14-12
Starting a Continuous Logical-Log File Backup 14-12
Ending a Continuous Logical-Log Backup 14-13
Devices that Logical-Log Backups Must Use 14-13

In This Chapter

This chapter describes how to use **ontape** to back up storage spaces and logical-log files, and how to change the database logging status. The **ontape** utility can back up and restore the largest chunk files that your database server supports. The **ontape** utility cannot back up temporary dbspaces and temporary sbspaces.

Summary of ontape Tasks

The **ontape** utility lets you complete a wide variety of tasks:

- "Changing Database Logging Status" on page 14-2
- "Creating a Backup" on page 14-3
- "Starting a Continuous Logical-Log File Backup" on page 14-12
- "Performing a Restore" on page 15-4

© Copyright IBM Corp. 1996, 2008

- "Using External Restore Commands" on page 16-4

Starting ontape

When you need more than one tape during a backup, **ontape** prompts for each additional tape.

If the database server is in maintenance mode, for example, during a conversion, then ontape can only be started by one of the following users:

- **root**
- **informix**
- The user who started the database server (if not the user **root** or **informix**)

Warning: Do not start **ontape** in background mode (that is, using the UNIX **&** operator on the command line). You could also need to provide input from the terminal or window. When you execute **ontape** in background mode, you can miss prompts and delay an operation.

The **ontape** utility does not include default values for user interaction, nor does it support retries. When **ontape** expects a yes/no response, it assumes that any response not recognized as a "yes" is "no".

Using ontape Exit Codes

The **ontape** utility has the following two exit codes:

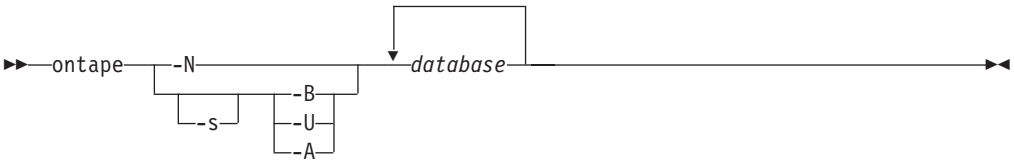
0 indicates a normal exit from **ontape**.

1 indicates an exception condition.

Changing Database Logging Status

To change database logging status:

Changing Database Logging Status



When you add logging to a database, you must create a level-0 backup before the change takes effect.

-A directs **ontape** to change the status of the specified database to ANSI-compliant logging.

-B directs **ontape** to change the status of the specified database to buffered logging.

database The name of the database. The database name cannot include a database server name.

-N directs **ontape** to end logging for the specified database.

-s initiates a backup.

-U directs **ontape** to change the status of the specified database to unbuffered logging.

For considerations about changing the logging status of a database, see your *IBM Informix Administrator's Guide*.

Creating a Backup

This section explains how to plan for and create backups of your database server data.

Backup Levels

The **ontape** utility supports level-0, level-1, and level-2 backups. For information on scheduling backups, see "Planning a Recovery Strategy" on page 2-1.

Tip: Establish a backup schedule that keeps level-1 and level-2 backups small. Schedule frequent level-0 backups to avoid restoring large level-1 and level-2 backups or many logical-log backups.

Level-0 Backups

When a fire or flood, for example, completely destroys a computer, you need a level-0 backup to completely restore database server data on the replacement computer. For online backups, the data on the backup tape reflects the contents of the storage spaces at the time the level-0 backup began. (The time the backup started could reflect the last checkpoint before the backup started.)

A level-0 backup can consume lots of time because **ontape** must write all the pages to tape.

Level-1 Backups

A level-1 backup usually takes less time than a level-0 backup because you copy only part of the database server data to the backup tape.

Level-2 Backups

A level-2 backup after a level-1 backup usually takes less time than another level-1 backup because only the changes made after the last level-1 backup (instead of the last level-0) get copied to the backup tape.

Back Up After Changing the Physical Schema

You must perform a level-0 backup to ensure that you can restore the data after you make the following administrative changes. Consider waiting to make these changes until your next regularly scheduled level-0 backup.

- Changing TAPEDEV or LTAPEDEV from **/dev/null**
- Adding logging to a database
- Adding a dbspace, blobspace, or sbspace before you can restore it with anything less than a full-system restore
- Starting mirroring for a dbspace that contains logical-log files
- Dropping a logical-log file
- Moving one or more logical-log files
- Changing the size or location of the physical log and after you set up shared memory
- Dropping a chunk before you can reuse the dbspace that contains that chunk
- Renaming a chunk during a cold restore

Tip: Although you no longer need to backup immediately after adding a logical-log file, your next backup should be level-0 because the data structures have changed.

Prepare for a Backup

When you create a backup, take the following precautions:

- Avoid using temp tables during heavy activity.
- Make sure you make sufficient logical-log space to create a backup.
- Keep a copy of your ONCONFIG file.
- Verify data consistency.
- Run the database server in the appropriate mode.
- Plan for operator availability.
- Synchronize with other administrative tasks.
- Do not use background mode.
- If necessary, label tapes appropriately.
- If necessary, prepare for writing to standard output.

The following sections address each of these topics.

Avoid Using Temp Tables During Heavy Activity

When you create a temp table during a backup while using the **ontape** utility, that table is placed in DBSPACETEMP. When heavy activity occurs during the backup process, the temp table can keep growing and can eventually fill up DBSPACETEMP. When this situation occurs, the backup aborts and your monitor displays a NO FREE DISK error message.

Making Sure Enough Logical-Log Space Exists

When the total available space in the logical log amounts to less than half a single logical-log file, the database server does not create a backup. You must back up the logical-log files and attempt the backup again.

You cannot add mirroring during a backup.

Important: When you use only one available tape device, make sure you back up all your logical-log files before you start your backup to reduce the likelihood of filling the logical log during the backup.

Keep a Copy of Your ONCONFIG File

Keep a copy of the current ONCONFIG file when you create a level-0 backup. You need this information to restore database server data from the backup tape.

Verify Consistency Before a Level-0 Backup

To ensure the integrity of your backups, periodically verify that all database server data and overhead information is consistent before you create a full-system level-0 backup. You need not check this information before every level-0 backup, but we recommend that you keep the necessary tapes from the most recent backup created immediately after the database server was verified as consistent. For information on consistency checking, see your *IBM Informix Administrator's Guide*.

Online and Quiescent Backups

You can create a backup while the database server is online or in quiescent mode. The terminal you use to initiate the backup command is dedicated to the backup

(displaying messages) until the backup completes. Once you start a backup, the database server must remain in the same mode until the backup finishes; changing the mode terminates the backup activity.

Online Backup: You can use an online backup when you want your database server accessible while you create the backup.

Some minor inconveniences can occur during online backups. An online backup can slow checkpoint activity, and that can contribute to a loss in performance. However, this decline in performance is far less costly than the time that you lose when users were denied access to the database server during a backup.

During an online backup, allocation of some disk pages in storage spaces can temporarily freeze. Disk-page allocation is blocked for one chunk at a time until you back up the used pages in the chunk.

Quiescent Backup: You create a quiescent backup while the database server is quiescent. Use quiescent backups when you want to eliminate partial transactions in a backup.

Do not use quiescent backups when users need continuous access to the databases.

Back Up to Tape

When you back up to tape, you must ensure that an operator is available and that you have sufficient media.

A backup can require multiple tapes. After a tape fills, **ontape** rewinds the tape, displays the tape number for labelling, and prompts the operator to mount the next tape when you need another one. Follow the prompts for labelling and mounting new tapes. A message informs you when the backup is complete.

Ensure That the Operator Is Available: Keep an operator available during a backup to mount tapes as prompted.

A backup could take several reels of tape. When an operator is not available to mount a new tape when one becomes full, the backup waits. During this wait, when the backup is an online backup, the physical log space could fill up, and that causes the database server to abort the backup. Thus, make sure an operator is available.

Label Tapes Created with ontape: When you label tapes created with **ontape**, the label must include the following information:

- Backup level
- Date and time
- Tape number that **ontape** provides

The following example shows what a label can look like:

```
Level 1: Wed Nov 27, 2001 20:45 Tape # 3 of 5
```

Each backup begins with its first tape reel numbered 1. You number each additional tape reel consecutively thereafter. You number a five-tape backup 1 through 5. (Of course, it is possible that you could not know that it is a five-tape backup until it is finished.)

Back Up to Standard Output

If you choose to back up to standard output, you do not need to provide tapes or other storage media.

A backup to standard output creates an archive in the memory buffer provided by the operating system. Backing up to standard output has the following advantages:

- There are no expensive write and read operations to disk or tape.
- You can use operating system utilities to compress or otherwise process the data.
- You can use the archive to create a duplicate of the server by immediately restoring the data onto another database server.

If you back up to standard output, you must also restore from standard input.

When **ontape** performs a backup to standard output, the data is written to an output file. The output file's directory must have enough disk space to hold the backed-up data. You can use operating system utilities to compress the data. In addition, the user executing the backup command must have write permission to the file to which the backup is diverted or permission to create the file.

When you back up to standard output, **ontape** does not prompt for user interaction. Error and information messages are written to stderr instead of being directed to standard output.

The TAPESIZE configuration parameter is not used because the capacity of standard output is assumed to be unlimited. The TAPEBLK configuration parameter, however, is valid because it defines the size of the transport buffer between the backend server and the **ontape** client. You can optimize throughput by setting TAPEBLK to an appropriate value.

You can simultaneously back up and restore a database server to clone it or set up High-Availability Data Replication. For more information, see "Simultaneous Backup and Restore Using Standard I/O" on page 15-14.

Back Up to a Directory

If you choose to back up to a directory, you do not need to provide tapes. Instead, you back up the data to a directory of a local file system or a directory that has been mounted on the local system.

The person who runs the backup must have write permission to the directory. The directory must have enough disk space to hold the backed-up data. You can use operating system utilities to compress the data after it is backed up.

Backing up to a directory has the following advantages:

- Multiple instances can simultaneously back up to the same directory file system.
- You can use operating system utilities to compress or otherwise process the data.
- You can easily configure your system to automatically back up a log file when the file is full.

Setting the file directory path: Use the TAPEDEV configuration parameter to specify the absolute path name on a directory of a file system to use for the storage-space archive file. This is the destination where **ontape** writes storage space data during an archive and the source from which **ontape** reads data during a restore. You specify the directory where the logical log backup files are written with the LTAPEDEV configuration parameter.

Tip: When you back up to a directory file system, specify the `-d` option to turn off **ontape** interactive prompts.

Renaming of existing files: When ontape repeats an archive operation, it renames the existing files so that old files are not rewritten. A timestamp is added to the file name, which provides a way for related storage space or logical log files to be organized together.

Renaming conventions:

- Storage-space archive files: The archive checkpoint time is added, and has the format *filename_YYYYMMDD_hhmmss_archive-level*.
- Logical log backup files: The backup time is added, and has the format *filename_YYYYMMDD_hhmmss*.

For example, the file My_instance_L0 is renamed to My_instance_20080913_091527_L0

When restoring from a file system directory, ontape requires that storage-space archive and logical-log backup files be named as specified by the TAPEDEV and LTAPEDEV parameters. If files have been renamed, including by ontape because of repeated archives and backups, files must be manually renamed to their original file names.

Overriding the default name of the archive files: You can override the default name of the archive files. When TAPEDEV or LTAPEDEV is a directory path, the default permanent file name consists of *hostname_servernum*_L*n* (for levels), and *hostname_servernum*_Log*nnnnnnnnnn* (for log files). You can override the prefix part of the permanent filename, *hostname_servernum*, by setting the environment variable IFX_ONTAPE_FILE_PREFIX.

For example, if you set IFX_ONTAPE_FILE_PREFIX to “My_Instance”, then during archive, the files are named My_Instance_L0, My_Instance_L1, My_Instance_L2, and, My_Instance_Log0000000001, My_Instance_Log0000000002, and so on. During restore, ontape searches for files in the TAPEDEV directory with file names like My_Instance_L0, and searches for files in the LTAPEDEV directory with file names like My_Instance_Log0000000001.

Performing a Backup

Before you begin a backup, perform the following steps:

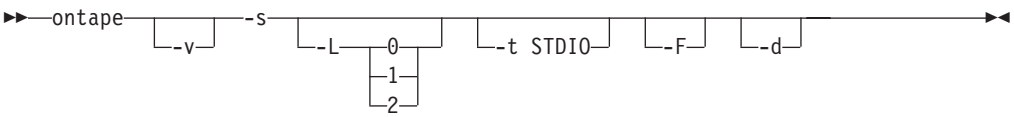
- If necessary, place a write-enabled tape on the tape-drive device that TAPEDEV specifies.

  If you set TAPEDEV to STDIO, ensure that there is enough memory for the backup data.
- Put the device online with the appropriate operating-system command.
- Place the database server in online or quiescent mode

Do not store more than one backup on the same tape; begin every backup with a different tape. (Often, a backup spans more than one tape.)

To create a backup, use the **-s** option of the **ontape** command.

Creating a Backup



Element	Purpose	Key Considerations
-F	Directs **ontape** to perform a fake backup.	A fake backup is only applicable during a backup to standard output. A fake backup is useful for cloning the data in a server. For example, to populate the secondary server in an High-Availability Data Replication pair. To avoid compromising the normal backup activities, do not keep a record of a fake backup Alternatively, you can use the SQL administration API equivalent: ARCHIVE FAKE. See *IBM Informix Guide to SQL: Syntax* for more information.
-L	Directs **ontape** to create a backup of the level specified.	If you are backing up to tape, use the **-L** option to specify the backup level as part of the command, you can avoid being prompted for it. If you are backing up to standard output, and do not specify a backup level, **ontape** performs and level-0 backup.
-s	Directs **ontape** to create a backup.	**ontape** prompts you to supply the backup level (0, 1, or 2) that you want to create if you do not supply a value using the **-L** option.
-t STDIO	Directs **ontape** to back up to standard output.	The **-t STDIO option overrides the value of the TAPEDEV configuration parameter for the current backup.**
-v	Directs **ontape** to write informational message to stderr during a backup to standard output.	Verbose mode is useful for monitoring the progress of a backup to standard output.
-d	Directs **ontape** to proceed without interactive prompts.	You can turn off the prompts if you are backing up to or restoring from a directory of a file system. This option does not apply to tape devices, which must pause the backup while you change tapes.

The **ontape** utility backs up the storage spaces in the following order: root dbspaces, blobspaces, sbspaces, and dbspaces.

Backup Examples

Execute the following command to start a backup to tape without specifying a level:

```
ontape -s
```

You can use the **-L** option to specify the level of the backup as part of the command, as the following example shows:

```
ontape -s -L 0
```

Use the -d option to avoid interactive prompts when you are backing up to or restoring from a directory:

```
ontape -s -L 0 -d
```

When you do not specify the backup level on the command line, **ontape** prompts you to enter it. Figure 14-1 illustrates a simple **ontape** backup session.

```
ontape -s
Please enter the level of archive to be performed (0, 1, or 2) 0

Please mount tape 1 on /dev/rst0 and press Return to continue ...
16:23:13 Checkpoint Completed: duration was 2 seconds
16:23:13 Level 0 Archive started on rootdbs
16:23:30 Archive on rootdbs Completed.
16:23:31 Checkpoint Completed: duration was 0 seconds

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 3

Program over.
```

Figure 14-1. Example of a Simple Backup Created with ontape

The following example shows how to create a level-0 archive of all storage spaces to standard output, which is diverted to a file named **level_0_archive** in the directory **/home**:

```
ontape -s -L 0 >/home/level_0_archive -t STDIO
```

The following example assumes TAPEDEV STDIO in **onconfig** and creates a level-1 archive to standard output , which is diverted to a pipe:

```
ontape -v -s -L 1|compress -c >/home/compressed/level_1_archive
```

The **compress** system utility reads from the pipe as input, compresses the data, and writes the data to the file **level_1_archive** in the **/home/compressed** directory. The **ontape** information messages are sent to stderr.

Backing Up Raw Tables

You can use **ontape** to back up a raw table, however, raw tables are not logged. Therefore, when you restore a raw table, any changes that occurred since the last backup cannot be restored. It is recommended that you use raw tables only for initial loading of data and subsequently alter raw tables to standard tables before performing transactions. For more information, see the *IBM Informix Dynamic Server Administrator's Guide*.

When the Logical-Log Files Fill During a Backup

When the logical log fills during a backup, the console displays a message and the backup suspends normal processing. How you handle the logical-log filling depends on whether you can use one or two tape devices.

When You Can Use Two Tape Devices

When you can use two tape devices with the database server, log in as user **informix** at a free terminal.

Verify that LTAPEDEV and TAPEDEV specify different path names that correspond to separate tape devices. When they do, back up the logical-log files. See "Creating a Backup" on page 14-3.

When LTAPEDEV and TAPEDEV are identical, assign a different value to the logical-log tape device (LTAPEDEV) and initiate a logical-log-file backup. Otherwise, your options are to either leave normal database server processing suspended until the backup completes or cancel the backup.

When Only One Tape Device Is Available

When you create a backup with the only available tape device, you cannot back up any logical-log files until you complete the backup. When the logical-log files fill during the backup, normal database server processing halts. You can either abort the backup (using `ctrl-c` *only*) to free the tape device and back up the logical logs to continue processing, or leave normal processing suspended until the backup completes.

You can take steps to prevent this situation. The section "Starting an Automatic Logical-Log Backup" on page 14-12 describes these steps.

When a Backup Terminates Prematurely

When you cancel or interrupt a backup, sometimes the backup progresses to the point where you can consider it complete. When listed in the monitoring information, as described in "Monitoring Backup History Using oncheck" on page 14-10, you know the backup completed.

Monitoring Backup History Using oncheck

You can monitor the history of your last full-system backup using **oncheck.**

Execute **oncheck -pr** to display reserved-page information for the root dbspace. The last pair of reserved pages contains the following information for the most recent backup:

- Backup level (0, 1, or 2)
- Effective date and time of the backup
- Time stamp describing when the backup began (expressed as a decimal)
- ID number of the logical log that was current when the backup began
- Physical location in the logical log of the checkpoint record (that was written when the backup began)

The effective date and time of the backup equals the date and time of the checkpoint that this backup took as its starting point. This date and time could differ markedly from the time when the backup process was started.

For example, when no one accessed the database server after Tuesday at 7 P.M., and you create a backup Wednesday morning, the effective date and time for that backup is Tuesday night, the time of the last checkpoint. In other words, when there has been no activity after the last checkpoint, the database server does not perform another checkpoint at the start of the backup.

Backing Up Logical-Log Files with ontape

You must only use **ontape** to back up logical-log files when you use **ontape** to make your backup tapes.

In addition to backing up logical-log files, you can use **ontape** to switch to the next log file, move logical-log files to other dbspaces, or change the size of the logical log. Instructions for those tasks appear in your *IBM Informix Administrator's Guide*.

Before You Back Up the Logical-Log Files

Before you back up the logical-log files, you need to understand the following issues:

- Whether you need to back up the logical-log files
- When you need to back up the logical-log files
- Whether you want to perform an automatic or continuous backup

For more information on these issues, see "Logical-Log Backup" on page 1-2.

Using Blobspace TEXT and BYTE Data Types and Logical-Log Files

You must keep the following two points in mind when you use TEXT and BYTE data types in a database that uses transaction logging:

- To ensure timely reuse of blobpages, back up logical-log files. When users delete TEXT or BYTE values in blobspaces, the blobpages do not become freed for reuse until you free the log file that contains the delete records. To free the log file, you must back it up.
- When you must back up an unavailable blobspace, **ontape** skips it and makes it impossible to recover the TEXT or BYTE values when it becomes necessary. (However, blobpages from deleted TEXT or BYTE values do become free when the blobspace becomes available even though the TEXT or BYTE values were 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 logical-log file that records the event is not the current logical-log file. For information on switching logical-log files, see your *IBM Informix Administrator's Guide*.

Using /dev/null When You Do Not Need to Recover

When you decide that you do not need to recover transactions or administrative database activities between backups, you can set the database server configuration parameter LTAPEDEV to **/dev/null**.

Warning: When you set LTAPEDEV to **/dev/null**, it has the following implications:

- You can only restore the data that your database server manages up to the point of your most recent backup and any previously backed-up logical-log files.
- When you perform a recovery, you must always perform a full-system restore. (See "Full-System Restores" on page 15-2.) You cannot perform partial restores or restore when the database server is online.

When you set LTAPEDEV to **/dev/null**, the database server marks a logical-log file as backed up (status B) as soon as it becomes full. The database server can then reuse that logical-log file without waiting for you to back it up. As a result, the database server does not preserve any logical-log records.

Fast recovery and rolling back transactions are not impaired when you use **/dev/null** as your log-file backup device. For a description of fast recovery, see

your *IBM Informix Administrator's Guide*. For information about rolling back transactions, see the ROLLBACK WORK statement in the *IBM Informix Guide to SQL: Syntax*.

When to Back Up Logical-Log Files

You must attempt to back up each logical-log file as soon as it fills. You can tell when you can back up a logical-log file because it has a *used* status. For more information on monitoring the status of logical-log files, see your *IBM Informix Administrator's Guide*.

Starting an Automatic Logical-Log Backup

The database server can operate online when you back up logical-log files. To back up all full logical-log files, use the **-a** option of the **ontape** command.

Requesting a Logical-Log Backup

The **-a** option backs up all full logical-log files and prompts you with an option to switch the logical-log files and back up the formerly *current* log.

When the tape mounted on LTAPEDEV becomes full before the end of the logical-log file, **ontape** prompts you to mount a new tape.

When you press the Interrupt key while a backup occurs, the database server finishes the backup and then returns control to you. Any other full logical-log files receive a used status.

To back up all full logical-log files, execute the following command:

```
ontape -a
```

Starting a Continuous Logical-Log File Backup

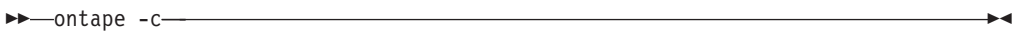
When you do not want to monitor the logical-log files and start backups when the logical-log files become full, you can start a continuous backup.

When you start a continuous backup, the database server automatically backs up each logical-log file as it becomes full. When you perform continuous logical-log file backups, the database server protects you against ever losing more than a partial logical-log file, even in the worst case media failure when a chunk that contains logical-log files fails.

With continuous backups you also do not need to remember to back up the logical-log files, but someone must always make media available for the backup process. Also, you must dedicate the backup device to the backup process.

To start a continuous backup of the logical-log files, use the **-c** option of the **ontape** command.

Starting Continuous Backups



The **-c** option initiates continuous backup of logical-log files. The database server backs up each logical-log file as it becomes full. Continuous backup does not back up the current logical-log file.

The database server can operate in online mode when you start continuous backups. To start continuous logging, execute the following command:

```
ontape -c
```

When the mounted tape specified by LTAPEDEV becomes full before the end of the logical-log file, the database server prompts the operator for a new tape.

You can also create a continuous logical-log file backup to a directory. The logical logs are automatically backed up as long as space is available. For more information, see the *IBM Informix: Administrator's Reference*.

Procedure:

1. Set the LTAPEDEV parameter to an existing directory. Make sure this directory is owned by **informix** and group **informix**.
2. Edit the ALARMPROGRAM script (**$INFORMIXDIR/etc/alarmprogram.sh** on UNIX or Linux or **%INFORMIXDIR%\etc\alarmprogram.bat** on Windows), as follows:
   a. Set the BACKUPLOGS parameter within the file to `Y`.
   b. Change the backup program from `onbar -b -l` to `ontape -a -d`.
3. Restart the database server.

Ending a Continuous Logical-Log Backup

To end continuous logical-log backup, press the Interrupt key (`CTRL-C`).

When you press the Interrupt key while the database server backs up a logical-log file to a local device, all logs that were completely backed up before the interrupt are captured on the tape and are marked as backed up by the database server.

When you press the Interrupt key while the database server waits for a logical-log file to fill (and thus is not backing up any logical-log files), all logs that were backed up before the interrupt reside on the tape and are marked as backed up by the database server.

When you press the Interrupt key while the database server performs a continuous backup to a remote device, any logical-log files that were backed up during this operation can or cannot reside on the tape, and are not marked as backed up by the database server (a good reason why you should not do continuous remote backups).

After you stop continuous logging, you must start a new tape for subsequent log backup operations.

You must explicitly request logical-log backups (using **ontape -a**) until you restart continuous logging.

Devices that Logical-Log Backups Must Use

The **ontape** utility uses parameters defined in the ONCONFIG file to define the tape device for logical-log backups. However, consider the following issues when you choose a logical-log backup device:

- When the logical-log device differs from the backup device, you can plan your backups without considering the competing needs of the backup schedule.
- When you specify **/dev/null** as the logical-log backup device in the ONCONFIG parameter LTAPEDEV, you avoid having to mount and maintain backup tapes. However, you can only recover data up to the point of your most recent backup tape. You cannot restore work done after the backup. See the warning about setting LTAPEDEV to **/dev/null** in "Using /dev/null When You Do Not Need to Recover" on page 14-11.
- When your tape device runs slow, the logical log could fill up faster than you can copy it to tape. In this case, you could consider performing the backup to disk and then copying the disk backup to tape.

Chapter 15. Restoring with ontape

In This Chapter 15-1
Types of Physical Restore 15-2
  Full-System Restores 15-2
  Restores of Dbspaces, Blobspaces, and Sbspaces 15-2
Cold, Warm, or Mixed Restores 15-2
  Cold Restores 15-2
  Warm Restores 15-3
  Mixed Restores 15-3
Performing a Restore 15-4
Restoring the Whole System 15-5
  Gathering the Appropriate Tapes 15-6
    Backup Tapes 15-6
    Logical-Log Tapes 15-6
    File Names when Restoring from Directory 15-6
  Deciding on a Complete Cold or a Mixed Restore 15-6
  Verifying Your Database Server Configuration 15-6
    Setting Shared-Memory Parameters to Maximum Assigned Value 15-6
    Setting Mirroring Configuration to Level-0 Backup State 15-7
    Ensuring That Needed Devices Are Available 15-7
  Performing a Cold Restore 15-7
    Salvaging Logical-Log Files 15-7
    Mounting Tapes During the Restore 15-8
    Restoring Logical Log Files 15-8
    Bringing the Database Server Online When the Restore Is Over 15-8
  Restoring Selected Storage Spaces 15-8
    Gathering the Appropriate Tapes 15-9
    Ensuring That Needed Device Are Available 15-9
    Backing Up Logical-Log Files 15-9
    Performing a Warm Restore 15-9
  Restoring Raw Tables 15-9
Configuring Continuous Log Restore Using ontape 15-10
Renaming Chunks During a Restore 15-10
  Validation Sequence for Renaming Chunks 15-11
  New Chunk Requirements 15-11
  Renaming Chunks with Command Line Options 15-12
  Renaming Chunks with a File 15-12
  Renaming Chunks While Specifying Other Options 15-12
  Renaming a Chunk to a Nonexistent Device 15-12
Restoring from Standard Input 15-13
Restoring Data to a Remote Server 15-13
Simultaneous Backup and Restore Using Standard I/O 15-14

In This Chapter

This section provides instructions for restoring data using **ontape** for the following procedures:

- A whole-system restore
- A restore of selected dbspaces, blobspaces, and sbspaces

Before you start restoring data, you must understand the concepts in "Restore Systems" on page 1-4. As explained in that section, a complete recovery of database server data generally consists of a physical restore and a logical restore.

© Copyright IBM Corp. 1996, 2008

Types of Physical Restore

If a failure causes the database server to go offline, you must restore all the database server data. This type of restore is a *full-system* restore. You can only restore data to the same version of Dynamic Server. When the failure did not cause the database server to go offline, you can restore only the storage spaces that failed. For illustrations of the restore types, see "Warm, Cold, and Mixed Restores" on page 1-4.

Full-System Restores

When your database server goes offline because of a disk failure or corrupted data, it means that a *critical dbspace* was damaged. The following list shows critical dbspaces:

- The root dbspace
- The dbspace that contains the physical log
- A dbspace that contains logical-log files

When you need to restore any critical dbspace, you must perform a full system restore to restore all the data that your database server manages. You must start a full-system restore with a *cold restore*. See "Cold, Warm, or Mixed Restores" on page 15-2.

Restores of Dbspaces, Blobspaces, and Sbspaces

When your database server *does not* go offline because of a disk failure or corrupted data, the damage occurred to a noncritical dbspace, blobspace, or sbspace.

When you do not need to restore a critical dbspace, you can restore only those storage spaces that contain a damaged chunk or chunks. When a media failure occurs in one chunk of a storage space that spans multiple chunks, all active transactions for that storage space must terminate before the database server can restore it. You can start a restore operation before the database server finishes the transactions, but the restore becomes delayed until the database server verifies that you finished all transactions that were active at the time of the failure.

Cold, Warm, or Mixed Restores

When you restore the database server data, you must decide whether you can do it while the database server is offline or online. This decision depends in part on the data that you intend to restore.

Cold Restores

Perform a *cold restore* while the database server is offline. It consists of both a physical restore and a logical restore. You must perform a cold restore to restore any critical dbspaces.

The database server is offline when you begin a cold restore but it goes into recovery mode after it restores the reserved pages. From that point on it stays in recovery mode until either a logical restore finishes (after which it works in

quiescent mode) or you use the **onmode** utility to shift it to another mode.

Dynamic Server

You can rename chunks by specifying new chunks paths and offsets during a cold restore. This option is useful if you need to restore storage spaces to a different disk from the one on which the backup was made. You can rename any type of chunk, including critical chunks and mirror chunks. For more information, see "Renaming Chunks During a Restore" on page 15-10. You can also rename chunks for an external cold restore; see "Renaming Chunks" on page 16-5 for more information.

A cold restore can be performed after a dbspace has been renamed and a level-0 backup or a backup of the rootdbs and renamed dbspace is performed.

End of Dynamic Server

Warm Restores

A warm restore restores noncritical storage spaces while the database server is in online or quiescent mode. It consists of one or more physical restore operations (when you restore multiple storage spaces concurrently), a logical-log backup, and a logical restore.

During a warm restore, the database server replays backed-up logical-log files for the storage spaces that you restore. To avoid overwriting the current logical log, the database server writes the logical-log files that you designate for replay to temporary space. Therefore, a warm restore requires enough temporary space to hold the logical log or the number of log files being replayed, whichever is smaller. For information on how the database server looks for temporary space, see the discussion of **DBSPACETEMP** in your *IBM Informix Administrator's Guide*.

Warning: Make sure enough temporary space exists for the logical-log portion of the warm restore; the maximum amount of temporary space that the database server needs equals the size of all the logical-log files.

Dynamic Server

A warm restore can be performed after a dbspace has been renamed and a level-0 archive of the rootdbs and renamed dbspace is taken.

End of Dynamic Server

Mixed Restores

A *mixed restore* is a cold restore followed by a warm restore. A mixed restore restores some storage spaces during a cold restore (the database server is offline) and some storage spaces during a warm restore (the database server is online). You could do a mixed restore when you perform a full-system restore, but you need to provide access to a particular table or set of tables as soon as possible. In this case, perform a cold restore to restore the critical dbspaces and the dbspaces that contain the important tables.

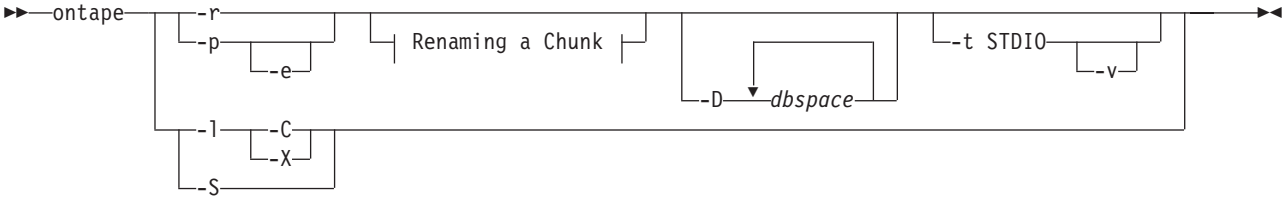
A cold restore takes less total time to restore all your data than a mixed restore, even though the database server is online during part of a mixed restore because a

mixed restore requires two logical restores (one for the cold restore and one for the warm restore). A mixed restore, however, requires the database server to go *offline* for less time than a cold restore.

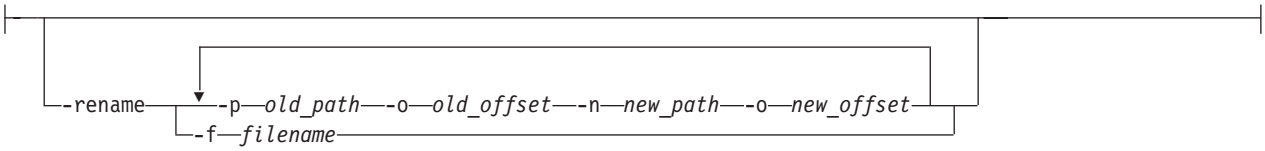
The dbspaces not restored during the cold restore do not become available until after the database server restores them during a warm restore, even though a critical dbspace possibly did not damage them.

Performing a Restore

Use the **-r** option to perform a full physical and logical restore of the database server data with **ontape**. Use the **-D** option to restore selected storage spaces. Use the **-rename** option to rename chunks during the restore.



Renaming a Chunk:



Element	Purpose	Key Considerations
-C	Restores logs from the current logical log tape without sending prompts to mount the tape.	The server is placed in suspend log restore state, and the command exits after the last applicable log is restored. The server sends a prompt if a log spans tapes.
-D	Directs **ontape** to restore only the storage spaces you specify.	The database server must go into online or quiescent mode to do a warm restore. When you use the **-D** option, you can restore selected storage spaces. When you do not specify the **-D** option, **ontape** performs a full-system restore. The database server must go offline to do a full-system restore. For more information, see "Restoring Selected Storage Spaces" on page 15-8.
dbspace	Is the name of a storage space to restore.	You can specify multiple storage spaces, but you must include the root dbspace.
-e	Directs **ontape** to perform an external restore	For more information, see Chapter 16, "Performing an External Backup and Restore," on page 16-1. This option is compatible with renaming chunks for external cold restores.

Element	Purpose	Key Considerations
-f *filename*	Specifies a file containing the names and offsets of chunks to be renamed and their new locations. Use to rename a large number of chunks at one time.	The filename can be any valid UNIX or Windows filename, including simple (**listfile_1**), relative (**../backup_lists/listfile_2** or **..\backup_lists\listfile2**), and absolute (**/usr/informix/backup_lists/listfile3** or **c:\informix\backup_lists\listfile3**) filenames. In the file, list the old chunk path name and offset and the new chunk path name and offset, with a blank space or a tab between each item. Put information for each chunk on a separate line. Blank lines are ignored. Begin comment lines with a # symbol.
-l	Directs **ontape** to perform a logical restore.	The **-l** option restores data from the logical-log backup tapes you created after (and including) your last level-0 backup.
-p	Directs **ontape** to perform a physical data restore.	The **-p** option restores data from the backup tape you created after (and including) your last level-0 backup. During the restore, the database server is in single-user mode.
-p *old_path* **-o** *old_offset* **-n** *new_path* **-o** *new_offset*	Specifies the chunk to be renamed and its new location. Use to rename one or more chunks at one time.	The variables for this element are: • *old_path* is the current path and filename of the chunk • *old_offset* is the current offset of the chunk, in kilobytes • *new_path* is the new path and filename of the chunk • *new_offset* is the new offset of the chunk
-r	Directs **ontape** to perform a data restore (both physical and logical).	The **-r** option restores data from the backup tape and the logical-log backup tapes you created after (and including) your last level-0 backup.
-rename	Directs **ontape** to rename the specified chunks.	For more information on renaming chunks during a restore, see "Renaming Chunks During a Restore" on page 15-10.
-S	Directs **ontape** to perform a logical log salvage.	If you want to salvage logical logs, you must use the **-S** option prior to performing a restore from standard input. The LTAPEDEV configuration parameter must be set to the logical log tape device.
-t STDIO	Directs **ontape** to restore from standard input.	The **-t** option overrides the value of the TAPEDEV configuration parameter for the current restore.
-v	Directs **ontape** to write informational message to stderr during a restore from standard input.	Verbose mode is useful for monitoring the progress of a restore from standard input.
-X	Quiesces a server in logical restore suspend state without restoring additional logs.	Include this option with `-r -l` to end continuous log restore of logical logs.

Restoring the Whole System

This section outlines the steps you need to perform to restore your entire database server with **ontape**. The following list describes the main steps in a full-system restore:

1. Gather the appropriate tapes.
2. Decide on a complete cold or a mixed restore.
3. Verify your database server configuration.

4. Perform a cold restore.

Familiarize yourself with these instructions before you attempt a full-system restore.

Gathering the Appropriate Tapes

Gather the appropriate backup and logical-log tapes.

Backup Tapes

Before you start your restore, gather together all the tapes from your latest level-0 backup that contain the storage spaces you are restoring and any subsequent level-1 or level-2 backups.

Identify the tape that has the latest level-0 backup of the root dbspace on it; you must use this tape first.

Logical-Log Tapes

Gather together all the logical-log tapes from the backup after the latest level-0 backup of the storage spaces you are restoring.

File Names when Restoring from Directory

When restoring from a file system directory, ontape requires that storage-space archive and logical-log backup files be named as specified by the TAPEDEV and LTAPEDEV parameters. If files have been renamed, including by ontape because of repeated archives and backups, files must be manually renamed to their original file names. To learn about the naming conventions for storage-space archive files and logical-log backup files, see "Back Up to a Directory" on page 14-6 for the naming conventions of these files.

Deciding on a Complete Cold or a Mixed Restore

As mentioned in "Cold, Warm, or Mixed Restores" on page 15-2, when you restore your entire database server, you can restore the critical dbspaces (and any other storage spaces you want to come online quickly) during a cold restore, and then restore the remaining storage spaces during a warm restore. Decide before you start the restore if you want a completely cold restore or a mixed restore.

Verifying Your Database Server Configuration

During a cold restore, you cannot set up shared memory, add chunks, or change tape devices. Thus, when you begin the restore, the current database server configuration must remain compatible with, and accommodate, all parameter values assigned after the time of the most recent backup.

For guidance, use the copies of the configuration file that you create at the time of each backup. However, do not set all current parameters to the same values as were recorded at the last backup. Pay attention to the following three groups of parameters:

- Shared-memory parameters
- Mirroring parameters
- Device parameters

Setting Shared-Memory Parameters to Maximum Assigned Value

Make sure that you set your current shared-memory parameters to the *maximum* value assigned after the level-0 backup. For example, if you decrease the value of USERTHREADS from 45 to 30 sometime after the level-0 backup, you must begin

the restore with USERTHREADS set at 45, and not at 30, even though the configuration file copy for the last backup could register the value of USERTHREADS set at 30. (When you do not possess a record of the maximum value of USERTHREADS after the level-0 backup, set the value as high as you think necessary. You could reassign values to BUFFERPOOL, LOCKS, and TBLSPACES as well because the minimum values for these three parameters are based on the value of USERTHREADS.)

Setting Mirroring Configuration to Level-0 Backup State

Verify that your current mirroring configuration matches the configuration that was in effect at the time of the last level-0 backup. Because it is recommended that you create a level-0 backup after each change in your mirroring configuration, this creates no problems. The most critical parameters are the mirroring parameters that appear in the ONCONFIG configuration file, MIRRORPATH and MIRROROFFSET.

Ensuring That Needed Devices Are Available

Verify that the raw devices or files that you used for storage (of the storage spaces being restored) after the level-0 backup are available.

For example, if you drop a dbspace or mirroring for a dbspace after your level-0 backup, you must make the dbspace or mirror chunk device available to the database server when you begin the restore. When the database server attempts to write to the chunk and cannot find it, the restore does not complete. Similarly, if you add a chunk after your last backup, you must make the chunk device available to the database server when it begins to roll forward the logical logs.

Performing a Cold Restore

To perform a cold restore, the database server must be offline.

You must log in as user **informix** or **root** to use **ontape**. Execute the following **ontape** command to restore all the storage spaces:

```
ontape -r
```

When you perform a mixed restore, you restore only some of the storage spaces during the cold restore. You must restore at least all the critical dbspaces, as the following example shows:

```
ontape -r -D rootdbs llogdbs plogdbs
```

Salvaging Logical-Log Files

Before the cold restore starts, the console prompts you to salvage the logical-log files on disk. To salvage the logical-log files, use a new tape. It saves log records that you did not back up and enables you to recover your database server data up to the point of the failure.

The following example shows a log salvage:

```
...
Continue restore? (y/n) y
Do you want to back up the logs? (y/n) y

Please mount tape 1 on /dev/ltapedev and press Return to continue.
Would you like to back up any of logs 31 - 32? (y/n) y
Logical logs 31 - 32 may be backed up.
Enter the id of the oldest log that you would like to backup? 31

Please label this tape as number 1 in the log tape sequence.
```

```
This tape contains the following logical logs:
    31-32
Log salvage is complete, continuing restore of archive.
Restore a level 1 archive (y/N) y
Ready for level 1 tape
...
```

Mounting Tapes During the Restore

During the cold restore, **ontape** prompts you to mount tapes with the appropriate backup files.

When restoring from a directory, the prompt specifies the absolute path name of the directory. Before responding to the prompt, you can copy or rename the file in the directory.

You can avoid the prompt by using the ontape -d option. When using this option, ensure that storage-space archive and logical-log backup files exist in the directory, as specified by the TAPEDEV and LTAPEDEV parameters. **Ontape** scans the directories for the files and uses them for the restore. After restoring the newest applicable logical-log backup file, ontape automatically commits the restore and brings the IDS instance into quiescent mode.

Restoring Logical Log Files

When you perform a mixed restore, you must restore all the logical-log files backed up after the last level-0 backup.

When you perform a full restore, you can choose not to restore logical-log files. When you do not back up your logical-log files or choose not to restore them, you can restore your data only up to the state it was in at the time of your last backup. For more information, see “Backing Up Logical-Log Files with ontape” on page 14-10.

To restore the logical logs, use the following command:

```
ontape -l
```

Bringing the Database Server Online When the Restore Is Over

At the end of the cold restore, the database server is in quiescent mode. You can bring the database server online at this point and continue processing as usual.

When you restore only some of your storage spaces during the cold restore, you can start a warm restore of the remaining storage spaces after you bring the database server online.

Restoring Selected Storage Spaces

This section outlines the steps that you must perform during a restore of selected storage spaces with **ontape** while the database server is in online or quiescent mode (a warm restore). During a warm restore, you do not need to worry about shared-memory parameters as you do for cold restores.

The following list describes the main steps in a warm restore:

1. Gather the appropriate tapes.
2. Verify your database server configuration.
3. Back up logical-log files.
4. Perform a warm restore.

Before you attempt a restore, familiarize yourself with these instructions.

Gathering the Appropriate Tapes

Gather the appropriate backup and logical-log tapes.

Backup Tapes: Before you start your restore, gather together all the tapes from your latest level-0 backup that contain the storage spaces you are restoring and any subsequent level-1 or level-2 backups.

Logical-Log Tapes: Gather together all the logical-log tapes from the logical-log backup after the latest level-0 backup of the storage spaces you are restoring.

Ensuring That Needed Device Are Available

Verify that storage devices and files are available before you begin a restore. For example, when you drop a dbspace or mirroring for a dbspace after your level-0 backup, you must ensure that the dbspace or mirror chunk device is available to the database server when you begin the restore. If the storage device is not available, the database server cannot write to the chunk and the restore fails.

When you add a chunk after your last backup, you must ensure that the chunk device is available to the database server when it rolls forward the logical logs.

Backing Up Logical-Log Files

Before you start a warm restore (even when you perform the warm restore as part of a mixed restore), you must back up your logical-log files. See "Backing Up Logical-Log Files with ontape" on page 14-10.

After the warm restore, you *must* roll forward your logical-log files to bring the dbspaces that you are restoring to a state of consistency with the other dbspaces in the system. Failure to roll forward the logical log after restoring a selected dbspace results in the following message from **ontape**:

```
Partial system restore is incomplete.
```

Performing a Warm Restore

To perform a warm restore, the database server must operate in online or quiescent mode.

You must log in as user **informix** or **root** to use **ontape**. To restore selected storage spaces, execute the **ontape** command, with the options that the following example shows:

```
ontape -r -D dbspace1 dbspace2
```

You cannot restore critical dbspaces during a warm restore; you must restore them as part of a cold restore, described in "Restoring the Whole System" on page 15-5.

During the restore, **ontape** prompts you to mount tapes with the appropriate backup files.

At the end of the warm restore, the storage spaces that were down go online.

Restoring Raw Tables

When you use **ontape** to restore a raw table, it contains only data that existed on disk at the time of the backup. Because raw tables are not logged, any changes that occurred since the last backup cannot be restored. For more information, see "Backing Up Raw Tables" on page 14-9 and the *IBM Informix Administrator's Guide*.

Configuring Continuous Log Restore Using ontape

Use continuous log restore to restart a log restore with newly available logs after all currently available logs have been restored. For more information, see "Continuous Log Restore" on page 7-5.

Prerequisites:

The version of Dynamic Server must be identical on both the primary and secondary systems.

Procedure:

1. On the primary system, perform a level-0 archive with the following command:

   ```
   ontape -s -L 0
   ```

2. On the secondary system, copy the files or mount the tape (as assigned by LTAPEDEV) and perform a physical restore with the following command:

   ```
   ontape -p
   ```

   Respond to the following prompts:

   ```
   Continue restore? Y
   Do you want to back up the logs? N
   Restore a level 1 archive? N
   ```

   After the physical restore completes, the database instance waits in fast recovery mode to restore logical logs.

3. On the primary system, back up logical logs with the following command:

   ```
   ontape -a
   ```

4. On the secondary system, copy the files or mount the tape that contains the backed up logical logs from the primary system. Perform a logical log restore with the following command:

   ```
   ontape -l -C
   ```

5. Repeat steps 3 and 4 for all logical logs that are available to back up and restore.

6. If you are doing continuous log restore on a secondary system as an emergency standby, run the following commands to complete restoring logical logs and quiesce the server.

   If logical logs are available to restore:

   ```
   ontape -l
   ```

   After all available logical logs are restored:

   ```
   ontape -l -X
   ```

Renaming Chunks During a Restore

You can rename chunks during a cold restore with **ontape**. This option is useful if you need to restore storage spaces to a different disk from the one on which the backup was made. You can rename any type of chunk, including critical chunks and mirror chunks.

The **ontape** rename chunk restore only works for cold restores.

The critical dbspaces (for example, the rootdbs) must be restored during a cold restore. If you do not specify the list of dbspaces to be restored, then the server

will restore the critical dbspaces and all the other dbspaces. But if you specify the list of dbspaces to be restored, then the critical dbspaces must be included in the list.

For the syntax of renaming chunks with **ontape**, see "Performing a Restore" on page 15-4.

Tip: If you use symbolic links to chunk names, you might not need to rename chunks; you need only edit the symbolic name definitions. For more information, see the *IBM Informix Administrator's Guide*.

You can rename chunks during an external cold restore. See "Renaming Chunks" on page 16-5 for more information.

Validation Sequence for Renaming Chunks

During a cold restore, **ontape** performs the following validations to rename chunks:

1. It validates that the old chunk path names and offsets exist in the archive reserved pages.
2. It validates that the new chunk path names and offsets do not overlap each other or existing chunks.
3. If renaming the primary root or mirror root chunk, it updates the ONCONFIG file parameters ROOTPATH and ROOTOFFSET, or MIRRORPATH, and MIRROROFFSET. The old version of the ONCONFIG file is saved as **$ONCONFIG.***localtime*.
4. It restores the data from the old chunks to the new chunks (if the new chunks exist).
5. It writes the rename information for each chunk to the online log.

If either of the validation steps fails, the renaming process stops and **ontape** writes an error message to the **ontape** activity log.

Warning: Perform a level-0 archive after you rename chunks; otherwise your next restore will fail.

Important: If you add a chunk after performing a level-0 archive, that chunk cannot be renamed during a restore. Also, you cannot safely specify that chunk as a new path in the mapping list.

Important: Renaming chunks for database servers participating in HDR involves a significant amount of offline time for both database servers. For more information, see the *IBM Informix Administrator's Guide*.

New Chunk Requirements

To rename a chunk, follow these guidelines for new chunks:

- The new chunk does not need to exist

  You can install the new chunk later and perform a warm restore of a storage space containing it. If you specify a nonexistent chunk, **ontape** records the rename information in the chunk reserved pages, but does not restore the data. The renamed (but not restored) chunks have a status of offline, designated by `D`, in the **onstat -d** chunk status command output.
- New chunks must have the proper permissions.

Rename operations fail unless the chunks have the proper permissions. For more information, see the *IBM Informix Administrator's Guide*.

Renaming Chunks with Command Line Options

To rename the chunks by supplying information on the command line, use this command:

```
ontape -r -rename -p /chunk1 -o 0 -n /chunk1N -o 20000
         -rename -p /chunk2 -o 10000 -n /chunk2N -o 0
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming Chunks with a File

To rename the chunks by supplying a file named **listfile**, use this command:

```
ontape -r -rename -f listfile
```

The contents of the **listfile** file are:

```
/chunk1 0 /chunk1N 20000
/chunk2 10000 /chunk2N 0
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming Chunks While Specifying Other Options

To rename the chunks using command-line options while performing a restore of **dbspace1** and **dbspace2** where the **rootdbs** is the rootdbs, use the following command:

```
ontape -r -rename -p /chunk1 -o 0 -n /chunk1N -o 20000
         -rename -p /chunk2 -o 10000 -n /chunk2N -o 0
         -D rootdbs dbspace1 dbspace2
```

Alternatively, to rename the chunks using file while performing a restore of **dbspace1** and **dbspace2**, use the following command:

```
ontape -r -rename -f listfile -D rootdbs dbspace1 dbspace2
```

Perform a level-0 archive after the rename and restore operation is complete.

Renaming a Chunk to a Nonexistent Device

To rename a chunk to a device that does not yet exist, you specify the new path name, but you do not restore its storage spaces until after you install the physical device. This option is useful if you need to rename a chunk and it is convenient to perform a cold restore before you install the new device. When the new chunk device is ready, you can perform a warm restore of a storage space onto it.

You can combine renaming chunks with existing devices and renaming chunks with nonexistent devices in the same rename operation. This example shows how to rename a single chunk to a nonexistent device name.

The following table lists example values for the chunks used in this example.

Storage Space	Old Chunk Path	Old Offset	New Chunk Path	New Offset
sbspace1	/chunk3	0	/chunk3N	0

To rename a chunk to a nonexistent device:

1. Rename the chunk:

```
ontape -r -rename -p /chunk3 -o 0 -n /chunk3N -o 0
```

2. When the following prompt appears, enter y to continue:

```
The chunk /chunk3N does not exist. If you continue, the restore
may fail later for the dbspace which contains this chunk.
Continue without creating this chunk? (y/n)
```

   The chunk **/chunk3** is renamed to **/chunk3N**, but the data has not yet been restored to **/chunk3N**.

3. Perform a level-0 archive.
4. Add the physical device for **/chunk3N**.
5. Perform a warm restore of **sbspace1**:

```
ontape -r -D sbspace1
```

6. Perform a level-0 archive.

Restoring from Standard Input

To perform a restore from standard input, you must first have performed a backup to standard output. For more information, see "Back Up to Standard Output" on page 14-6.

When you perform a restore from standard input, **ontape** does not prompt you for options or information. If **ontape** cannot perform the operation with the information you provided in the restore command, **ontape** exits with an appropriate error. Restoring from standard input differs from restoring from tapes in the following ways:

- No logical restore or logical log salvage occurs.

  To perform a logical restore, use the **ontape -l** command after the physical restore.

  To salvage logical logs, use the **ontape -S** command prior to the physical restore.
- You are not prompted to confirm the restore. Informational messages about the archive are sent to stderr.

  If you detect a problem, you can interrupt the restore during the 10 second delay between the completion of the archive information and starting the database server.

In the following example, **ontape** performs a physical restore from the file **level_0_archive**, which contains the archive previously performed to standard output:

```
cat /home/level_0_archive | ontape -p
```

In the following example, **ontape** performs a restore of a level-0 archive, followed by a restore of a level-1 archive:

```
cat /home/level_0_archive /home/level_1_archive | ontape -r
```

In the following example, **ontape** performs a restore of sbspace1:

```
cat/home/level_0_archive | ontape -r -D spspace1 -t STDIO
```

When these restores are completed, the database server is left in single-user mode.

Restoring Data to a Remote Server

You can restore data to a remote server with the following command:

```
ontape -s -L 0 -F | rsh remote_server "ontape -p"
```

However, the process might hang after completing successfully. You have three primary options:

- Terminate the remote shell process
- Execute the remote shell from the remote server with the following command:

  ```
  rsh local_server "ontape -s -L 0 -F" | ontape -p
  ```

- Redirect the standard output (stdout) and standard error (stderr) on the remote server with the following command from the sh or bash shell:

  ```
  ontape -p >/dev/null 2>&1
  ```

- You can simplify this redirection by placing it in a shell script, ontape.sh, on the remote server. You can issue the following command from the local server:

  ```
  ontape -s -L 0 -F | rsh remote_server /my/path/ontape.sh
  ```

- The shell script ontape.sh contains the following text:

  ```
  #!/bin/sh
  #define some IDS environment variables, such as

  INFORMIXDIR=/... ; export INFORMIXDIR
  INFORMIXSQLHOSTS=/...; export
  INFORMIXSQLHOSTS ONCONFIG=/...; export ONCONFIG
  INFORMIXSERVER=/...; export INFORMIXSERVER
  PATH=/...; export PATH
  # invoke ontape with stdout/stderr redirection

  ontape -p >/dev/null 2>&1
  ```

Simultaneous Backup and Restore Using Standard I/O

To clone a database server or quickly set up High-Availability Data Replication (HDR), you can perform a simultaneous backup to standard output and restore from standard input. If you perform the backup and restore solely to duplicate a database server, use the **-F** option to prevent the archive from being saved.

Note: On HDR, the secondary server can restore only level-0 archives.

For example, the following command loads data into the secondary server of an HDR pair (named **secondary_host**):

```
ontape -s -L 0 -F| rsh secondary_host "ontape -p"
```

This command performs a fake level-0 archive of the database server on the local computer, pipes the data to the remote computer using the **rsh** system utility, and performs a physical restore on the remote computer by reading the data directly from the pipe.

Note: The previous example requires that the INFORMIXDIR, INFORMIXSERVER, INFORMIXSQLHOSTS, and ONCONFIG environment variables are set in the default environment for the user on the remote computer on which the command is executed. The user must be **informix** or **root**.

Chapter 16. Performing an External Backup and Restore

In This Chapter 16-1
Recovering Data Using an External Backup and Restore 16-1
Data That is Backed Up in an External Backup 16-1
Rules for an External Backup 16-2
Performing an External Backup 16-2
Preparing for an External Backup 16-3
Blocking and Unblocking Dynamic Server 16-3
Tracking an External Backup 16-3
Data That is Restored in an External Restore. 16-4
Using External Restore Commands 16-4
Rules for an External Restore 16-5
Renaming Chunks 16-5
Performing an External Restore 16-5
Cold External Restore Procedure. 16-5
Examples of External Restore Commands. 16-6
Initializing HDR with an External Backup and Restore 16-6

In This Chapter

This chapter discusses recovering data using external backup and restore using the **ontape** utility.

Recovering Data Using an External Backup and Restore

An external backup and restore eliminates the downtime of systems because the backup and restore operations are performed external to the Informix system. The **ontape** utility does not move the data during the backup or physical restore. An external backup allows you to copy disks that contain storage-space chunks without using **ontape**. When disks fail, replace them and use third-party software to restore the data, then use **ontape** for the logical restore. For more information, see "Data That is Restored in an External Restore" on page 16-4.

The following are typical scenarios for external backup and restore:

- *Availability with disk mirroring.* If you use hardware disk mirroring, you can get your system online faster with external backup and restore than with conventional **ontape** commands.
- *Cloning.* You can use external backup and restore to clone an existing production system for testing or migration without disturbing the production system.

Data That is Backed Up in an External Backup

Before you begin an external backup, block the database server. Blocking forces a checkpoint, flushes buffers to disk, and blocks user transactions that involve temporary tables. During the blocking operation, users can access that database server in read-only mode. Then you can physically back up or copy the data to another set of disks or storage media using operating-system or third-party tools. When you complete the external backup, unblock the database server so that transactions can resume. You should include all the chunk files in each storage space and administrative files, such as **ONCONFIG**, in an external backup.

© Copyright IBM Corp. 1996, 2008

Important: To make tracking backups easier, it is recommended that you back up all storage spaces in each external backup.

The **ontape** utility treats an external backup as equivalent to a level-0 backup. You cannot perform an external backup and then use **ontape** to perform a level-1 backup, or vice versa because **ontape** does not have any record of the external backup. For more information, see "Performing an External Restore" on page 16-5.

Rules for an External Backup

Before you begin an external backup, keep the following rules in mind:

- The database server must be online or quiescent during an external backup.
- Use **ontape** to back up all logical logs including the current log so that you can restore the logical logs at the end of the external restore.
- Suspend continuous logical-log backups before you block the database server for an external backup. After the external backup is complete, resume the continuous logical-log backup.
- Wait until all **ontape** backup sessions have completed before you block the database server. If any backup sessions are active, the block command displays an error message.
- Any OLTP work or queries are suspended while the database server is blocked. They resume after the database server is unblocked.
- All critical dbspaces of the database server instance must be backed up together simultaneously within the same command bracket of **onmode -c block ... onmode -c unblock**. Backups of different critical dbspaces done at different times cannot be restored to a consistent system.

Important: Because the external backup is outside the control of **ontape**, you must track these backups manually. For more information, see "Tracking an External Backup" on page 8-7.

Performing an External Backup

The database server must be online or in quiescent mode during an external backup.

To perform an external backup without disk mirroring:

1. To obtain an external backup, block the database server. The system takes a checkpoint and suspends all update transactions. Users can access the database server in read-only mode.

   Use the following command:

   ```
   onmode -c block
   ```

2. To back up the storage spaces and administrative files, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or **copy** on Windows, or a file-backup program. You must back up all chunks in the storage spaces.
3. To allow normal operations to resume, unblock the database server.

   Use the following command:

   ```
   onmode -c unblock
   ```

4. Back up all the logical logs including the current log so that checkpoint information is available for the external restore.

   Warning: Because external backup is not done through **ontape**, you must ensure that you have a backup of the current logical log from the

time when you execute the **onutil** *EBR BLOCK* or **onmode -c block** command. Without a backup of this logical-log file, the external backup is not restorable.

5. After you perform an external backup, back up the current log.

   Use the following command to back up the current log:

   ```
   ontape -a
   ```

If you lose a disk or the whole system, you are now ready to perform an external restore.

Preparing for an External Backup

This section describes the commands used to prepare for an external backup. For the procedure, see "Performing an External Backup" on page 16-2.

Blocking and Unblocking Dynamic Server

This section shows the syntax of the block and unblock commands.

Element	Purpose	Key Considerations
-c	Performs a checkpoint and blocks or unblocks the database server	None.
block	Blocks the database server from any transactions	Sets up the database server for an external backup. While the database server is blocked, users can access it in read-only mode. Sample command: `onmode -c block`
unblock	Unblocks the database server, allowing data transactions and normal database server operations to resume	Do not unblock until the external backup is finished. Sample command: `onmode -c unblock`

Tracking an External Backup

The database server and **ontape** do *not* track external backups. To track the external backup data, use a third-party storage manager or track the data manually. Table 16-1 shows which items we recommend that you track in an external backup.

Table 16-1. Items to Track When You Use External Backup and Restore

Items to Track	Examples
Full path names of each chunk file for each backed up storage space	**/work/dbspaces/rootdbs** (UNIX) **c:\work\dbspaces\rootdbs** (Windows)
Object type	Critical dbspaces, noncritical storage spaces
ins_copyid_hi and **ins_copyid_lo**	Copy ID that the storage manager assigns to each backup object
Backup date and time	The times that the database server was blocked and unblocked
Backup media	Tape volume number or disk path name
Database server version	Version 11.50

Data That is Restored in an External Restore

If you lose a disk or the whole system, you can externally restore data only if it was externally backed up. You must use the same third-party utility for both the external backup and restore. To externally restore the storage spaces, copy the backed up data to disk. Use the **ontape -p -e** command to mark the storage spaces as physically restored, replay the logical logs with the **ontape -l** command, and bring the storage spaces back online. If you do not specify an external restore command, the database server cannot update the status of these storage spaces to online.

You can only perform a cold external restore with ontape. A cold external restore marks storage spaces as physically restored, then performs a logical restore of all storage spaces.

Warning: When you perform a cold external restore, **ontape** does not first attempt to salvage logical-log files from the database server because the external backup has already copied over the logical-log data.

To salvage logical logs, perform **ontape -S** before you copy the external backup and perform the external restore (**ontape -p -e**).

Using External Restore Commands

Use the **ontape -p -e** command to perform a cold external restore. This command marks the storage spaces as physically restored. The following diagram shows the external physical restore syntax.

Performing an External Physical Restore

Element	Purpose	Key Considerations
-e	Specifies an external restore	Must be used with the **-p** option.
-p	Specifies a physical restore	In a cold restore, if you do not specify storage space names, all of them are marked as restored. After the physical restore completes, you must perform a logical restore.

Use the **ontape -l** command to perform a logical restore. For more information, see "Performing a Restore" on page 15-4.

Rules for an External Restore

Before you begin an external restore, keep the following rules in mind:

- You must externally restore from an external backup. Although the external backup is treated as a level-0 backup, it might actually be a non-Informix incremental backup.
- You cannot externally restore temporary dbspaces.
- You cannot externally restore from regular **ontape** backups.
- You cannot verify that you are restoring from the correct backup and that the storage media is readable using **ontape**.
- If the external backups are from different times, the external restore uses the beginning logical log from the oldest backup.
- Salvage the logical logs (**ontape -l**) before you switch the disks that contain the critical storage spaces.
- If you are restoring critical dbspaces, the database server *must* be offline.
- If you are restoring the rootdbs, disable mirroring during the restore.
- The external backups of all critical dbspaces of the database server instance must have been simultaneous. All critical dbspaces must have been backed up within the same **onmode -c block ... onmode -c unblock** command bracket.

Renaming Chunks

You can rename chunks in an external cold restore using the rename options syntax for other restores. Use the following commands to rename chunks during an external cold restore:

```
ontape -p -e -rename -f filename
```

or

```
ontape -p -e -rename -p old_path -o old_offset-n new_path-o new_offset
```

Performing an External Restore

This section describes procedures for performing cold external restores.

Cold External Restore Procedure

If you specify the **ontape -p -e** command in a cold restore, you must restore all storage spaces. Use the **ontape -p -e** command to restore all or specific storage spaces.

To perform a cold external restore:

1. Shut down the database server.

   To shut down the database server, use the **onmode -ky** command.

2. To restore the storage spaces from an external backup, use a copy command, such as **cp**, **dd**, or **tar** on UNIX or a file-backup program.

   You must restore the storage spaces to the same path as the original data and include all the chunk files.

3. To perform an external restore of all storage spaces followed by a logical restore, use the following commands:

   ```
   ontape -p -e
   ontape -l
   ```

To perform a physical external restore of specific storage spaces, followed by a logical restore, use these commands:

```
ontape -p -e rootdbs
ontape -p -e critical_space1
ontape -p -e other_dbspaces
ontape -l
```

This step brings the database server to fast-recovery mode. The **ontape** utility and the database server roll forward the logical logs and bring the storage spaces online.

Examples of External Restore Commands

The following table contains an example of external restore commands.

Action	External Restore Command	Comments
Physical external restore and logical restore	**ontape -p -e** **ontape -l**	The system restores the logical logs from the oldest external backup.
External cold restore with renamed chunks	**ontape -p -e -rename -f**	

Initializing HDR with an External Backup and Restore

You can use external backups to initialize High-Availability Data Replication (HDR).

To initialize HDR with an external backup and restore:

1. Block the source database server with the following command:

   ```
   onmode -c block
   ```

2. Externally back up all chunks on the source database server.
3. When the backup completes, unblock the source database server:

   ```
   onmode -c unblock
   ```

4. Make the source database server the primary server:

   ```
   onmode -d primary secondary_servername
   ```

5. On the target database server, restore the data from the external backup with a copy or file-backup program.
6. On the target database server, restore the external backup of all chunks:

   ```
   ontape -p -e
   ```

7. Make the target database server the secondary server:

   ```
   onmode -d secondary primary_servername
   ```

8. If the logical-log records written to the primary database server since step 1 still reside on the primary database server disk, the secondary database server reads these records to perform the logical recovery. Otherwise, perform the logical recovery:

   ```
   ontape -l
   ```

   The `database server operational` messages appear in the message log on the primary and secondary servers.

Part 4. Administering Backup and Restore

© Copyright IBM Corp. 1996, 2008

Chapter 17. Restoring Table-level Data

In This Chapter . . . 17-2
Overview of archecker . . . 17-2
  Configuration File . . . 17-3
  Schema Command File . . . 17-3
  Table-Level Restore and Locales . . . 17-3
Data Restore with archecker . . . 17-3
  Physical Restore . . . 17-4
  Logical Restore . . . 17-4
    The Stager . . . 17-5
    The Applier. . . . 17-5
Restoring Data with archecker . . . 17-5
  Syntax . . . 17-5
  Examples . . . 17-7
    Manually Controlling a Logical Restore . . . 17-7
    Performing a Restore with Multiple Storage Managers (XPS) . . . 17-7
    Performing a Parallel Restore . . . 17-8
    Restoring Transformed Data . . . 17-8
    When to Delete Restore Files . . . 17-9
The archecker Schema Reference. . . . 17-9
  The CREATE TABLE Statement . . . 17-9
    Syntax . . . 17-9
    Usage. . . . 17-9
    Examples . . . 17-10
  The CREATE EXTERNAL TABLE Statement . . . 17-10
    Syntax . . . 17-10
    Usage . . . 17-11
  The DATABASE Statement . . . 17-11
    Syntax . . . 17-11
    Usage . . . 17-11
    Examples . . . 17-11
  The INSERT Statement . . . 17-12
    Syntax . . . 17-12
    Examples . . . 17-13
  The RESTORE Statement . . . 17-13
    Syntax . . . 17-13
    Usage . . . 17-13
    Example . . . 17-13
  The SET Statement . . . 17-13
    Syntax . . . 17-13
    Examples . . . 17-14
  SQL Comments . . . 17-14
  Schema Command File Examples . . . 17-14
    Simple Schema Command File . . . 17-14
    Restoring a Table From a Previous Backup . . . 17-14
    Restoring to a Different Table . . . 17-15
    Extracting a Subset of Columns. . . . 17-15
    Using Data Filtering . . . 17-15
    Restoring to an External Table . . . 17-15
    Restoring Multiple Tables. . . . 17-16
    Performing a Distributed Restore . . . 17-16
The archecker Configuration Parameter Reference . . . 17-16
  AC_DEBUG . . . 17-17
  AC_IXBAR . . . 17-17
  AC_LTAPEBLOCK . . . 17-17
  AC_LTAPEDEV (IDS) . . . 17-18

© Copyright IBM Corp. 1996, 2008

AC_MSGPATH 17-18
AC_PAGESIZE (XPS) 17-18
AC_RESTORE_FILTER 17-18
AC_SCHEMA 17-18
AC_STORAGE 17-19
AC_TAPEBLOCK 17-19
AC_TAPEDEV (IDS) 17-19
AC_VERBOSE 17-19

In This Chapter

This chapter describes how to use the **archecker** utility to perform point-in-time table-level restores that extract tables or portion of tables from archives and logical logs. For information on using the **archecker** utility to verify backups, see Chapter 6, "Verifying Backups," on page 6-1.

The **archecker** utility restores tables by specifying the source table to be extracted, the destination table where the data is placed, and an INSERT statement that links the two tables.

Overview of archecker

IBM Informix servers provide several utilities for recovering data from an archive. Which utility you should use depends on the situation.

The **archecker** utility is useful where portions of a database, a table, a portion of a table, or a set of tables need to be recovered. It is also useful in situations where tables need to be moved across server versions or platforms.

Use **archecker** in the following situations:

- Restore data

  You can use the **archecker** utility to restore a specific table or set of tables that have previously been backed up with ON-Bar or **ontape.** These tables can be restored to a specific point in time. This is useful, for example, to restore a table that has accidentally been dropped.
- Copy data

  The **archecker** utility can also be used as a method of copying data. For example, you can move a table from the production system to another system.

  The **archecker** utility is more efficient than other mechanisms for copying data. Because **archecker** extracts data as text, it can copy data between platforms or server versions.
- Migrate data

  You can also use the **archecker** utility as a migration tool to move a table to other IBM Informix servers.

The **archecker** utility is designed to recover specific tables or sets of tables; therefore, use ON-Bar or **ontape** in the following data recovery scenarios:

- Full system restore
- Recovery from disk failure

To configure the behavior of the **archecker** utility, use the **archecker** configuration file. To define the schema of the data that **archecker** recovers, use the **archecker** schema command file. These files are described in the following sections.

Note: You cannot use a shared memory connection when performing a table-level restore.

Configuration File

The **archecker** utility uses a configuration file to set certain parameters. Set the **AC_CONFIG** environment variable to the full path name of the **archecker** configuration file. By default, the **AC_CONFIG** environment variable is set to **$INFORMIXDIR/etc/ac_config.std**. If you set **AC_CONFIG** to a user-defined file, you must specify the entire path including the filename.

For information on the configuration parameters used in this file, see "The archecker Configuration Parameter Reference" on page 17-16 later in this chapter.

Schema Command File

The **archecker** utility uses a schema command file to specify the following:

- Source tables
- Destination tables
- Table schemas
- Databases
- External tables
- Point in time the table is restored to
- Other options

This file uses an SQL-like language to provide information **archecker** uses to perform data recovery. For complete information on the supported statements and syntax, see "The archecker Schema Reference" on page 17-9.

There are two methods to set the schema command file:

- Set the AC_SCHEMA configuration parameter in the **archecker** configuration file. For more information, see "AC_SCHEMA" on page 17-18.
- Use the **-f** *cmdname* command line option. For more information, see "Syntax" on page 17-5.

If both methods are specified, the **-f** command line option takes precedence.

Table-Level Restore and Locales

For table-level restore, if the table being restored (table on the archive) has a locale code set different from the default locale (en_US.8859-1) the DB_LOCALE environment variable must be set to have the same code set as the locale of the archived table being restored.

No code set conversion is performed during a table-level restore; the locale code set of the database or table being restored must match the locale code set of the database or table that the data is being restored to. In addition, the same DB_LOCALE information will be used for all of the tables being restored using the same table-level restore command schema file.

Data Restore with archecker

There are two types of restores that **archecker** performs:

- A physical restore which is based on a level-0 archive.

- A physical restore followed by a logical restore which uses both a level-0 archive and logical logs to restore data to a specific point in time.

When reading the command file, **archecker** determines whether to perform a physical restore only or a physical restore followed by a logical restore. By default, **archecker** performs a physical and logical restore. If you use the WITH NO LOG clause, **archecker** does not perform a logical restore.

The procedures and resources that **archecker** uses differ between a physical-only restore and a physical and logical restore. These procedures are outlined in the following sections.

Physical Restore

In a physical restore, data is extracted from a level-0 archive. When performing a physical restore, **archecker** performs the following tasks:

- Disables all constraints (including foreign constraints that reference the target table), indexes, and triggers until the data is restored. Restore performance is better if the table has no constraints, indexes, or triggers.
- Reads the schema command file to determine the following:
  - The source tables
  - The destination tables
  - The schema of all tables
  - The dbspace names of where tables are located
  - The specific archive to extract data from
- Scans the archive for pages belonging to the tables being restored
- Processes each row from the data page and determines if the row is complete or partial.

  If the row is a partial row, then **archecker** determines if the remaining portion of the row has been staged, and if not, it stages the row for later processing.
- For a physical-only restore, applies filters to the row and rejects rows that are not required.
- Inserts the row into the destination table.

Logical Restore

Following the physical restore, logical recovery can further restore tables to a user-specified point in time. To do this, the **archecker** utility reads backed-up logical logs, converts them to SQL statements, and then replays these statements to restore data. Before performing a logical recovery, ensure that all transactions you want to restore are contained in backed-up logical logs. The **archecker** utility cannot replay transactions from the current log. You cannot perform a logical restore on an external table.

If a table is altered, dropped, or truncated during a logical restore, the restore terminates for that table. Termination occurs at the point that the alter was performed. A message in the **archecker** message log file records that an alter operation occurred.

When performing a logical restore, **archecker** uses two processes that run simultaneously:

- Stager: assembles the logical logs and saves them in tables.
- Applier: converts the log records to SQL statements and executes the statements.

The Stager

To collect the pertinent logical log records, the stager performs the following steps:

1. Scans only the backed-up logical logs.

   The stager reads the backed-up logical log files and assembles complete log records.
2. Tests the logical log records

   Any log record that is not applicable to the tables being restored is rejected.
3. Inserts the logical log information into a table

   If the logical log record is not rejected, it is inserted into a stage table.

The Applier

The applier reads data from the control table created by the stager. It begins processing the required transaction and updates the control table to show that this transaction is in process. Next, it operates on each successive log record, row by row, until the transaction commits.

All updates to the control table occur in the same transaction as the log record modification. This allows all work to be completed or undone as a single unit, maintaining integrity at all times. If an error occurs, the transaction is rolled back and the error is recorded in the control table entry for this transaction.

Note: When data is being restored and the DBA has elected to include a logical restore, two additional work columns and an index are added to the destination table. These columns contain the original rowid and original part number. These columns provide a unique key which identifies the location of the row on the original source archive. To control the storage of the index, use the SET WORKSPACE command (see "The SET Statement" on page 17-13). Otherwise, the index is stored in the same space as the table.

After the applier has finished and the restore is complete, these columns, and any indexes created on them, are dropped from the destination table.

Restoring Data with archecker

The **archecker** utility provides a command line interface for restoring data from an archive. To use **archecker**, you must specify both a configuration file and a schema command file. Both of these are described later in this section.

Syntax

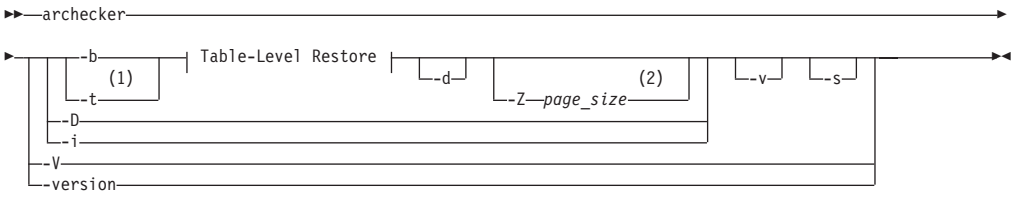
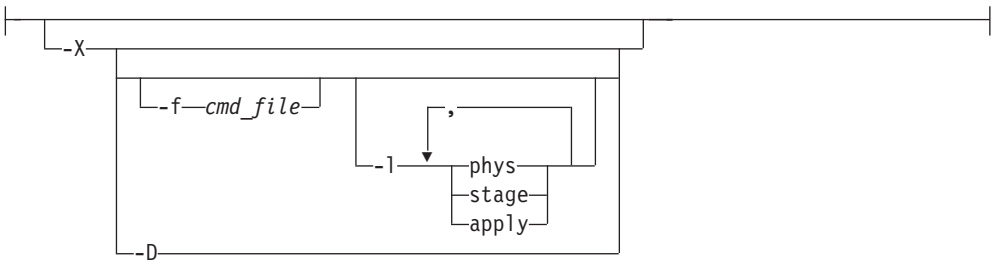


Table-Level Restore:



Notes:

1 Dynamic Server Only

2 Extended Parallel Server Only

Element	Description
-b	Provides direct XBSA access.
-d	Deletes previous **archecker** restore files, except the **archecker** message log. For more information, see "When to Delete Restore Files" on page 17-9.
-D	Deletes previous **archecker** restore files, except the **archecker** message log, and then exits. The **-D** option can be used with the **-X** option to delete previous restore files plus any table-level-restore working tables in the **sysutils** database. For more information, see "When to Delete Restore Files" on page 17-9.
-f *cmdfile*	Specifies that **archecker** use the command file specified by *cmdfile*. This option overrides the value of the AC_SCHEMA configuration parameter. For more information, see "Schema Command File" on page 17-3.
-i	Manually initializes the system.
-lphys,stage,apply	Specifies the level of logical restore: • **phys**: Starts a logical restore of the system, but stops after physical recovery is complete. The backed up logical logs must be available. • **stage**: After physical recovery is complete, extracts the logical logs from the storage manager and stages them in their corresponding tables, and starts the stager. • **apply**: Starts the applier. The applier takes the transactions stored in the stage tables and converts them to SQL and replays the operations. The default level of logical restore if **-l** is not listed is **-lphys,stage,apply**. You can specify any combination of the logical restore levels, separated with commas. Spaces are not allowed between **-l** and levels. For more information, see "Manually Controlling a Logical Restore" on page 17-7.
-s	Prints a status message to the screen.
-t	Specifies **ontape** as the backup utility for Dynamic Server only.
-v	Specifies verbose mode.
-V	Displays IDS version information.

Element	Description
-version	Displays additional version information on the build operation system, build number, and build date for IDS.
-X	Specifies a table-level restore.
-Z *page_size*	Specifies the page size used by the database server: 2, 4, or 8. This option overrides the value of the AC_PAGESIZE configuration parameter. For more information, see "AC_PAGESIZE (XPS)" on page 17-18.

Examples

The following sections illustrate how to use **archecker** commands.

Manually Controlling a Logical Restore

You can manually control the stager and applier using the **-l** command line option. For Extended Parallel Server, manual control is required when you have multiple storage managers that contain logical logs.

The following examples show how to perform a logical restore. In all examples, the name of the schema command file is **cmdfile**.

The following example is a typical usage:

```
archecker -bvs -f cmdfile
```

This command is equivalent to the following command:

```
archecker -bvs -f cmdfile -lphys,stage,apply
```

After the physical restore is complete, the **archecker** utility starts the stager. After the stager has started, the applier is automatically started.

In the following example, the **-lphys** option performs a physical-only restore:

```
archecker -bvs -f cmdfile -lphys
```

In the following example, the **-lstage** option starts the **archecker** stager. The stager extracts the logical log records from the storage manager and saves the applicable records to a table.

```
archecker -bvs -f cmdfile -lstage
```

The stager should only be started after physical recovery has completed.

In the following example, the **-lapply** option starts the **archecker** applier. It looks in the **acu_control** table for the transaction to recover. The applier should only be started after the stager has been started.

```
archecker -bvs -f cmdfile -lapply
```

Performing a Restore with Multiple Storage Managers (XPS)

If you use multiple storage managers, you can perform a table-level restore with **archecker** by configuring **archecker** on every node.

To perform a table-level restore that involves multiple storage managers:

1. Create an **archecker** configuration file on every node.
2. Create a schema command file on every node.
3. Remove old restores by executing the following command on a single node:

```
archecker -DX
```

4. Start the physical restore by executing the following command on each node:

   ```
   archecker -bX -lphys
   ```

   Do not use the **-d** option.
5. After the physical restore completes, start the logical restore by executing the following command on each node that contains logical log records:

   ```
   archecker -bX -lstage
   ```

   Do not use the **-d** option.
6. After starting all stagers, complete the restore by executing the following command on a single node:

   ```
   archecker -bX -lapply
   ```

Performing a Parallel Restore

If you have a fragmented table that resides in separate dbspaces, you can perform a physical table-level restore in parallel by executing multiple **archecker** commands with different schema command files for each dbspace.

During a level-0 archive, there cannot be any open transactions that would change the schema of the table.The table or table fragments being recovered must exist in the level-0 archive. The table or fragment cannot be created or added during the logical recovery. Tables created or fragments added during the logical recovery are ignored.

Because a detached fragment is no longer part of the original table, the applier does not process the detached fragment log record or any other log records for this fragment from this point forward. A message in the **archecker** message log file indicates a detach occurred.

In this example, the table is fragmented across three dbspaces. The corresponding schema command files are named **cmdfile1**, **cmdfile2**, **cmdfile3**. The following commands delete previous restores and then perform physical restores on each dbspace in parallel:

```
archecker -DX
archecker -bvs -f cmdfile1 -lphys
archecker -bvs -f cmdfile2 -lphys
archecker -bvs -f cmdfile3 -lphys
```

You cannot perform a logical restore in parallel.

Restoring Transformed Data

If you are restoring data that was transformed using the BACKUP_FILTER, you must restore the data with the AC_RESTORE_FILTER specified in the **onconfig** file. This parameter is similar to the RESTORE_FILTER parameter specified in **onconfig** file for ontape and ON-Bar restore.

In the **onconfig** file, you would specify:

```
AC_RESTORE_FILTER   path_name options
```

where *path_name* is the path for the filter program and *options* are the program options for the filter. See “AC_RESTORE_FILTER” on page 17-18 for more information.

To learn more about BACKUP_FILTER, see “BACKUP_FILTER” on page 10-6.

When to Delete Restore Files

If you repeatedly run the same archecker table-level restore, you must clean up the archecker table-level restore working files and tables from the previous runs. These working tables refer to acu_ tables in the sysutils database that are created during an archecker table-level restore. The archecker table-level restore working files and tables are kept after an archecker table-level restore completes in case these files and tables are needed for diagnosing problems.

You can remove the working files and tables by explicitly running the command `archecker -DX` or by using the `-d` option when you run the next archecker table-level restore command. The -d option indicates that all files and tables from the previous run of archecker table-level restore are removed before the new restore begins.

- ontape example: `archecker -tdvs -f` *`schema_command_file`*
- onbar example: `archecker -bdvs -f` *`schema_command_file`*

The archecker Schema Reference

This section provides a complete description of the command statements supported by the **archecker** schema command file. Use this file to specify the source and destination tables and to define the table schema.

For more information on specifying which command file **archecker** uses, see "Schema Command File" on page 17-3.

The following are statements supported by **archecker**:

- CREATE TABLE
- DATABASE
- INSERT INTO
- RESTORE
- SET

The syntax of these statements is described in the following sections.

The CREATE TABLE Statement

The CREATE TABLE statement describes the schema of the source and target tables. If the target table is external, use the CREATE EXTERNAL TABLE statement described in the section "The CREATE EXTERNAL TABLE Statement" on page 17-10.

Syntax

The syntax of the CREATE TABLE used in the **archecker** schema command file is identical to the corresponding IBM Informix SQL statement. For a description of this syntax, see the *IBM Informix Guide to SQL: Syntax*.

Usage

You must include the schema for the source table in the **archecker** schema command file. This schema must be identical to the schema of the source table at the time the archive was created.

Note: The source table's schema is not validated by **archecker**. Failing to provide an accurate schema will lead to unpredictable results.

The source table cannot be a synonym or view. The schema of the source table only needs the column list and storage options. Other attributes such as extent sizes, lock modes, and so on are ignored. For an ON-Bar archive, **archecker** uses the list of storage spaces for the source table to create its list of objects to retrieve from the storage manager. If the source table is fragmented, you must list all dbspaces that contain data for the source table. The **archecker** utility only extracts data from the dbspaces listed in the schema command file.

If the source table contains constraints, indexes, or triggers, they are automatically disabled during the restore. Foreign constraints that reference the target table are also disabled. After the restore is complete, the constraints, indexes, and triggers are enabled. For better performance, remove constraints, indexes, and triggers prior to performing a restore.

You must also include the schema of the target table in the command file. If the target table does not exist at the time the restore is performed, it is created using the schema provided.

If the target table already exists, its schema must match the schema specified in the command file. Data is then appended to the existing table.

Examples

The schema of the source and target tables do not have to be identical. The following example shows how you can repartition the source data after performing the data extraction:

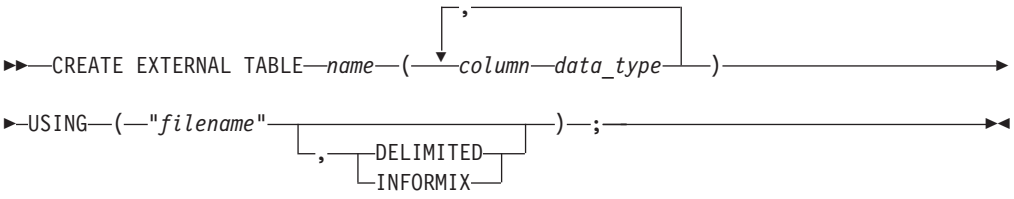
```
CREATE TABLE source (col1 integer, ...) IN dbspace1;
CREATE TABLE target (col1 integer, ...)
   FRAGMENT BY EXPRESSION
      MOD(col1, 3) = 0 in dbspace3,
      MOD(col1, 3) = 1 in dbspace4,
      MOD(col1, 3) = 2 in dbspace5;
INSERT INTO target SELECT * FROM source;
```

The CREATE EXTERNAL TABLE Statement

The CREATE EXTERNAL TABLE statement describes the schema of an external target table.

Syntax

The syntax of the CREATE EXTERNAL TABLE statement for the **archecker** schema file is not identical to the SQL CREATE EXTERNAL TABLE statement.



Element	Description
column	The name of the column. Must conform to SQL identifier syntax rules. For more information, see the *IBM Informix Guide to SQL: Syntax*.

Element	Description
data_type	The built-in data type of the column. For more information about data types, see the *IBM Informix Guide to SQL: Reference*.
filename	Either the name of the file in which to place the data or a pipe device. The pipe device must exist prior to starting the **archecker** utility.
name	The name of the table to store the external data. Must be unique among names of tables, views, and synonyms in the current database. Must conform to SQL database object name rules. For more information, see the *IBM Informix Guide to SQL: Syntax*.

Usage

When you use the CREATE EXTERNAL TABLE statement to send data to an external table, the data is only extracted from a level-0 archive. Logical logs are not rolled forward on an external table.

You can specify either of the following formats for external files:

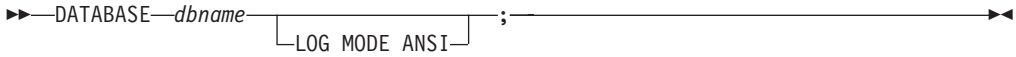
- DELIMITED: ASCII delimited file. This is the default format.
- INFORMIX: internal binary representation. To optimize performance, filters are not applied to external tables. If filters exist, a warning indicates that they will be ignored.

For an example of using the CREATE EXTERNAL TABLE statement, see "Restoring to an External Table" on page 17-15.

The DATABASE Statement

The DATABASE statement sets the current database.

Syntax



Element	Description
dbname	The name of the current database.

Usage

Multiple DATABASE statements can be used. All table names referenced following this statement are associated with the current database.

If the logging mode of the source database is ANSI and default decimal columns are used in the table schemas, then the logging mode of the database must be declared.

Note: If the logging mode of the source database is not declared no error will be returned, but unexpected results and data can occur.

Examples

In the following example, both the source and target tables reside in the same database **dbs**.

```
DATABASE dbs;
CREATE TABLE source (...);
CREATE TABLE target (...);
INSERT INTO target SELECT * from source;
```

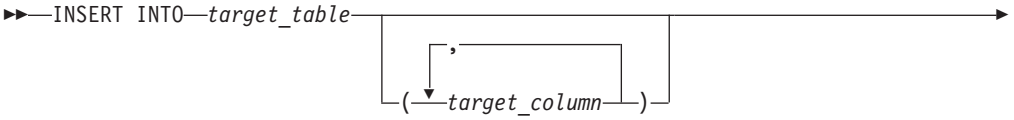
You can use multiple database statements to extract a table from one database into another database.

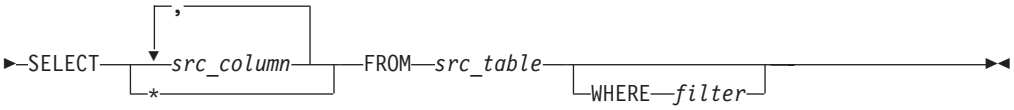
```
DATABASE dbs1;
CREATE TABLE source (...) IN dbspace1;
DATABASE dbs2;
CREATE TABLE target (...) IN dbspace2;
INSERT INTO dbs2:target SELECT * FROM dbs1:source;
```

The INSERT Statement

The INSERT statement tells the **archecker** utility what tables to extract and where to place the extracted data.

Syntax





Element	Description
filter	The following filters are supported by the INSERT statement: • =, !=, <> • >, >=, <, <= • [NOT] MATCH, [NOT] LIKE • IS [NOT] NULL • AND, OR • TODAY, CURRENT The following operators are not supported by the **archecker** utility: • Aggregates • Functions and procedures • Subscripts • Subqueries • Views • Joins Filters can only be applied to physical-only restore.
src_column	A list of columns to be extracted.
src_table	The source table on the archive where the data is restored from.
target_column	The destination column or columns where the data will be restored.
target_table	The destination table where the data will be restored.

Examples

The following example demonstrates the simplest form of the INSERT statement. This statement extracts all rows and columns from the source to the target table.

```
INSERT INTO target SELECT * FROM source;
```

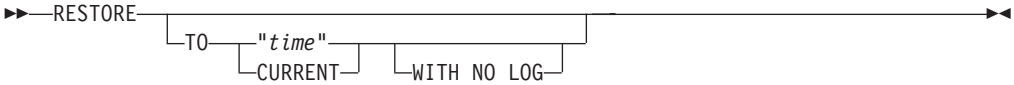
You can also extract a subset of columns. In the following example, only two columns from the source table are inserted into the destination table.

```
CREATE TABLE source (col1 integer, col2 integer, col3 integer, col4 integer);
CREATE TABLE target (col1 integer, col2 integer);
INSERT INTO target (col1, col2) SELECT (col3, col4) FROM source;
```

The RESTORE Statement

This is an optional command used to specify a single point in time the tables specified in the command file should be restored to.

Syntax



Element	Description
"time"	The date and time the table is to be restored to.

Usage

The TO clause is used to restore the table to a specific point in time, which is specified by a date and time or the reserved word CURRENT.

Only one RESTORE statement can be specified in a command file. If this statement is not present in the command file, then the system will be restored to the most current time using logical logs.

If the WITH NO LOG clause is present, only a physical restore is performed. In addition, the two extra columns and the index are not added to the destination table. Physical-only restores are based on level-0 archives only.

Tip: Use this option when you do not have logical logs. You will not receive any messages about logical recovery.

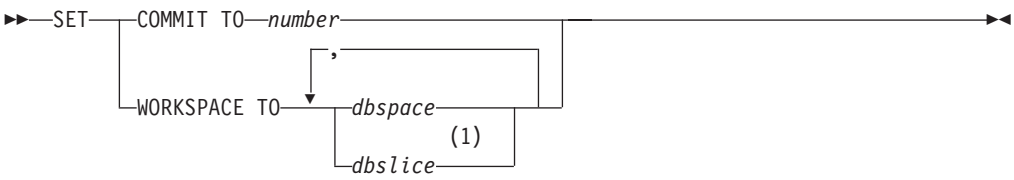
Example

```
RESTORE TO CURRENT WITH NO LOG;
```

The SET Statement

The SET statement controls the different features in the table-level unload library.

Syntax



Notes:

1 Extended Parallel Server Only

Element	Description
number	Sets the number of records to insert before committing during a physical restore. The default is 1000.
dbspace	The dbspaces to use for the working storage space. The default is the root dbspace. You cannot use temporary dbspaces for the working storage space.
dbslice	The dbslices to use for the working storage space.

The **archecker** utility creates several tables for the staging of logical log records during a logical restore. These tables are created in the sysutils database and stored in the working storage space.

Examples

```
SET COMMIT TO 20000;
SET WORKSPACE to dbspace1;
```

SQL Comments

Standard SQL comments are allowed in the **archecker** utility file and are ignored during processing.

Schema Command File Examples

The following examples show different command file syntax for different data recovery scenarios.

Simple Schema Command File

The schema command file in this example extracts a table from the most recent level-0 backup of **dbspace1**. The data is placed in the **table test1:tlr** and the logs are applied to bring the table **tlr** to the current point in time.

```
database test1;
create table tlr (
   a_serial serial,
   b_integer integer,
   c_char char,
   d_decimal decimal
   ) in dbspace1;
insert into tlr select * from tlr;
```

Restoring a Table From a Previous Backup

The schema command file in this example extracts a table from the level-0 backup of **dbspace1**. The logical logs are used to bring the table to the time of ″2003-01-01 01:01:01″. The data is placed in the table **test1:tlr**.

```
database test1;
create table tlr (
   a_serial serial,
   b_integer integer,
   c_char char,
   d_decimal decimal
   ) in dbspace1;
insert into tlr select * from tlr;
restore to '2003-01-01 01:01:01';
```

Restoring to a Different Table

The schema command file in this example extracts a table called test1:tlr from the most recent backup of dbspace1 and places the data in the table test1:tlr_dest.

```
database test1;
create table tlr    (
   a_serial   serial,
   b_integer  integer,
   c_char     char(20),
   d_decimal  decimal,
   ) in dbspace1;
create table tlr_dest    (
   a_serial   serial,
   b_integer  integer,
   c_char     char(20),
   d_decimal  decimal
   ) in dbspace2;
insert into tlr_dest select * from tlr;
```

Extracting a Subset of Columns

The schema command file in this example extracts a table **test1:tlr** from the most recent backup of **dbspace1** and places a subset of the data into the table **test1:new_dest**

```
database test1;
create table tlr    (
   a_serial   serial,
   b_integer  integer,
   c_char     char(20),
   d_decimal  decimal
   ) in dbspace1;
create table new_dest (
   X_char     char(20),
   Y_decimal  decimal,
   Z_name     char(40)
   ) in dbspace2;
insert into new_dest (X_char, Y_decimal) select c_char,d_decimal from tlr;
```

Using Data Filtering

The schema command file in this example extracts a table **test1:tlr** from the most recent backup of **dbspace1** and places the data in the table **test1:tlr** only where the list conditions are true.

Note: Filters can only be applied to a physical restore.

```
database test1;
create table tlr (
   a_serial   serial,
   b_integer  integer,
   c_char     char(20),
   d_decimal  decimal,
   ) in dbspace1;
insert into tlr
  select * from tlr
  where c_char matches 'john*'
  and d_decimal is NOT NULL
  and b_integer > 100;
restore to current with no log;
```

Restoring to an External Table

The schema command file in this example extracts a table called **test1:tlr** from the most recent backup of **dbspace1** and places the data in a file called **/tmp/tlr.unl**.

```
database test1;
create table tlr
(a_serial serial,
```

```
  b_integer integer
) in dbspace1;
create external table tlr_dest
  (a serial serial,
   b_integer integer
  ) using ("/tmp/tlr.unl", delimited );
insert into tlr_dest select * from tlr;
restore to current with no log;
```

Restoring Multiple Tables

The schema command file in this example extracts a table **test1:tlr_1** and **test1:tlr_2** from the most recent backup of **dbspace1** and places the data in **test1:tlr_1_dest** and **test1:tlr_2_dest**. This is an efficient way of restoring multiple tables because it requires only one scan of the archive and logical log files.

```
database test1;
create table tlr_1
  (  columns  ) in dbspace1;
create table tlr_1_dest (  columns  );
create table tlr_2
  (  columns  ) in dbspace1;
create table tlr_2_dest (  columns  );
insert into tlr_1_dest select * from tlr_1;
insert into tlr_2_dest select * from tlr_2;
```

Performing a Distributed Restore

The schema command file in this example extracts a table **test:tlr_1** from the most recent backup of **dbspace1** and places the data on the database server **rem_srv** in the table **rem_dbs:tlr_1**.

```
database rem_dbs
create table tlr_1
   ( columns  );
database test1;
create table tlr_1
  (  columns  ) in dbspace1;
insert into rem_dbs@rem_srv.tlr_1
  select * from tlr_1;
```

The archecker Configuration Parameter Reference

The following configuration parameters are valid in the **archecker** configuration file.

Configuration Parameter	Purpose
AC_DEBUG	Prints debugging messages in the **archecker** message log
AC_IXBAR	Specifies the path name to the IXBAR file
AC_LTAPEBLOCK	Specifies the **ontape** block size for reading logical logs.
AC_LTAPEDEV (IDS)	Specifies the local device name used by **ontape for reading logical logs.**
AC_MSGPATH	Specifies the location of the **archecker** message log
AC_PAGESIZE (XPS)	Specifies the online page size used by the server (can be overridden by the **-Z** command line parameter)
AC_SCHEMA	Specifies the path name to the **archecker** schema command file
AC_STORAGE	Specifies the location of the temporary files that **archecker** builds
AC_TAPEBLOCK	Specifies the tape block size in kilobytes.
AC_TAPEDEV (IDS)	Specifies the local device name used by the **ontape utility**

Configuration Parameter	Purpose
AC_VERBOSE	Specifies either verbose or terse mode for **archecker** messages
BAR_BSALIB_PATH	Identical to the BAR_BSALIB_PATH server configuration parameter. For more information, see "BAR_BSALIB_PATH" on page 10-8.

Dynamic Server

If you do not set the following **archecker** configuration parameters in the **ac_config** file, the default values for their corresponding ONCONFIG configuration parameters are used:

- AC_IXBAR
- AC_TAPEDEV
- AC_TAPEBLOCK
- AC_LTAPEDEV
- AC_LTAPEBLOCK

If the ONCONFIG environment variable is not set, then the default values listed in this section are used.

End of Dynamic Server

AC_DEBUG

Default value Off

Range 1-16

The AC_DEBUG configuration parameter causes debugging messages to be printed in the **archecker** message file. Use this parameter only as directed by technical support. The use of this configuration parameter can cause the **archecker** message log file to grow very large and substantially slow down **archecker** processing.

AC_IXBAR

Default value None

Range Any valid path name

The AC_IXBAR configuration parameter specifies the location of the IXBAR file (for IBM Informix Dynamic Server) or the BIXBAR file (for IBM Informix Extended Parallel Server).

AC_LTAPEBLOCK

Default value 32 kilobytes

Range 0 - 2,000,000,000

The AC_LTAPEBLOCK configuration parameter specifies the size of the tape block in kilobytes when an archive is performed using the following commands:

- **onbar -b**

  When you use ON-Bar, the value of AC_TAPEBLOCK should be the value the BAR_XFER_BUFSIZE (XPS) or BAR_XFER_BUF_SIZE (IDS) configuration

parameter multiplied by the current page size. For more information, see "BAR_XFER_BUFSIZE (XPS)" on page 10-18 or "BAR_XFER_BUF_SIZE (IDS)" on page 10-17.

Dynamic Server

- **ontape -t**

  When you use **ontape**, the value of AC_LTAPEBLOCK should be the value that the TAPEBLK ONCONFIG configuration parameter was set to at the time of the archive. For more information, see "Specifying the Tape-Block-Size Parameters" on page 13-5.

End of Dynamic Server

AC_LTAPEDEV (IDS)

Default value None

Range Any valid path name or STDIO

The AC_LTAPEDEV configuration parameter specifies the device name used by the **ontape** utility. If the tape device is set to STDIO, **archecker receives input from standard input.**

AC_MSGPATH

Default value /tmp/ac_msg.log

Range Any valid path name

The AC_MSGPATH configuration parameter specifies the location of the **archecker** message log (**ac_msg.log**). You must specify the entire path of the message log in the AC_CONFIG file.

AC_PAGESIZE (XPS)

Default value 4

Range 2, 4, 8

The AC_PAGESIZE configuration parameter specifies the online page size used by the server. This configuration parameter can be overridden using the **-Z** command line option.

AC_RESTORE_FILTER

Default value none

Range Full path name and options for the filter program.

The AC_RESTORE_FILTER configuration parameter specifies a path and options for a filter program that restores transformed data to its original state prior to backup. It is required when the data to be restored was transformed, such as compressed or encrypted, with the BACKUP_FILTER parameter in **onconfig**.

AC_SCHEMA

Default value None

Range Any valid path name

The AC_SCHEMA configuration parameter specifies the path name to the **archecker** schema command file. This configuration parameter is overridden by the **-f** *cmdfile* command line option.

AC_STORAGE

Default value /tmp

Range Any valid directory path name

The AC_STORAGE parameter specifies the location of the directory where **archecker** stores its temporary files. You must specify the entire path of the storage directory in the AC_CONFIG file.

AC_TAPEBLOCK

Default value 32 kilobytes

Range 0 - 2,000,000,000

The AC_TAPEBLOCK configuration parameter specifies the size of the tape block in kilobytes when an archive is performed using the following commands:

- **onbar -b**

  When you use ON-Bar, the value of AC_TAPEBLOCK should be the value the BAR_XFER_BUFSIZE (XPS) or BAR_XFER_BUF_SIZE (IDS) configuration parameter multiplied by the current page size. For more information, see "BAR_XFER_BUFSIZE (XPS)" on page 10-18 or "BAR_XFER_BUF_SIZE (IDS)" on page 10-17.

Dynamic Server

- **ontape -t**

  When you use **ontape**, the value of AC_TAPEBLOCK should be the value that the TAPEBLK ONCONFIG configuration parameter was set to at the time of the archive. For more information, see "Specifying the Tape-Block-Size Parameters" on page 13-5.

End of Dynamic Server

AC_TAPEDEV (IDS)

Default value None

Range Any valid path name or STDIO

The AC_TAPEDEV configuration parameter specifies the device name used by the **ontape** utility. If the tape device is set to STDIO, **archecker receives input from standard input.**

AC_VERBOSE

Default value 1

Range

1	verbose messages

0	terse messages

The AC_VERBOSE configuration parameter specifies either verbose or terse output in the **archecker** message log and to the screen.

Part 5. Appendixes

© Copyright IBM Corp. 1996, 2008

Appendix A. Troubleshooting

This appendix lists some error messages you can receive during a backup or restore, describes under what circumstances the errors might occur, and provides possible solutions or workarounds.

Corrupt Page During an Archive

The message `Archive detects that page is corrupt` indicates that page validation failed.

During an archive, the database server validates every page before writing it to the archive device. This validation checks that the elements on the page are consistent with the expected values. When a page fails this validation, a message similar to the following is written to the **online.log** file:

```
16:27:49 Assert Warning: Archive detects that page 1:10164 is corrupt.
16:27:49 IBM Informix Dynamic Server Version 10.00.FC1
16:27:49 Who: Session(5, informix@cronus, 23467, 10a921048)
Thread(40, arcbackup1, 10a8e8ae8, 1)
File: rsarcbu.c Line: 2915
16:27:49 stack trace for pid 23358 written to /tmp/af.41043f4
16:27:49 See Also: /tmp/af.41043f4
16:27:49 Archive detects that page 1:10164 is corrupt.
16:27:50 Archive on rootdbs Completed with 1 corrupted pages detected.
```

The archive aborts after detecting 10 corrupt pages. The **online.log** file displays the full error message, including the page address, for the first 10 errors. Subsequently, only the count of the number of corrupt pages is put in to the **online.log**.

After you receive this message, identify which table the corrupt page belongs to by examining the output of the **oncheck –pe** command. To determine the extent of the corruption, execute the **oncheck –cID** command for that table.

A corrupt page is saved onto the backup media. During a restore, the corrupt page is returned in its corrupt form. No errors messages are written to the **online.log** when corrupt pages are restored, only when they are archived.

Log Backup Already Running

When using ON-Bar to create a backup, the message `log backup is already running` in the **bar_act.log** file, or the message `Process exited with return code 152` in the **online.log** file do not indicate a problem. They can appear under the following circumstances:

- The ALARMPROGRAM configuration parameter is set to **log_full.sh**. Periodically, events cause **log_full.sh** to trigger the **onbar -b -l** command. If a log fills while the **onbar -b -l** command is running, then ON-Bar backs up that log as well. If the backup has not completed by the time of the next event trigger, it generates a warning in the **bar_act.log** file. At the time of the next event trigger, the log backup can continue.
- A level-0 archive (especially when started with the **-w** option) first archives the database and then automatically start the **onbar -b -l** command to back up any

© Copyright IBM Corp. 1996, 2008

logical logs that are currently full and not yet backed up. There might not be a **log_full.sh** message in **online.log**, because the **onbar -b -l** command is started directly.

- When you mount a new tape after filling a previous tape, a **log_full.sh** event is scheduled but not triggered. As soon as the next log fills and generates an event trigger in the **log_full.sh** file, all available logs are archived. You can force the archive by running **onbar -b -l** or force **log_full.sh** to be triggered by running **onmode -l**.

No Server Connection During a Restore

During a whole system restore with ON-Bar, the error `archive api error: no server connection.` might appear in the **bar_act.log** file. ON-Bar then connects to the storage manager successfully, but eventually fails with the error `archive api error: not yet open`.

The **bar_act.log** file contains information similar to the following messages:

```
2000-03-09 10:51:06 19304  19303 /usr/informix/bin/onbar_d -r -w
2000-03-09 10:51:09 19304  19303 ERROR: Unable to start the physical restore:
Archive API error: no server connection.
2000-03-09 10:51:09 19304  19303 Successfully connected to Storage Manager.
2000-03-09 10:51:36 19304  19303 Process 19304 received signal 3. Process will
exit after cleanup.
2000-03-09 10:59:13 19811  19810 /usr/informix/bin/onbar_d -r -w
2000-03-09 10:59:16 19811  19810 ERROR: Unable to start the physical restore:
Archive API error: no server connection.
2000-03-09 10:59:16 19811  19810 Successfully connected to Storage Manager.
2000-03-09 11:01:12 19811  19810 Begin cold level 0 restore llog1.
2000-03-09 11:01:12 19811  19810 ERROR: Unable to write restore data to the
database server: Archive API error: not yet open.
```

To solve this problem, check if the database server is still running. If it is, shut down the database server and run the command again.

Dropping a Database Before a Restore

If you perform a level-0 archive using ON-Bar and a storage manager, then drop a database, and then perform a restore with the **onbar -r** command, the database remains dropped. The restore salvages the logs and the logs contains the DROP DATABASE statement. When the logs are salvaged, or replayed, the database is dropped.

To prevent this situation, perform a physical restore using the **onbar -r -p** command, and then a logical restore using the **onbar -r -l** command. This sequence does not salvage the logs and does restore the database.

No dbspaces or blobspaces During a Backup or Restore

If the emergency boot file, **ixbar.***servernum*, does not have the correct entries for objects in the backup, the message `There are no DB/BLOBspaces to backup/restore` appears in **bar_act.log** file during a restore started with the **onbar -r** or **onbar -r -w** command.

This error can appear under the following circumstances:

- During an external restore, if the emergency boot file was not copied from the source system.

- If the emergency boot file was recreated after the archive backup was made. The previous file is saved in the form: **ixbar.xx.xxxx**.
- An attempt to execute the **onbar -r -w** command with a backup that is not a full system backup.

Restoring blobspace BLOBs

You can use table-level restore to restore a BLOB that is stored in a table. However, restoring a BLOB that is stored in a blobspace is not supported. If you attempt to restore a blobspace BLOB, the column is set to NULL.

Changing the System Time on the Backup System

Time lines use the UNIX time as the archive checkpoint time for dbspaces and the closing time for logical logs. If logs are not automatically backed up and the system clock is changed, the time line can get corrupted.

For example, if you have logical logs that were closed before the archive checkpoint time, they have a timestamp that is higher than the archive checkpoint time. The dbspace does not need the logs and ON-Bar will try to restore the backup immediately. if a log cannot be found, ON-Bar fails with the following message: `There are no storage spaces or logical logs to backup or restore.`

To restore the storage space and logical logs:

1. Change the clock back to its original value.
2. Recover the system from backup.
3. Change the clock back to the new time.

Appendix B. onstat Command Reference

This appendix describes forms of the **onstat** command that are relevant to ON-Bar and **ontape**. The **onstat** utility reads shared-memory structures and provides statistics about the database server that are accurate at the instant that the command executes. For general information about **onstat**, refer to the *IBM Informix Administrator's Reference*.

onstat -d

Use the **-d** option to display information for chunks in each storage space. You can interpret output from this option as follows. The first section of the display describes the storage spaces:

address The address of the storage space in the shared-memory space table

number The unique ID number of the storage space assigned at creation

flags Uses the following hexadecimal values to describe each storage space:

0x00000000	Mirror not allowed and dbspace is unmirrored
0x00000001	Mirror is allowed and dbspace is unmirrored
0x00000002	Mirror is allowed and dbspace is mirrored
0x00000004	Down
0x00000008	Newly mirrored
0x00000010	Blobspace
0x00000020	Blobspace on removable media
0x00000040	Blobspace is on optical media
0x00000080	Blobspace is dropped
0x00000100	Blobspace is the optical STAGEBLOB
0x00000200	Space is being recovered
0x00000400	Space is fully recovered
0x00000800	Logical log is being recovered
0x00001000	Table in dbspace is dropped
0x00002000	Temporary dbspace
0x00004000	Blobspace is being backed up
0x00008000	Sbspace
0x0000a001	Temporary sbspace
0x00010000	Physical or logical log changed
0x00020000	Dbspace or chunk tables have changed

fchunk The ID number of the first chunk

nchunks The number of chunks in the storage space

flags Uses the following letter codes to describe each storage space:

© Copyright IBM Corp. 1996, 2008

Position 1:

M	Mirrored
N	Not Mirrored

Position 2:

R	Being recovered
P	Physically recovered, waiting for P -- logical recovery
L	Being logically recovered
X	Newly mirrored
D	Disabled

Position 3:

B	Blobspace
S	Sbspace
T	Temporary dbspace

Position 4:

B	Has big chunks (IDS)

owner The owner of the storage space

name The name of the storage space

The line immediately following the storage-space list includes the number of active spaces (the current number of dbspaces in the database server instance including the rootdbs) and the number of total spaces.

Active spaces refers to the current number of storage spaces in the database server instance including the rootdbs. **Total** refers to total *allowable* spaces for this database server instance.

The second section of the **onstat -d** output describes the chunks:

address The address of the chunk

chk/dbs The chunk number and the associated space number

offset The offset into the file or raw device in pages

size Is the size of the chunk in pages (in units of the dbspace page size to which the chunk belongs)

free Is the number of free pages in the chunk for a dbspace (in units of the dbspace page size to which the chunk belongs).

For a blobspace, a tilde indicates an approximate number of free blobpages.

For an sbspace, indicates the number of free pages of user data space and total user data space.

bpages The size of the chunk in blobpages

Blobpages can be larger than disk pages; therefore, the **bpages** value can be less than the **size** value.

For an sbspace, is the size of the chunk in sbpages

flags Provides the chunk status information as follows:

Position 1:

P	Primary
M	Mirror

Position 2:

O	Online
D	Down
X	Newly mirrored
I	Inconsistent
R	Being recovered
N	Renamed and either down or inconsistent (IDS)

Position 3:

-	Dbspace
B	Blobspace
S	Sbspace

Position 4:

B	Big chunk (IDS)

pathname The path name of the physical device

The line immediately following the chunk list displays the number of active chunks (including the root chunk) and the total number of chunks.

For information on how to solve problems indicated by flags in position 2 of both the storage space and chunk sections, see "Monitoring Restores" on page 7-7.

onstat -g bus (XPS)

Prints current backup scheduler sessions, backups in progress, and backups to be performed. Issue from any coserver.

onstat -g bus_sm (XPS)

Prints current storage manager configuration. Issue from any coserver.

onstat -l

Use the **-l** option to display information about physical and logical logs. You can interpret output from this option as follows. The first section of the display describes the physical-log configuration:

buffer The number of the physical-log buffer

bufused The number of pages of the physical-log buffer that are used

bufsize The size of each physical-log buffer in pages

numpages The number of pages written to the physical log

numwrits The number of writes to disk

pages/io Is calculated as `numpages/numwrits`

This value indicates how effectively physical-log writes are being buffered.

phybegin The physical page number of the beginning of the log

physize The size of the physical log in pages

phypos The current position in the log where the next log-record write is to occur

phyused The number of pages used in the log

%used The percent of pages used

The second section of the **onstat -l** display describes the logical-log configuration:

buffer The number of the logical-log buffer

bufused The number of pages used in the logical-log buffer

bufsize The size of each logical-log buffer in pages

numrecs The number of records written

numpages The number of pages written

numwrits The number of writes to the logical log

recs/pages Is calculated as `numrecs/numpages`

You cannot affect this value. Different types of operations generate different types (and sizes) of records.

pages/io is calculated as `numpages/numwrits`

You can affect this value by changing the size of the logical-log buffer (specified as LOGBUFF in the ONCONFIG file) or by changing the logging mode of the database (from buffered to unbuffered, or vice versa).

The following fields are repeated for each logical-log file:

address The address of the log-file descriptor

number Is logid number for the logical-log file

The logid numbers might be out of sequence because either the database server or administrator can insert a log file in-line.

flags Provides the status of each log as follows:

A Newly added (and ready to use)

B Backed up

C Current logical-log file

D Marked for deletion

To drop the log file and free its space for reuse, you must perform a level-0 backup of all storage spaces

F Free, available for use

L The most recent checkpoint record

U Used

uniqid	The unique ID number of the log
begin	The beginning page of the log file
size	The size of the log in pages
used	The number of pages used
%used	The percent of pages used
active	The number of active logical logs
total	The total number of logical logs

The database server uses *temporary logical logs* during a warm restore because the permanent logs are not available then. The following fields are repeated for each temporary logical-log file:

address	The address of the log-file descriptor
number	Is logid number for the logical-log file
flags	Provides the status of each log as follows: B Backed up C Current logical-log file F Free, available for use U Used
uniqid	The unique ID number of the log
begin	The beginning page of the log file
size	The size of the log in pages
used	The number of pages used
%used	The percent of pages used
active	The number of active temporary logical logs

Appendix C. Migrating Data, Servers, and Tools

Using Data-Migration Tools for Recovery

If ON–Bar and **ontape** are not working, you can use data-migration utilities, such as **onunload**, the High-Performance Loader (HPL), **onpladm**, or **dbexport**, as a substitute for a backup. For more information on these utilities, see the *IBM Informix Migration Guide*.

Important: None of the data-migration utilities are coordinated with the information stored in the logical-log files and, unlike backups, they do not save a copy of system-overhead information important to the database server.

Preparing for a Database Server or Storage-Manager Upgrade

Important: The database server conversion software automatically recreates the **sysutils** database when you upgrade to the latest version of the database server. All backup and restore information from the old database server version is lost. Backups that you make under the older version of the database server are not compatible with the newer version of the database server.

To prepare for an upgrade:

1. Use ON–Bar to perform a level-0 backup of all your data before you upgrade your database server, ISM, or change storage-manager vendors.
2. Save these backups so that you can restore the data in case you need to revert to the old database server version.
3. Before you upgrade, back up the administrative files.
4. After you upgrade the database server, back up all storage spaces and logical logs.

   For more information on database server migration, see the *IBM Informix Migration Guide*.

Upgrading Your Storage Manager

If you install a new version of a third-party storage manager, install it before you bring up the database server. Update the **sm_versions** file with the new storage-manager definition. If you have continuous logical-log backup set up on the database server, ON–Bar can start backing up the logical logs soon after the database server comes online. Also make sure that the new storage-manager version is able to read media written with your old version.

Make sure that the storage manager can find the backup objects that ON–Bar requests. Use the **onsmsync** utility to expire old backup history in the **sysutils**

© Copyright IBM Corp. 1996, 2008

database and emergency boot files.

Extended Parallel Server

In Extended Parallel Server only, ensure that the BAR_SM parameters in the ONCONFIG file match the new storage-manager definition.

End of Extended Parallel Server

Changing Storage-Manager Vendors

When you switch storage-manager vendors, the transition can be difficult. Ensure that the new data formats are identical, that a reversion utility is provided, or that you do not use new features that change the data formats. Differences usually occur in the following areas:

- The new storage manager might support different storage devices. If you also upgrade a storage device, make sure the old storage device is available until you successfully back up and restore on the new storage device.
- If you change physical connectivity, such as moving a storage device from a local connection to a network server, make sure the storage manager can still move the data across the network.
- If you use software compression or encryption, make sure all versions of the compression or encryption algorithms are available for restores.
- Ensure that the storage manager can send multiple data streams to storage devices. It also might use a different version of XBSA.

You can switch between certain storage managers more easily than others. For details, contact Technical Support or your vendor.

Migrating from ontape to ON-Bar (IDS)

You cannot back up data with **ontape** and restore it using ON–Bar, or conversely because the data storage formats and backup capabilities are different. You can use **ontape** with the database server in online or quiescent mode.

To migrate to ON-Bar:

1. Use **ontape** to perform a full backup.

   For details, see Chapter 14, "Backing Up with ontape," on page 14-1.
2. Take the backup media offline to prevent possible reuse or erasure.
3. Configure the storage manager to be used with ON–Bar.

   For details, see Chapter 4, "Configuring the Storage Manager and ON-Bar," on page 4-1.
4. Configure your ON–Bar environment:
   a. Set ONCONFIG parameters.
   b. Create the **sm_versions** file with the storage-manager definition.

      For details, see Chapter 10, "ON-Bar Configuration Parameters," on page 10-1, and "Updating the sm_versions File" on page 4-4.
5. Use ON–Bar (**onbar -b** or **onbar -b -w**) to perform a full backup.
6. Verify the backup with **onbar -v**.

   For details, see Chapter 6, "Verifying Backups," on page 6-1.

Migrating Private ON-Bar Scripts

This section describes the procedures for migrating private ON–Bar scripts after you upgrade the database server version.

If your script searches for ON–Bar process IDs, be aware that the relationship between the **onbar-driver** and the **onbar_d** child processes or **onbar-worker** processes is quite different for Dynamic Server and Extended Parallel Server.

Dynamic Server

If you reuse a private script for Extended Parallel Server on Dynamic Server, remove the following ON–Bar options that Dynamic Server does not support:

- **-q** (session name)
- **-b -p** (physical-only backup)

End of Dynamic Server

Extended Parallel Server

If you reuse a private script for Dynamic Server on Extended Parallel Server, remove the following ON–Bar options that Extended Parallel Server does not support:

- **-w** (whole-system backup)
- **-c** (current log backup)
- **-C** (continuous-log backup)
- **-RESTART** (restartable restore)

End of Extended Parallel Server

Appendix D. GLS Support

Using GLS with ON-Bar

ON–Bar supports Global Language Support (GLS), which allows users to work in their native language. The language that the client application uses is called the *client locale*. The language that the database uses for its server-specific files is called the *server locale*.

ON–Bar must run on the same computer as the database server. However, you can run ON–Bar in any locale for which you have the supporting message and localization files. For example, if the server locale is English and the client locale is French, you can issue ON–Bar commands in French.

The following command performs a level-0 backup of the dbspaces specified in the file, **tombé**:

```
onbar -b -L 0 -f tombé
```

Windows Only

On Windows, you cannot use multibyte filenames in backup or restore commands because they are not supported.

End of Windows Only

The **sysutils** database, the emergency boot files, and the storage-manager boot file are created with the **en_us.8859-1** (default English) locale. The ON–Bar catalog tables in the **sysutils** database are in English. Change the client and database locales to **en_us.8859-1** before you attempt to connect to the **sysutils** database with DB–Access or third-party utilities.

Identifiers That Support Non-ASCII Characters

The *IBM Informix GLS User's Guide* describes the SQL identifiers that support non-ASCII characters. Non-ASCII characters include both 8-bit and multibyte characters. You can use non-ASCII characters in the database names and filenames with the ON–Bar, **ondblog**, and **onutil** commands, and for filenames in the ONCONFIG file.

For example, you can specify a non-ASCII filename for the ON–Bar activity log in BAR_ACT_LOG and a non-ASCII path name for the storage-manager library in BAR_BSALIB_PATH.

Identifiers That Require 7-Bit ASCII Characters

You must use 7-bit ASCII characters for the following identifiers:

- Storage-space names
- Database server names

Locale of ON-Bar Messages

All ON–Bar messages appear in the activity log in the client locale except the messages that the database server issues. For example, the part of the message that

© Copyright IBM Corp. 1996, 2008

tells you that a database server error occurred appears in the client locale, and the server-generated part appears in the server locale.

Using the GL_DATETIME Environment Variable with ON-Bar

The database server must know how to interpret and convert the end-user formats when they appear in date or time data that the client application sends. You can use the **GL_DATE** and **GL_DATETIME** environment variables to specify alternative date and time formats. If you do not set these environment variables, ON–Bar uses the date and time format of the client locale.

If you perform a point-in-time restore, enter the date and time in the format specified in the **GL_DATETIME** environment variable if it is set.

Point-in-Time Restore Example

For example, the default date and time format for the French locale, `fr_fr.8859-1`, uses the format `"%A %.1d %B %iY %H:%M:%S."` The ON–Bar command for a point-in-time restore is as follows:

```
onbar -r -t "Lundi 9 Juin 1997 11:20:14"
```

You can set **GL_DATETIME** to a different date and time format that uses the date, month, two-digit year, hours, minutes, and seconds.

```
%.1d %B %iy %H:%M:%S
```

The ON–Bar command for a point-in-time restore is as follows:

```
onbar -r -t "9 Juin 97 11:20:14"
```

Tip: For more information on how to use GLS and the **GL_DATE** and **GL_DATETIME** environment variables, refer to the *IBM Informix GLS User's Guide*.

Using GLS with ontape

The **ontape** utility supports GLS in the same way as ON–Bar does. You can specify the database name in the national locale.

Appendix E. Accessibility

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

Accessibility features for IBM Informix Dynamic Server

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 Dynamic Server. These features support:

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

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

Keyboard Navigation

This product uses standard Microsoft® Windows navigation keys.

Related Accessibility Information

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

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

IBM and Accessibility

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

Dotted Decimal Syntax Diagrams

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

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

© Copyright IBM Corp. 1996, 2008

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

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

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

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

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

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

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

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines

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

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

Notes:

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

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

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

© Copyright IBM Corp. 1996, 2008

programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, 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, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

Index

Special characters

-Admin group 5-2
/dev/null, ontape
  as a tape device 13-4
  for logical–log backup 14-14

A

AC_CONFIG
  file 10-2
  setting archecker parameters 10-2
AC_CONFIG environment variable 6-7, 10-3, 10-4
AC_CONFIG file environment variable 10-2
ac_config.std file 10-3, 10-4
AC_DEBUG configuration parameter 17-17
AC_IXBAR configuration parameter 17-17
ac_msg.log.
  See archecker message log.
AC_MSGPATH configuration parameter 10-3, 10-22, 17-18
AC_PAGESIZE configuration parameter 17-18
AC_RESTORE_FILTER configuration parameter 17-8, 17-18
AC_SCHEMA configuration parameter 17-18
AC_STORAGE configuration parameter 6-7, 10-3, 17-19
AC_TAPEBLOCK configuration parameter 17-19
AC_TAPEDEV configuration parameter 17-19
AC_TIMEOUT configuration parameter 10-3, 10-4
AC_VERBOSE configuration parameter 10-3, 10-4, 17-19
accessibility E-1
  keyboard E-1
  shortcut keys E-1
Accessibility
  dotted decimal format of syntax diagrams E-1
  syntax diagrams, reading in a screen reader E-1
Activity log, ON-Bar
  checking restore status 7-32
  defined 3-11
  diagnosing failed verification 6-7
  error messages 12-2, 12-5
  finding backup time 7-22
  message format 12-1
  message numbers 12-2
  monitoring backups 5-8
  overview 3-11
  return codes 12-6
  skipped dbspace message 7-14
  specifying location 4-11, 10-6
  specifying performance monitoring 10-13
  specifying performance statistics 4-10
  specifying progress messages 4-10, 4-12, 10-14
  status of archecker 6-2, 10-22
  verification message 6-6
  warning about /dev/null 10-20
Administrative files
  backing up 5-4
  copying before cold restore 7-17, 7-25
  external backup 8-2, 8-8, 8-9, 16-1, 16-2
Administrative tasks
  backing up after changing schema 5-10, 14-3
  using ISA 5-12
ALARM_ALL_EVENTS configuration parameter 4-9
ALARMPROGRAM configuration parameter
  alarmprogram.sh or alarmprogram.bat 10-5
  defined 4-9, 10-5
  setting
    log_full.sh or log_full.bat 5-19
    no_log.sh or no_log.bat 5-19
alarmprogram.sh script 5-20, 10-5
ALRM_ALL_EVENTS configuration parameter 10-5
Altering table type
  light append 7-34
  restoring data 5-11
Applier, manually controlling logical restore using 17-4, 17-5
archecker utility
  AC_CONFIG parameters 10-3
  AC_RESTORE_FILTER configuration parameter 17-8, 17-18
  blobspaces 6-5
  configuration file, valid parameters 17-16
  defined 17-1
  described 17-2
  estimating temporary space 6-4
  fixing verification problems 6-7, 6-8
  interpreting verification messages 6-6
  message log 10-3
  messages 6-6, 10-3
  onsmsync 6-7, 6-8
  point-in-time verification 6-5
  restoring transformed data 17-8, 17-18
  sbspaces 6-6
  sbspaces and blobspaces 6-4
  syntax
    diagram 6-3, 17-5
    elements 17-6
  table-level restore 17-7
  temporary files 6-4, 10-4
  uses 17-2
  using 17-5
  verification failure, causes of 6-6, 6-7
  verifying expired backups 6-7
  whole-system backup 6-5
Autochanger 4-16
Automatic backups.
  See Continuous log backup.

B

Backup Scheduler
  defined 3-6
  how it works 5-26
  monitoring status 9-11
  physical 5-6
  SMI tables overview 11-8
  sysbuobject table 11-9
  sysbuobjses table 11-9
  sysbusession table 11-10
  sysbusm table 11-10
  sysbusmdbspace table 11-10
  sysbusmlog table 11-10
  sysbusmworker table 11-11
  sysbuworker table 11-11

© Copyright IBM Corp. 1996, 2008

BACKUP_FILTER
about 2-4
configuration parameter 4-9, 10-6, 13-2
defining in onconfig 3-12
ON-Bar utility 10-6
ontape utility 13-2
transforming data with 3-12
unique server names 3-12
Backups
advantages 14-6
catalogs
expiring and synchronizing 9-5
sysutils 9-5
directory 14-6
examples 14-8
expiring, on ISM 9-7
filled logical-log files 14-9
oncheck, monitoring history 14-10
ontape utility
creating 14-5
IFX_ONTAPE_FILE_PREFIX environment variable 14-6
logical log filled 14-9
LTAPEDEV configuration parameter 14-6
TAPEBLK configuration parameter 14-6
TAPEDEV configuration parameter 14-6
TAPESIZE configuration parameter 14-6
overview
levels 1-2
planning 2-2, 2-3
schedule 2-4, 3-8
Raw tables 14-9
removing expired 9-7
setting TAPEDEV to STDIO 14-6
standard output
defined 14-6
syntax 14-7
tapes 14-5
Backups, ON-Bar
-O option 3-3, 5-9, 5-16
See also Continuous log backup.
assigning a session name 5-16
automatic log backups 10-5
changing database logging 5-11
checking data consistency 5-8
complete, example of 5-13
current log 3-3, 8-8, 16-3
defined 5-8, 5-17
external 3-3, 8-2, 16-1
fake backups 3-3, 5-16
imported restore 7-29, 7-31
incremental, example of 5-13
ISM catalog 5-11
large chunk files 5-4
list of storage spaces 5-13
logical log 1-2, 3-3, 5-18, 5-23
monitoring progress 3-11
operational tables 5-17
physical-only 3-3, 5-9, 5-17
raw tables 5-17
salvaging logs 1-3, 5-19, 5-24
sbspaces, example of 5-15
scratch tables 5-17
sequence
Dynamic Server 5-24
Extended Parallel Server 5-26
standard backup 3-3
Backups, ON-Bar *(continued)*
standard tables 5-17
static tables 5-17
storage-manager installation 5-5
syntax 5-8
table types 5-11, 5-17
temp tables 5-17
verification 3-3, 6-1, 6-8
whole system 3-3
Backups, ontape
before you create 14-4
interrupted 14-10
labelling the tape 14-5
levels 14-3
monitoring 14-10
one device 13-3
premature termination 14-10
BAR_ACT_LOG configuration parameter 3-11, 4-9, 4-11, 10-6
bar_action table 11-2
BAR_BOOT_DIR configuration parameter 4-11, 10-7
BAR_BSALIB_PATH configuration parameter 4-8, 4-9, 4-11, 10-8
BAR_DBS_COSVR configuration parameter 4-11, 10-9
BAR_DEBUG configuration parameter 3-11, 4-10, 10-9
BAR_DEBUG_LOG configuration parameter 3-11, 4-10, 10-10
BAR_HISTORY configuration parameter 4-10, 4-11, 10-10
BAR_IDLE_TIMEOUT configuration parameter 4-11, 5-27, 10-10
bar_instance table 11-2
bar_ixbar table 11-4
BAR_IXBAR_PATH configuration parameter 4-10, 10-11
BAR_LOG_COSVR configuration parameter 4-11, 10-11
BAR_MAX_BACKUP configuration parameter 4-10, 5-25, 10-12
BAR_NB_XPORT_COUNT configuration parameter 4-10, 10-13
bar_object table 11-6
BAR_PERFORMANCE configuration parameter 4-10, 10-13
BAR_PROGRESS_FREQ configuration parameter 4-10, 4-12, 10-13
BAR_RETRY configuration parameter 4-10, 4-12, 10-14
bar_server table 11-6
BAR_SIZE_FACTOR configuration parameter 10-15
BAR_SM configuration parameter 4-12, 10-16
BAR_SM_NAME configuration parameter 4-12, 10-16
BAR_WORKER_COSVR configuration parameter 4-12, 10-16
BAR_WORKER_MAX configuration parameter 4-12, 4-13, 10-16
BAR_XFER_BUF_SIZE configuration parameter 4-10, 10-17
page size 10-17
BAR_XFER_BUFSIZE configuration parameter
defined 4-12, 10-18
page size 10-19
BAR_XPORT_COUNT configuration parameter
defined 4-12, 10-19
page size 10-19
bargroup group 4-6
BIXBAR file, specifying location of 17-17
Bixbar.hostname.servernum file 3-10
bldutil.process_id file 10-22
Blobspaces
availability for backing up 5-16, 14-11
backing up 1-1
backing up offline 5-16
optical platters 5-4
temp space for archecker 6-4

Block size, ontape
parameter 13-5
tape device 13-6
Blocking, database server 8-2, 16-1, 16-3
Boot file
ixbar 3-10

C

Child process, ON-Bar 5-24
Chunks
creating new chunks 15-11
files
recreating during restore 7-22
renaming during restore 7-26
renaming 15-11
command line options 7-28
during a restore 7-26
examples using ON-Bar 7-28
listfile 15-12
nonexistent device 7-28, 15-12
nonexistent device, procedure 15-12
ON-Bar syntax 7-27
overview with ON-Bar 7-26
overview with ontape 15-10
specifying other options 7-28, 15-12
with a file 7-28, 15-12
Client locale D-1
cloning 8-1, 16-1
Cold restore
See also Restoring.
automatic log salvage 7-17
defined 1-4, 7-2
ON-Bar utility
example 7-16
renaming chunks during 7-26
sequence 7-41, 7-43
setting mode 7-3
ontape utility
defined 15-2
mixed restore 15-3
performing 15-7
renaming chunks during 15-10
procedure, in stages 7-15
Complete restore
onbar -r commands 7-8
procedure 7-8
Components, ON-Bar
Dynamic Server 3-4
Extended Parallel Server 3-5
compressing data, example 3-12
Configuration file
AC_CONFIG 10-3
archecker 17-3
ONCONFIG 10-5
Configuration parameters
AC_MSGPATH 10-3, 10-22
AC_STORAGE 10-3
AC_TIMEOUT 10-3, 10-4
AC_VERBOSE 10-3, 10-4
ALARM_ALL_EVENTS 4-9
ALARMPROGRAM 4-9, 5-19, 10-5
ALRM_ALL_EVENTS 10-5
BACKUP_FILTER 4-9, 10-6
BAR_ACT_LOG 3-11, 4-9, 4-11, 10-6
BAR_BOOT_DIR 4-11, 10-7
BAR_BSALIB_PATH 4-8, 4-9, 4-11, 10-8
Configuration parameters *(continued)*
BAR_DBS_COSVR 4-11, 10-9
BAR_DEBUG 3-11, 4-10, 10-9
BAR_DEBUG_LOG 3-11, 4-10, 10-10
BAR_HISTORY 4-10, 4-11, 10-10
BAR_IDLE_TIMEOUT 4-11, 5-27, 10-10
BAR_IXBAR_PATH 4-10, 10-11
BAR_LOG_COSVR 4-11, 10-11
BAR_MAX_BACKUP 4-10, 5-25, 10-12
BAR_NB_XPORT_COUNT 4-10, 10-13
BAR_PERFORMANCE 4-10, 10-13
BAR_PROGRESS_FREQ 4-10, 4-12, 10-13
BAR_RETRY 4-10, 4-12, 10-14
BAR_SIZE_FACTOR 10-15
BAR_SM 4-12, 10-16
BAR_SM_NAME 4-12, 10-16
BAR_WORKER_COSVR 4-12, 10-16
BAR_WORKER_MAX 4-13, 10-16
BAR_XFER_BUF_SIZE 4-10, 10-17
BAR_XFER_BUFSIZE 4-12, 10-18, 10-19
BAR_XPORT_COUNT 4-12, 10-19
BAR-WORKER_MAX 4-12
DBSERVERNAME 7-31, 11-6
Extended Parallel Server 4-10, 4-12
multiple storage managers 4-10
global 4-11
ISM_DATA_POOL 4-7, 4-10, 4-12, 10-19
ISM_LOG_POOL 4-7, 4-10, 4-12, 10-19
LOG_BACKUP_MODE 4-12
LTAPEDEV 4-10, 10-20
OFF_RECVRY_THREADS 7-14
ON_RECVRY_THREADS 7-14
ontape 13-1
RESTARTABLE_RESTORE 4-10, 7-35, 10-21
RESTORE_FILTER 4-10, 10-21
SERVERNUM 7-31
storage-manager section 4-11
Configuring
ISM 4-2
third-party storage manager 4-1
TSM 4-2
Continuous log backup
example 3-3
pausing 9-10
specifying 1-3, 5-18, 5-23
using ALARMPROGRAM 10-5
Continuous log restore
configuring with ON-Bar 7-25
configuring with ontape 15-9
defined 7-5
versus (High Availability Replication (HDR) 7-5
Controlling ON-Bar sessions 9-10
Cooked chunks
backing up 5-8
restoring 7-23
Copying data 17-2
CREATE EXTERNAL TABLE statement
archecker schema file 17-10
syntax 17-10
CREATE TABLE statement, syntax 17-9
Critical dbspaces
backing up 5-4
defined 5-4
cron command 3-7
Current log, backup 3-3

D

Data
  filtering
    example, schema command file 17-15
    physical restores 17-15
  migration tools C-1
  recovery
    See also Restoring.
    defined 1-1
  usage 2-2
  verifying consistency 5-8
Database logging
  backups 5-11
  log backups 5-18
Database logging status, changing with ontape 14-2
Database servers
  blocking 8-2, 16-1, 16-3
  evaluating 2-3, 4-18
  imported restore 7-29
  migration C-1
  storage-manager communication 3-5
  unblocking 8-4, 16-3
  upgrading 7-29, C-1
  versions with ON-Bar 3-1
DATABASE statement
  declaring logging mode 17-11
  syntax 17-11
DBSERVERNAME configuration parameter 6-4, 11-6
DBSERVERNAME Configuration parameter 7-31
Dbslice
  backing up 1-1
  restoring 7-15
dbspaces
  backing up 1-1
  critical 5-4
  rename
    cold restore 15-3
    warm restore 15-3
  restore selected, ontape 15-2
Debug log, location of 3-11
Debugging
  AC_DEBUG 17-17
  BAR_DEBUG configuration parameter 3-11, 10-9
  BAR_DEBUG_LOG configuration parameter 10-10
  levels of messages in ON-Bar debug log 3-11
  messages 17-17
  ON-Bar activity log 3-11
  ON-Bar utility, levels of 3-11
  with archecker 17-17
Default locale xii
Deferring index rebuild, logical restore 7-24
Deleting a bad backup, ON-Bar 9-8
Dependencies, software xii
Device, ontape
  /dev/null 14-14
  logical–log backup 14-13
  LTAPEDEV parameter 14-14
Disabilities, visual
  reading syntax diagrams E-1
disability E-1
Disaster recovery
  imported restore 7-29
  TSM administration services 3-8
Distributed restore
  performing 17-16
Documentation, types of
  backup and restore tasks 3-2
Dotted decimal format of syntax diagrams E-1
Dropped storage space
  onlog utility 7-23

E

Element, syntax
  See Syntax element.
Emergency boot file xi
  backing up 5-4
  ixbar 3-10
  regenerating 9-7
    IDS 9-7
    XPS 9-7
en_us.8859-1 locale xii, D-1
Environment variables
  AC_CONFIG 6-7, 10-2, 10-3, 10-4
  GL_DATE D-2
  GL_DATETIME D-2
  IFX_ONTAPE_FILE_PREFIX 14-6
  INFORMIXSQLHOSTS 11-6
  ISM_DATA_POOL 4-7
  ISM_LOG_POOL 4-7
  TSM
    setting the interface 4-7
    setting the interface, example of 4-7
Evaluating
  backup and restore time 2-3
  hardware and memory usage 2-3
  logging and transaction activity 2-4, 4-17
Event alarm
  alarmprogram.sh 5-20, 10-5
  setting ALARMPROGRAM 5-20, 10-5
ex_alarm.sh script 10-5
Examples, CREATE TABLE statement 17-9
Expiration policy 9-5
  retention date 9-5
  retention generation 9-5
  retention interval 9-5
Expiring backups
  all 9-9
  generation of 9-8
  multiple Point-in-Time restores 9-8
  on ISM 9-7
  retention date 9-8
  retention interval 9-8
  using onsmsync 9-9
External backup
  blocking database server 3-3, 8-3, 16-3
  defined 8-1, 8-2, 16-1
  mirror chunks 8-9, 8-14
  mirrored support 8-14
  procedure 8-7, 16-2
  recovering data, procedure 8-1
  third-party mirroring solutions 8-10
  tracking backup objects 8-7, 16-3
  unblocking database server 8-4, 16-3
External backup and restore
  cloning 8-1, 16-1
  disk mirroring 8-1, 16-1
  initializing HDR 8-16
  ON-Bar 8-1
  ontape 16-1
  recovering data 8-1, 16-1
External programs
  filters 2-4
  transforming data with 2-4, 3-12

External restore
cold restore 16-4
cold restore procedure 8-13, 16-5
examples 3-3, 8-15, 16-6
initializing HDR during 8-16, 16-6
ON-Bar, renaming chunks 8-16
ontape utility
-e command 16-4
-l command 16-4
-p command 16-4
performing 16-5
restored components 16-4
rules 16-5
salvaging logical logs 8-13
syntax diagram 8-11, 16-4
warm restore procedure 8-15
External tables
CREATE EXTERNAL TABLE statement 17-10
restoring
example 17-15
schema command file example 17-15

F

Fake backups 3-3, 5-16
File directory, ontape
syntax to specify 13-4
Files
logical log 1-2
ON-Bar boot 3-10
temporary, built by archecker 17-19
Filters
for compression 2-4
for data transformation, defined 3-12
restore, with archecker 17-8, 17-18
transforming data 2-4, 3-12
finderr utility 3-11, 12-1

G

GL_DATE environment variable D-2
GL_DATETIME environment variable D-2
Global Language Support (GLS) xii, D-1
GLS.
See Global Language Support.

H

Hardware resources, evaluating 2-3
HDR.
See also High-Availability Data Replication.
external backup and restore 8-16
High-Availability Data Replication
imported restore 7-29
initializing 7-29, 7-33
initializing with external restore 8-16, 16-6

I

I/O
simultaneous backup and restore 15-13
environment variables 15-14
example 15-14
secondary host 15-14
IBM Informix Server Administrator
restoring data 7-13
IBM Informix Storage Manager
configuring 4-2
overview 3-7
sm_versions file 4-4
upgrading C-1
Version 2.2 support xi
IBM Server Administrator
backing up data 5-12
IBM Storage Manager
backup requests 5-11
devices supported 4-17
ISM catalog 5-4, 5-11
requirements 4-16
volume pool names 4-7
IFX_ONTAPE_FILE_PREFIX environment variable 14-6
Imported restore
defined 7-4
ixbar.51 7-30
ixbar.52 7-30
performing 7-30
set up 7-30
Importing a restore 7-30
Importing restore
initializing HDR 7-33
Incremental backup
defined 2-3
example 5-13
level 1 5-7
level 2 5-7
INFORMIXDIR/bin directory xiii
INFORMIXSQLHOSTS environment variable 11-6
Initializing
HDR with imported restore 7-33
High-Availability Data Replication with ON-Bar 7-32
INSERT statements
physical-only restores 17-12
syntax, examples 17-12
ISA.
See IBM Informix Server Administrator.
ISM catalog
backing up 5-11
directory path 5-4
ism_catalog command 5-11, 5-12, 7-8
ism_chk.pl command 7-6
ISM_DATA_POOL configuration parameter 4-7, 4-10, 4-12, 10-19
ISM_LOG_POOL configuration parameter 4-7, 4-10, 4-12, 10-19
ism_startup command 4-5
ism_watch command 5-11, 7-8
ISM.
See IBM Informix Storage Manager.
ISMData volume pool 4-7
ISMLogs volume pool 4-7
ISO 8859-1 code set xii
ixbar.servernum file.
See Emergency boot file.

J

Jukebox 4-16

L

Large files, backup
  ON-Bar utility 5-4
Level-0 archive, open transactions during 17-8
Level-0 backup 1-2
Level-1 backup 1-2, 5-7
Level-2 backup 1-2, 5-7
Light appends 7-34
Locales
  defined xii
  using GLS with ON-Bar D-1
LOG_BACKUP_MODE configuration parameter 4-12
log_full.sh script 5-19
Logging activity, evaluating 2-4, 4-17
Logging mode, declaring in DATABASE Statement 17-11
Logical log
  -b option 5-21
  -l option 5-21
  -n option 5-21
  -P option 5-21
  -q option 5-21
  -t option 5-21
  -u option 5-21
  -x option 5-21
  backing up files 15-9
  backup
    -O option 5-16
    current log 8-8, 16-3
    defined 1-2, 5-18, 5-23
    stopping 5-19
    syntax, IDS 5-19
    syntax, XPS 5-22
  files
    restoring 15-8
    restoring, examples of 15-8
  ON-Bar utility
    automatic backup 5-19, 10-5
    blobspace issues 5-16
    checking available space 5-7
    completion messages 5-23
    continuous backup 1-3, 5-18
    current log backup 3-3
    manual backup 5-18, 5-19, 5-23
    replaying records in parallel 7-14
    salvaging 3-3, 7-16
    skipping replay 7-14
    status of backup 5-18
    when to back up 5-18
  onbar -P 5-20
  ontape utility
    automatic backup, starting 14-12
    backed-up status 14-12
    backup, changing parameters 13-7
    backup, device to use 14-13
    backup, if tape fills 14-9
    backup, on another computer 13-4
    backup, procedure 14-10, 14-11
    backup, separate devices 13-3
    backup, to /dev/null 14-14
    backup, when 14-12
    blobspace blobs 14-11
    continuous backup 14-12
    importance of backing up 1-2
    used status 14-12
  overview
    defined 1-2
    manual backup 1-3
Logical log *(continued)*
  overview *(continued)*
    salvaging 1-3
  salvage, example of 15-7
  viewing backed-up logs 5-20
Logical restore
  See also Restore
  See also Restoring.
  altering or adding tables during 17-4
  deferring index rebuild 7-24
  defined 1-5, 3-3
  ontape utility
    cold restore 15-2
    warm restore 15-3
  procedure 7-14
  tables and fragments not processed by applier during 17-8
logs_full.sh script 5-20, 10-5
LTAPDEV configuration parameter
  ontape utility
    continuous backup to a directory 14-13
LTAPEBLK configuration parameter
  ontape utility 13-2, 13-5
LTAPEDEV configuration parameter
  considerations 7-18
  ON–Bar 4-10, 10-20
  ontape utility
    changing to /dev/null 13-6
    if two tape devices 14-10
    purpose 14-14
    setting 13-2
LTAPESIZE configuration parameter 13-3, 13-5

M

Manual log backup
  example 5-19, 5-23
  specifying 1-3, 5-18
Massively parallel-processing system 4-17
Memory resources, evaluating 2-3, 10-8
Message file
  See also Activity log, ON-Bar
  See also Activity log, ON-Bar.
  usage messages xi, 12-2
Message log
  archecker 17-8
  output 17-19
  path 17-18
Migrating
  data with archecker 17-2
  storage-manager objects 7-30
Migration
  database server C-1
  ontape to ON-Bar C-2
  storage managers C-1
Mirroring configuration
  backup state 15-7
  setting 15-7
Mixbar.hostname.servernum file 3-10
Mixed restore
  defined 1-4
  ON-Bar utility 7-3
  ontape utility 15-3
  point-in-time 7-21
  restoring data 7-18
  strategies for using 7-19

Moving data
See Migrating data.
MPP system 4-17
Multiple tables, restoring 17-16

O

Offline storage spaces, restoring 7-14
ON_RECVRY_THREADS configuration parameter 7-16
ON-Bar
boot files 3-10
ON-Bar activity log.
See Activity log.
ON-Bar return codes 12-6
ON-Bar tables xi
bar_action 11-2
bar_instance 11-4
bar_ixbar 11-6
bar_object 11-6
bar_server 3-9
defined 11-8
map xi, 11-7
ON-Bar utility
-O option
backup 5-9, C-2
restore 5-16, 7-12, 7-22
whole-system restore 7-10
See also Configuration parameter
activity log
See Activity log.
archecker
setting configuration parameters 10-2
backup and restore time 2-3
backup sequence
Dynamic Server 5-24
Extended Parallel Server 5-24
cold restore sequence 7-13, 7-43
compared to ontape 1-7, 7-41
components on
Dynamic Server 3-4
Extended Parallel Server 3-4
configuration parameters 4-9, 4-12, 10-5, 10-21
database servers supported 3-1, 3-2
debugging
specifying the level of 3-11
deleting bad backups 9-8
external backup and restore 8-1
migrating from ontape C-2
mixed restore 7-3
monitoring restores
onstat -d 7-7
operations supported 1-7
parallel restore 7-4
planning recovery strategy
creating backup plan 1-7
data loss 2-2
data usage 2-1
failure severity 2-2
pre-recovery checklist 7-6
progress feedback 3-11
renaming chunks 7-5
restore
-e option 7-9
-f option 7-9
-i option 7-9
-I option 7-9
-n option 7-9
ON-Bar utility *(continued)*
restore *(continued)*
-O option 7-10
-q option 7-10
-r option 7-8
-t option 7-10
-w option 7-10
examples 7-13
processes 7-40
starting and stopping sessions 9-10
types of restores 7-2
usage messages 12-2, 12-5
warm restore sequence 7-40, 7-42
where to find information 3-2
whole-system restore 7-4
XBSA interface 3-9
onbar script
defined 9-2
usage and examples 11-2
onbar_d utility
See also onbar-driver
purpose xi
onbar_m utility
defined 3-7
ON-Bar diagram 3-6
purpose 5-26
onbar_w utility
default setting 9-10
defined 3-3, 3-7, 9-10
ON-Bar diagram 3-6
purpose 5-26
starting onbar-workers 9-4
onbar-driver
child process 5-24
defined 3-4, 3-7
oncfg file 5-4
ONCONFIG file
See Configuration parameter.
oninit -m command 7-17
Online storage spaces, restoring 7-22
onmode -c
block command 8-4, 16-3
unblock command 8-4, 16-3
onmode -d command 7-32
onmode -ky command 7-17
onmode -l command 5-16, 5-23
onmode -m command 7-18, 7-35
onmode -sy command 7-18
onsmsync utility
archecker 6-7, 6-8
defined 3-3
procedure 9-6
using 9-5
onsmsync, expiring old backups 9-9
onstat -d command 7-7, 7-15, B-1
onstat -g bus command 5-27, 9-11, B-3
onstat -g bus_sm command 5-27, 9-12, B-3
onstat -l command 5-16, 5-24, B-3
onstat command B-1, B-5
ontape utility
-C option 15-4
-d option 14-8
-D option 15-4
-F option 14-8
-L option 14-8
-s option 14-6
-t option 14-8

ontape utility *(continued)*
-v option 14-8
-X option 15-5
backing up logical log 14-10
backup levels 14-3
backups and modes 14-4
changing
database logging status 14-2
LTAPEDEV to /dev/null 13-6
parameters 13-6
TAPEDEV to /dev/null 13-6
checking configuration parameters 13-5
compared to ON-Bar 1-7
configuration parameters 13-1
creating an archive 14-3
device name 17-2
ensuring adequate memory 14-6
example 17-19
exit codes 14-9
external backup and restore 16-1
fake backup 14-8
labelling backup tapes 14-4
migrating to ON-Bar 14-4
operations supported C-2
option
-D 1-7
-L 15-5
-r 14-8
-s 15-5
overview 16-4
parameters
changing 14-8
overriding 14-8
parameters, changing 13-6
performing a restore 15-4
physical restore, choosing type 15-2
precautions, one tape device 15-2
premature backup termination 14-2
read to end of tape 14-12
restore, choosing mode 13-3, 15-2
restoring data 15-2
standard output 14-8
starting continuous backup 14-11
syntax
full backup 14-12
logical-log backup 14-1
writing to end of tape 14-12
onutil 3-3, 5-14, 6-6, 8-4
Operational table
backing up 5-17
restoring 7-34
Output, message log 17-19
overriding 17-18
Overriding internal checks
backup 3-3, 5-9, 5-16
restore 3-3, 7-10, 7-12, 7-22
whole-system restore 7-18

P

Page size, online 17-18
Parallel backup
backup sequence 5-25
BAR_MAX_BACKUP 5-5, 5-25
BAR_WORKER_MAX 10-17
defined 5-5
imported restore 7-4
Parallel backup *(continued)*
specifying 10-12
Parallel restore
BAR_WORKER_MAX 10-17
defined 7-4
example 3-3
ON-Bar utility 7-4
performing 17-8
specifying 10-12
Parameter, configuration
See archecker configuration file.
Performing backup 14-6
Performing external backup
mirror chunks 8-14
Performing imported restore 7-30
Physical backup 5-6
defined 5-9
example 3-3, 5-17
Physical restore
See also Restoring.
defined 3-3
ON-Bar utility, example 7-15
ON-Bar utility, procedure 15-2
ontape utility, cold restore 1-6
types xi
Physical schema, backing up 5-10
Planning a backup system 2-3
pload utility 5-17
Point-in-log restore 3-3, 7-22
Point-in-time restore
-r option 7-21
-t option 7-21
cold 7-20, 7-21
creating timelines 7-21
defined 7-4
example 3-3, 7-20, D-2
expiring bad backup 7-22
mixed 7-21
restore from older backup 7-22
specifying timelines 7-21
warm restore 7-21
Processes, ON-Bar 5-24, 5-26
Progress monitoring, backup or restore 3-11

R

Raw chunks
backing up 5-8
restoring 7-23
Raw table
backing up
ON-Bar utility 5-17
ontape utility 14-9
restoring 7-34, 15-9
Recovering data
See Restoring data.
Recovery strategy planning
creating backup plan 2-2
data loss 2-1
data usage 2-2
failure severity 2-1
Recovery system
comparing ON-Bar and ontape 1-1, 1-7
defined 1-1
invalid tools 7-15
Recovery, tables or fragments added or created during 17-8
Recreating chunk files 7-22

Remote device, ontape
interrupt key 14-13
syntax to specify 13-4
tape size 13-5
Rename chunks restore
defined 7-5, 15-3
ON-Bar
external 7-27
restartable 3-3
ON-Bar, external 8-16
ontape syntax 15-10
overview with ontape 15-10
Replaying log records 1-3, 7-14
Restartable restore
example 3-3, 7-35
overview 7-5
using 7-35
RESTARTABLE_RESTORE configuration parameter 4-10, 7-35, 10-21
Restore
Bringing the database back online 15-8
considerations 15-11
dropped storage space 7-23
failed restore 7-38
resolving 7-38
scenarios 7-39
from an older backup 7-22
logical 17-4
logical, manually controlling 17-7
mounting tapes 15-8
multiple storage managers 17-7
new-chunk requirements 7-26
nonlogging databases and tables 7-33
physical 17-4
point-in-time, cold 7-20
rename chunk restore with cold restore 7-5
restarting 7-35
standard input 15-13
example 15-13
RESTORE statement, syntax 17-13
RESTORE_FILTER
about 2-4
configuration parameter 4-10, 10-21, 13-2
defining in onconfig 3-12
ON-Bar utility 10-21
ontape utility 13-2
transforming data with 3-12
unique server names 3-12
Restore, logical
See Logical restore.
Restoring
ON-Bar utility
-O option 3-3, 7-12, 7-18, 7-22
cold, example 7-20
cold, setting mode 7-16
cooked chunks 7-23
dbslices 7-8
external 3-3, 8-11, 8-16
imported 7-4
logical, example 7-14
mixed 7-18
monitoring progress 3-11
offline storage spaces 7-8
online storage spaces 7-16
operational tables 7-22
parallel 7-34
physical, example 3-3
Restoring *(continued)*
ON-Bar utility *(continued)*
point-in-log 3-3
point-in-time example 7-20, 7-22
raw chunks D-2
raw tables 7-23
renaming chunks 7-26, 7-34
restartable 3-3, 7-35
scratch tables 7-34
selected spaces, example 7-35
standard tables 5-17
static tables 7-34
syntax 7-10, 7-27
table types 6-3, 7-33
temp tables 7-34
using ISA 7-13, 7-34
warm 7-13
whole system 7-13
ontape utility
-C option 15-4
-D option 15-4
-e option 15-4
-f option 15-5
-l option 15-5
-p option 15-5
-v option 15-5
-X option 15-5
full-system 3-3
selected storage spaces 15-2, 15-8
whole system, steps 15-5
overview
defined 1-5, 15-5
logical phase 1-4, 1-5
physical phase 1-6
Restoring data 7-26, 17-2
archecker 17-3
mixed restore 7-18
point-in-time 17-3
Return codes 12-6
Rewindable tape device 2-3, 13-5
Rixbar.hostname.servernum file 3-10
Root dbspace
onbar -r command 12-6
when to back up 3-10, 5-10

S

Salvaging logs
defined 5-24
example 1-3, 3-3
skipping 7-16
Sample-code conventions 7-16
Save sets
defined 5-11
restoring 5-8
sbspaces
archecker 6-4
backing up 5-15, 7-8
verifying 6-6
Scheduling backups 2-4, 5-10
Schema command file
example 17-15
restoring to a different a table 17-14
restoring to an external table 17-15
restoring to multiple tables 17-15
simple 17-16
using data filtering 17-14

Schema command file *(continued)*
setting 17-14
Schema command file example
extracting a subset of columns 3-8
performing a distributed restore 17-15
restoring a table from previous backup 17-16
Schema file
archecker 17-3
CREATE EXTERNAL TABLE statement 17-10
Scratch table
backing up 17-3
restoring 5-17
Screen reader
reading syntax diagrams E-1
Server locale D-1
SERVERNUM configuration parameter 7-31
Session name, assigning to backup 5-22
SET statement
examples 17-13
syntax 17-13
Severity of data loss 17-13
Shared files for ON-Bar utility, ISM, TSM 10-22
Shared libraries, XBSA
default location 2-1
specifying location 4-8, 10-8
Shared-memory parameters
setting to maximum assigned value 15-6
shortcut keys
keyboard E-1
Silos 4-8
Skipping
log salvage 1-3
logical replay 7-16
storage spaces 7-14
sm_versions file
backing up 5-8, 5-15
defining storage manager 5-4
Smart large objects
backing up 4-4
Restoring 7-34
SMI tables
See Backup Scheduler
Software dependencies xi
sqlhosts file
copying 5-4, 5-8
server name 5-8
Staged point-in-time restore 7-21
Stager, manually controlling logical restore using 11-6, 17-4
Standard backup
See Backup.
Standard input
overriding TAPEDEV 15-5
restoring 15-13
example 15-13
single-user mode 15-13
setting TAPEDEV 13-5
simultaneous backup and restore 15-14
Standard output
backing up to 14-6
setting TAPEDEV 13-5
simultaneous backup and restore 15-14
Standard table
backing up xi
restoring 5-17
start_worker.sh script 9-4
Static table
backing up 7-34
Static table *(continued)*
restoring 5-17
Storage devices
backups 6-2, 7-14, 7-23
continuous log backups 5-19
ISM support 5-11
ISM volume pools 4-17, 5-11
requirements 4-15
Storage manager xi
communication with ON-Bar 3-5, 4-17
configurations 3-5, 4-14
migration 4-12, C-1
onbar -P
viewing backed-up logical logs 5-20
requirements 5-5, C-1
role in ON-Bar system 4-15
sm_versions file 4-2
Storage spaces
backing up a list 5-8
backing up all 5-8, 5-13
defined 3-3
offline, restoring 5-13
physical-only backup 5-17
restartable restore 7-35
restoring 1-5, 7-14, 7-35
skipping during backup 7-14
when to back up 5-15, 14-12
stores_demo database xii
superstores_demo database xii
Symbolic links
chunking names 15-11
ontape utility
specify tape devices 13-3
specify tape size 13-5
restoring chunk files 7-23
Syntax diagrams
backup verification 6-3
conventions 5-9
external backup
IDS 8-3
XPS 8-3
external restore 8-4, 8-12, 16-3
keywords 5-2
logical-log backup 5-2, 5-8
onbar_w 9-10
onsmsync 9-6
reading in a screen reader E-1
restore 7-10, 7-27
starting and stopping sessions 9-10
storage-space backup 5-2
summary 5-2
sysbuobject table 11-8
sysbuobjses table 11-8
sysbusession table 11-8, 11-9
sysbusm table 11-8, 11-9
sysbusmdbspace table 11-8, 11-10
sysbusmlog table 11-8, 11-10
sysbusmworker table 11-8, 11-10
sysbuworker table 11-8, 11-10
System requirements
database 11-11
software 11-11
System time changes, and backups A-3
sysutils database
error messages 10-22
regenerating 9-8

T

Table types
  backing up 5-11, 5-17
  operational 5-11, 7-34
  raw 5-17, 7-34
  restoring 7-33, 7-34
  scratch 5-17, 7-34
  standard 5-17, 7-34
  static 5-17, 7-34
  temp 5-17, 7-34
Tables
  See also Logical restore.
  altering
    or adding during logical restore 17-8
    raw 7-34
    standard 10-22
  restoring to user-specified point in time 17-8
Tables and fragments
  added or created during logical recovery 17-8
  not processed by applier during logical restore 17-8
Tape autochangers 4-16
Tape block size
  ON-Bar utility 17-4
  ontape utility 17-18
Tape device
  ontape utility
    before opening and on closing 13-5
    block-size parameters 13-5, 17-18
    gathering for cold restore 13-5
    gathering for warm restore 13-5
    parameters, setting 13-1
    precautions with only one 13-3
    reading to end 13-3
    rewindable device 13-3
    specifying symbolic links 13-5
    using /dev/null 13-5
  setting TAPEDEV to stdio 13-5
  writing to end 13-3
Tape libraries 13-4
TAPEBLK configuration parameter
  ontape utility 4-16, 13-5
    defined 13-2
TAPEDEV configuration parameter
  ontape utility 13-2
    changing to /dev/null 13-5
    if two tape devices 14-9
TAPESIZE configuration parameter
  ontape utility 13-6
    defined 13-2
Task-documentation matrix 3-2
Temp table
  backing up 14-4
  restoring 5-17
Temporary spaces 7-34
Text editor, changing ontape parameters 13-6
Third-party mirroring solutions, external backup 8-10
Third-party storage manager
  configuring 4-2
  functions 4-1
  onbar script 4-1
Tivoli Storage Manager
  See TSM
Transaction activity
  evaluating 2-4, 4-17, 17-8
Transactions
  open during level-0 archive 3-8, 17-8
  projecting with logical log backups 1-3
transforming data with filters 2-4, 3-12, 17-8, 17-18
TSM
  Administration services 3-8
  and Network Data Management Protocol 4-16
  Backup and restore services 3-8
  client system options
    dsm.sys 4-3
  configuring 4-2
    client options files 4-3
    client system options 4-3
    client user options 4-3
  defined 3-8, 4-2
  environment variables
    setting the interface 4-7
  Informix interface
    initializing passwords 4-4
  management class
    assigning a backup 4-3
    INCLUDE option 4-3
  Managing resources 3-8
  registering 4-4
  sm_versions 4-5
  Specifying XBSA library location 4-8
  storage manager version, defining 4-5
  supported features 4-16
  Unsupported features 4-17

U

Unblocking database server 8-3, 8-4
UNIX operating system
  bargroup 4-6
  copying files 4-6, 4-9, 8-9
Upgrading database server 8-8, C-3
Using CREATE EXTERNAL TABLE statement 17-10
Using CREATE TABLE statement 17-9
Using DATABASE statement 17-11
Using mixed restore, strategies 7-18
Utilities
  archecker 6-1, 6-2, 7-19, C-1
  onsmsync 6-7, 6-8

V

Variables, in syntax diagrams 9-9
Verification, backup 6-1
Verifying raw devices 15-7
Virtual processors, ON-Bar utility 2-3
Visual disabilities
  reading syntax diagrams E-1
Volume pools
  backup locations 4-10
  default names 4-7

W

Warm restore
  defined 1-4, 7-2
  ON-Bar utility
    examples 7-15
    sequence 7-13, 7-40
  ontape utility
    critical dbspaces 15-9
    defined 15-3, 15-9
    mixed restore 15-3
    part of a mixed restore 15-3

Warm restore *(continued)*
ontape utility *(continued)*
performing 15-6
steps to perform 15-3
overview 15-9
Whole-system backup
defined 5-5
specifying 5-10, 5-25
Whole-system restore
-O option 7-12, 7-18
defined 7-4
example 5-10, 7-17
ON-Bar utility 7-4
restartable restore 7-17
syntax 7-5, 7-18
Windows
-Admin group 5-2
copying files 8-9, 8-10

X

XBSA interface, described 3-9
XBSA shared library
default location 4-8, 5-1
specifying location 3-9, 4-8
xcfg file 4-8

Printed in USA

SC23-7756-02

Spine information:

IBM Informix **Version 11.50**

IBM Informix Backup and Restore Guide